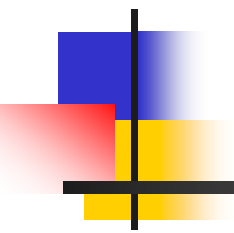
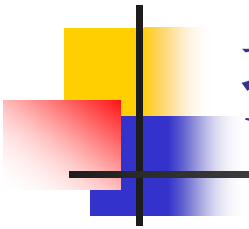


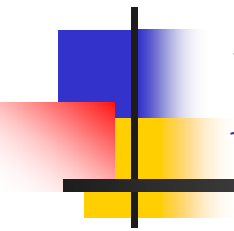
文本分类
(Text Categorization)

刘挺
哈工大信息检索研究室
2004年秋



提纲

- 文本分类概述
- 特征提取
- 主要分类算法
  - Rocchio 法
  - 贝叶斯
  - K近邻
  - 决策树



文本分类概述

分类的概念

- 给定:
  - 一个实例的描述, $x \in X$, X是实例空间
  - 一个固定的文本分类体系: $C=\{c_1, c_2,\ldots c_n\}$
  - 由于类别是事先定义好的，因此分类是有指导的（或者说是有监督的）
- 确定:
  - 实例x的类别 $c(x) \in C$, $c(x)$ 是一个分类函数，定义域是 X，值域是C

说明

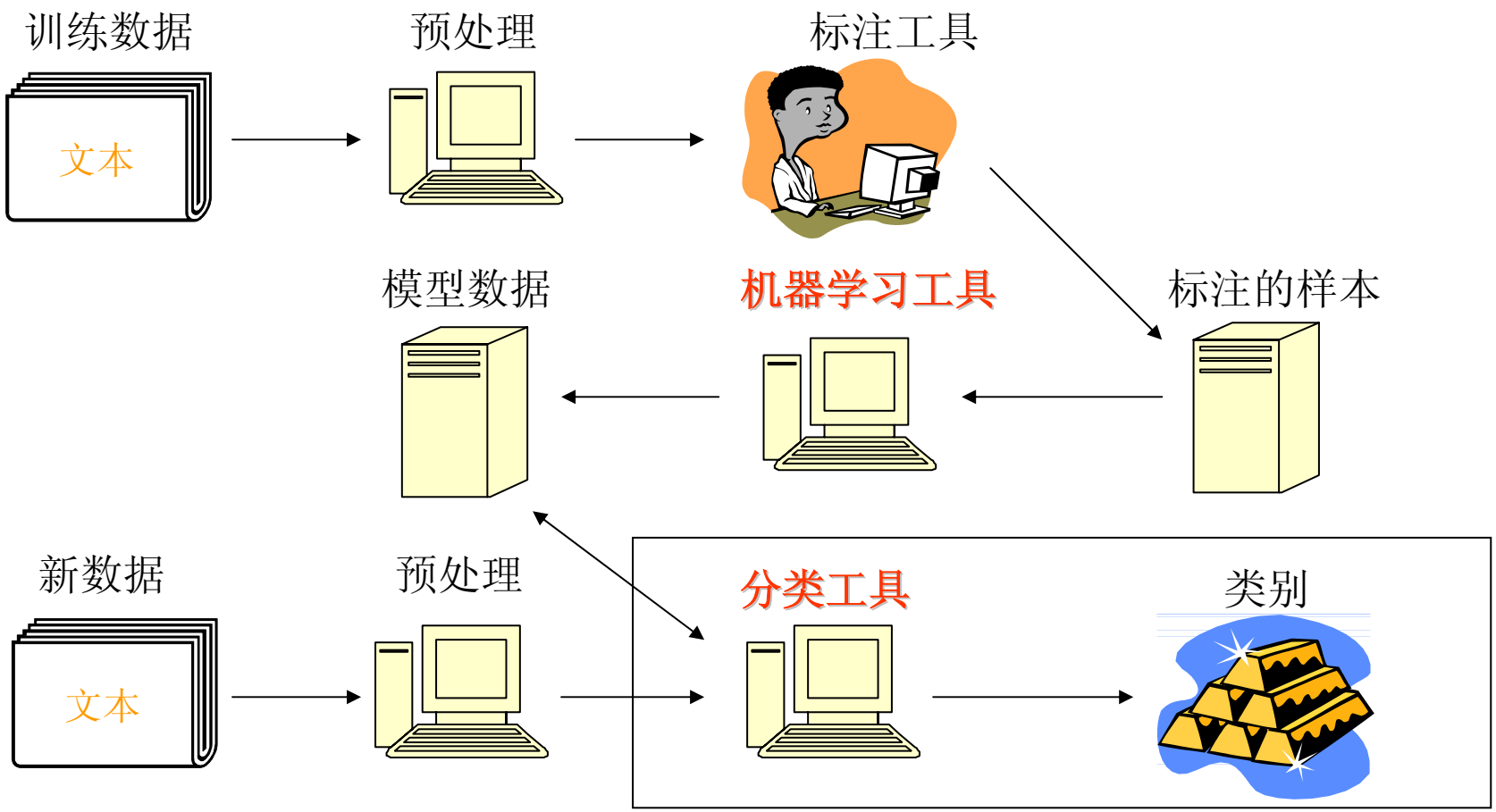
- 分类模式
  - 2类问题，属于或不属于(binary)
  - 多类问题，多个类别(multi-class)，可拆分成2类问题
  - 一个文本可以属于多类(multi-label)
- 分类体系一般人工构造
  - 政治、体育、军事
  - 中美关系、恐怖事件
  - 很多分类体系: Reuters分类体系、中图分类

中图分类法

A类　马列主义、毛泽东思想
B类　哲学
C类　社会科学总论
D类　政治、法律
E类　军事
F类　经济
G类　文化、科学、教育、体育
H类　语言、文字
I类　文学
J类　艺术
K类　历史、地理
N类　自然科学总论
O类　数理科学和化学
P类　天文学、地球科学
Q类　生物科学
R类　医药、卫生
S类　农业科学
U类　交通运输
V类　航空、航天
X类　环境科学、劳动保护科学（安全科学）

TB类　一般工业技术
TD类　矿业工程
TE类　石油、天然气工业
TF类　冶金工业
TG类　金属学、金属工艺
TH类　机械、仪表工艺
TJ类　武器工业
TK类　动力工业
TL类　原子能技术
TM类　电工技术
TN类　无线电电子学、电信技术
TP类　自动化技术、计算技术
TQ类　化学工业
TS类　轻工业、手工业
TU类　建筑科学
TV类　水利工程

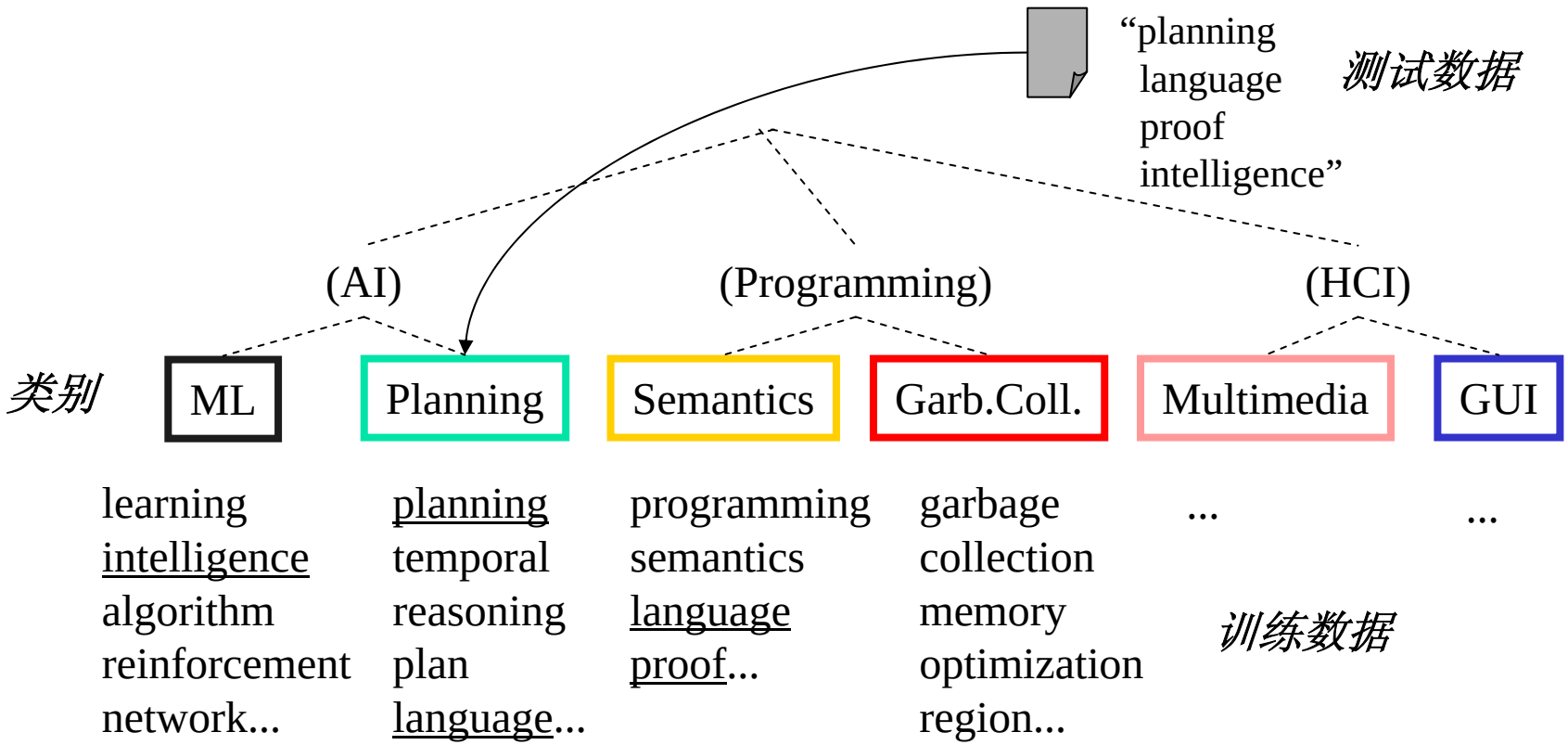
系统结构

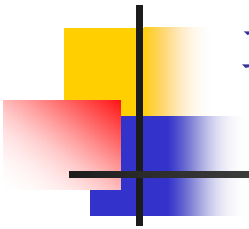


分类的一般过程

- 收集训练集和测试集，对文本进行预处理
- 对文本类别进行人工标注
- 对文本进行特征提取
- 训练（学习）
- 评价
  - 精确率、召回率、F1
  - 宏平均，微平均

文本分类示例


“planning
language
proof
intelligence”
测试数据
(AI)
(Programming)
(HCI)
类别
ML
Planning
Semantics
Garb.Coll.
Multimedia
GUI
learning
intelligence
algorithm
reinforcement
network...
planning
temporal
reasoning
plan
language...
programming
semantics
language
proof...
garbage
collection
memory
optimization
region...
...
...
训练数据



预处理

- 去掉网页中的导航信息
- 去掉HTML网页中的tag标记
- (中文)分词、词性标注、短语识别、…
- 去除停用词和词根还原(stemming)
- 数据清洗：去掉不合适的噪声文档或文档内垃圾数据
- ……

特征提取(Feature Selection)

- 特征提取
  - 在文本分类问题中遇到的一个主要困难就是高维的特征空间。
  - 通常一份普通的文本在经过文本表示后，如果以词为特征，它的特征空间维数将达到几千，甚至几万。
  - 大多数学习算法都无法处理如此大的维数。
  - 为了能够在保证分类性能的前提下，自动降低特征空间的维数，在许多文本分类系统的实现中都引入了特征提取方法。

学习

- 训练样本实例：$<x, c(x)>$
  - 一个文本实例 $x \in X$
  - 带有正确的类别标记 $c(x)$

- 学习的过程是在给定训练样本集合D的前提下，寻找一个分类函数$h(x)$, 使得:

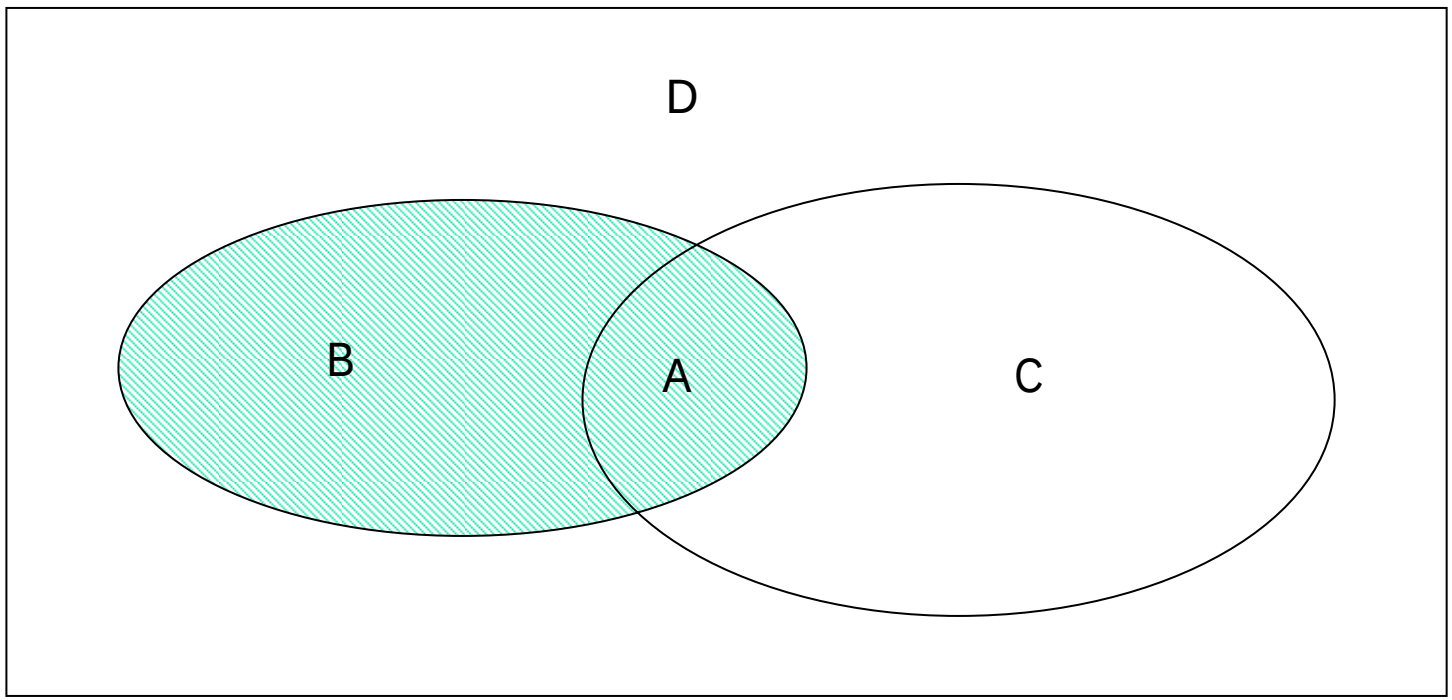
$$\forall < x, c(x) > \in D : h(x) = c(x)$$

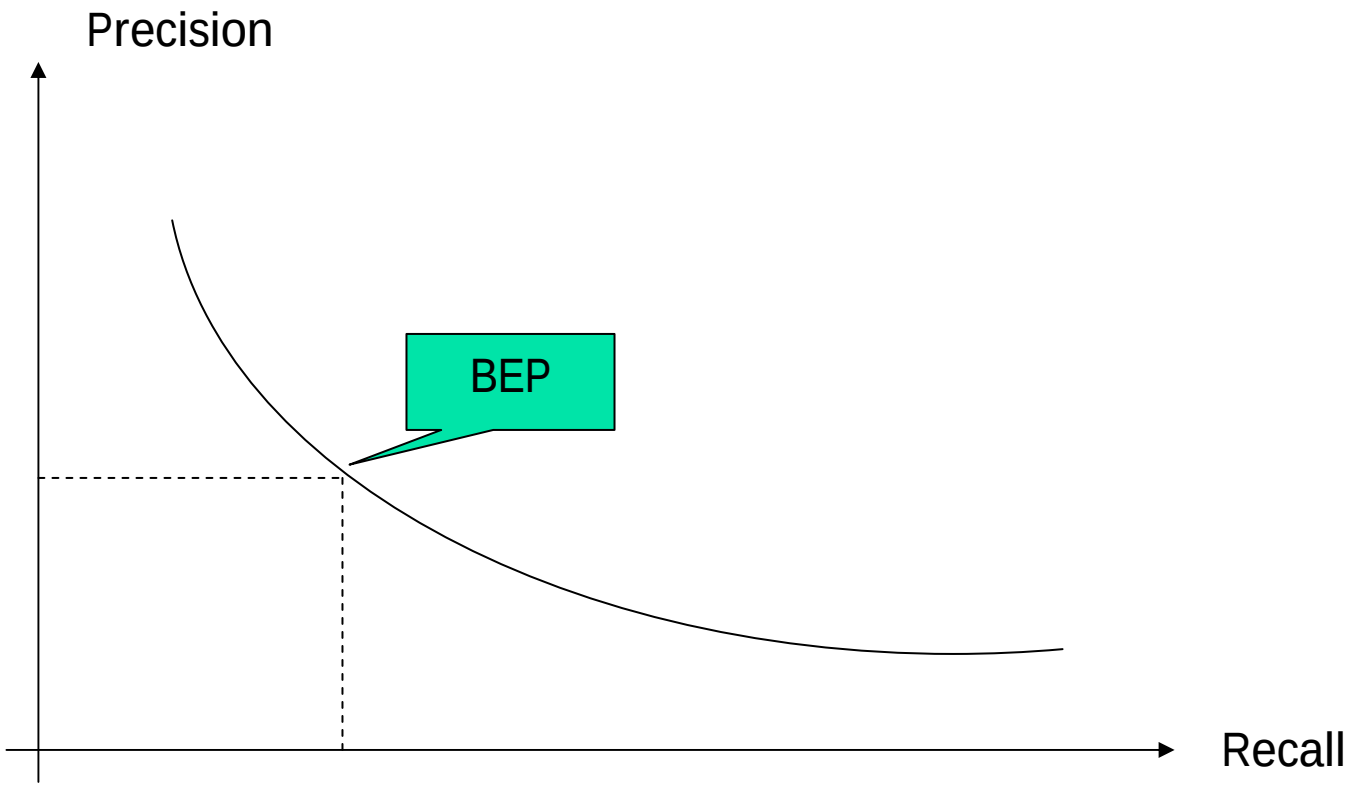
分类的评测

- 偶然事件表（Contingency Table）

	属于此类	不属于此类
判定属于此类	A	B
判定不属于此类	C	D

- 对一个分类器的度量
  - 准确率(precision) = a / (a + b)
  - 召回率(recall) = a / (a + c)
  - fallout = b / (b + d)


D
B
A
C

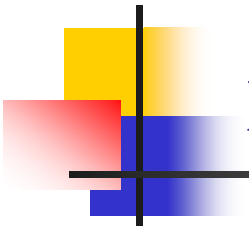

Precision
BEP
Recall

BEP和F测度

- BEP（break-even point）
  - 当准确率和召回率相等时的值即为BEP
- F测度，取 $\beta=1$

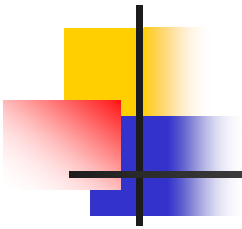
$$F_\beta(p,r)=\frac{(\beta^2+1)pr}{\beta^2 p+r} \qquad F_1=\frac{2\ pr}{p\ +\ r}$$

- BEP和F测度的值越大，则表示分类器的性能越好。
- BEP只是F1所有可能取值中的一个特定值（当p = r时），因此BEP小于或等于F1的最大值。



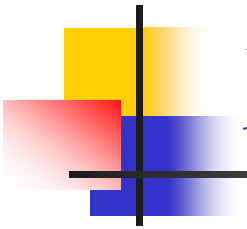
多类分类问题的评价

- 宏平均（macro-averaging）
  - 先对每个分类器计算上述量度，再对所有分类器求平均
  - 是关于类别的均值
- 微平均（micro-averaging）
  - 先合并所有分类器的偶然事件表中的各元素，得到一个总的偶然事件表，再由此表计算各种量度。
  - 是关于文本的均值



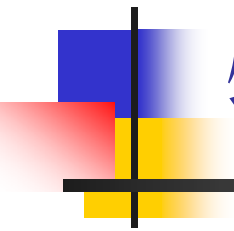
层次分类

- 分类系统可以是层次结构
  - 如Yahoo、中图分类
- 将文本分到一个类别体系(topic hierarchy)中
  - 通常是一个叶结点
  - 有时是中间结点
- 层次分类的方法
  - 一系列 N-way 的分类决策
    - 从分类体系的根部开始，每次选择最好的子类
    - 在不同的分支处，有不同的特征提取
    - 数据更充分，但是可能引入错误累积
  - 单独的 N-way 分类决策
    - 简单地从所有可能的最终类别中选择最佳类别

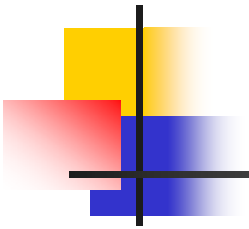


文本分类的应用

- 新闻出版按照栏目分类
  - 类别 {政治,体育,军事,...}
- 网页分类
  - 类似于Yahoo的分类
- 个性化新闻
  - 智能推荐
- 垃圾邮件过滤
  - 类别 {spam, not-spam}



特征提取



举例

- 对每类构造*k*个最有区别能力的term
- 例如：
  - *计算机领域：*
    - *主机、芯片、内存、编译 …*
  - *汽车领域:*
    - *轮胎，方向盘，底盘，气缸，…*

文本表示

- 向量空间模型(Vector Space Model)
  - M个无序标引项t_i(特征)
    - 词根/词/短语/其他
  - 每个文档d_j可以用标引项向量来表示
    - $(a_{1j},a_{2j},\ldots,a_{mj})$
  - 权重计算，N个训练文档
    - $A_{M*N}=(a_{ij})$
  - 相似度比较
    - Cosine计算
    - 内积计算

Term的粒度

- 字(Character)：中
- 词(Word)：中国
- 短语(Phrase)：中国人民银行
- 概念(Concept)：
  - 同义词：开心/高兴/兴奋
  - 相关词词簇(word cluster)：葛非/顾俊
- N-gram(N元组)：
  - 中国/国人/人民/民银/银行
- 某种规律性模式：比如某个window中出现的固定模式
- David Lewis等一致地认为：(英文分类中)使用优化合并后的 Words比较合适

用文档频率选特征

- 词频
  - *TF (Term Frequency)*
  - $TF_{i,j}$：特征*i*在文档*j*中出现次数
- 文档频率
  - *DF (Document Frequency)*
  - *DFi*：所有文档集合中出现特征*i*的文档数目
- 基本假设：稀少的词或者对于目录预测没有帮助，或者不会影响整体性能。
- 实现方法：先计算所有词的DF，然后删除所有DF小于某个阈值的词，从而降低特征空间的维数。
- 优缺点：
  - 最简单的降低特征空间维数的方法
  - 稀少的词具有更多的信息，因此不宜用DF大幅度地删除词

权重计算方法

- 布尔权重(boolean weighting)
  - a_{ij}=1(TF_{ij}>0) or (TF_{ij}=0)0
- *TFIDF*型权重
  - *TF*: $a_{ij}=TF_{ij}$
  - *TF*IDF*: $a_{ij}=TF_{ij}*log(N/DF_i)$
  - *TFC*: 对上面进行归一化
  - *LTC*: 降低TF的作用

$$a_{ij}=\frac{TF_{ij}*\log(N/DF_i)}{\sqrt{\sum_k[TF_{kj}*\log(N/DF_k)]^2}}$$

$$a_{ij}=\frac{\log(TF_{ij}+1.0)*\log(N/DF_i)}{\sqrt{\sum_k[\log(TF_{kj}+1.0)*\log(N/DF_k)]^2}}$$

信息增益

- term的熵
  - 该值越大，说明分布越均匀，越有可能出现在较多的类别中；
  - 该值越小，说明分布越倾斜，词可能出现在较少的类别中

$$Entropy(t) = -\sum_i P(c_i \mid t) \log P(c_i \mid t)$$

- 信息增益(Information Gain, IG)：
  - 该term为整个分类所能提供的信息量
  - 不考虑任何特征的熵和考虑该特征后的熵的差值
  - 信息增益计算的是已知一个词 t 是否出现在一份文本中对于目录预测有多少信息。
  - 这里的定义是一个更一般的、针对多个目录的定义。

信息增益

$$\begin{aligned}\text{Gain(t)} &= \text{Entropy}\ (S) - \text{Expected}\ \ \text{Entropy}(\ S_t) \\ &= \{-\sum_{i=1}^{M} P(c_i)\log P(c_i)\} - \\ &\quad [P(t)\{-\sum_{i=1}^{M} P(c_i \mid t)\log P(c_i \mid t)\} + \\ &\quad P(\bar{t})\{-\sum_{i=1}^{M} P(c_i \mid \bar{t})\log P(c_i \mid \bar{t})\}]\end{aligned}$$

$$G(t) = P_r(t)\sum_i P_r(c_i \mid t)\log\frac{P_r(c_i \mid t)}{P_r(c_i)} + P_r(\bar{t})\sum_i P_r(c_i \mid \bar{t})\log\frac{P_r(c_i \mid \bar{t})}{P_r(c_i)}$$

t 出现的概率

t 不出现

假定 *t* 出现时取第i个目录的概率

取第 i 个目录时的概率

交叉熵（Cross Entropy）

- 相对熵：也称为KL距离(Kullback-Leibler divergence)，反映了文本类别的概率分布和在出现了某个特定词汇条件下的文本类别的概率分布之间的距离，该值越大，词对文本类别分布的影响也大。

$$CE(t) = \sum_i P(c_i \mid t) \log \frac{P(c_i \mid t)}{P(c_i)}$$

- 交叉熵的定义与信息增益近似，不同之处在于交叉熵只考虑一个词t出现时的影响。它的定义为：

$$C(t) = P_r(t) \sum_i P_r(c_i \mid t) \log \frac{P_r(c_i \mid t)}{P_r(c_i)}$$

互信息（Mutual Information）

- 互信息(Mutual Information)：MI越大t和c共现程度越大
- 互信息的定义与交叉熵近似，只是互信息不考虑t出现的概率，它的定义为：

$$I(t,c)=\log\frac{P(t\wedge c)}{P(t)P(c)}=\log\frac{P(t\mid c)}{P(t)}=\log\frac{A\times N}{(A+C)(A+B)}$$

$$I_{AVG}(t)=\sum_{i=1}^{m}P(c_i)I(t,c_i)$$

$$I_{MAX}(t)=\max_{i=1}^{m}P(c_i)I(t,c_i)$$

x ²统计量(念CHI)：

- χ2统计量的定义可以从一个词*t*与一个目录*c*的偶然事件表引出（假设文本的总数为N ）

	c	~c
t	*A*	*B*
~t	*C*	*D*

$$\chi^2(t,c)=\frac{N(AD-CB)^2}{(A+C)(B+D)(A+B)(C+D)}$$

- 度量两者(term和类别)独立性的缺乏程度
  - x ² 越大，独立性越小，相关性越大
  - 若AD<BC,则类和词独立, N=A+B+C+D

$$\chi^2_{MAX}(t)=\max_{i=1}^{m}\{\chi^2(t,c_i)\}$$

$$\chi^2_{AVG}(t)=\sum_{i=1}^{m}P(c_i)\chi^2(t,c_i)$$

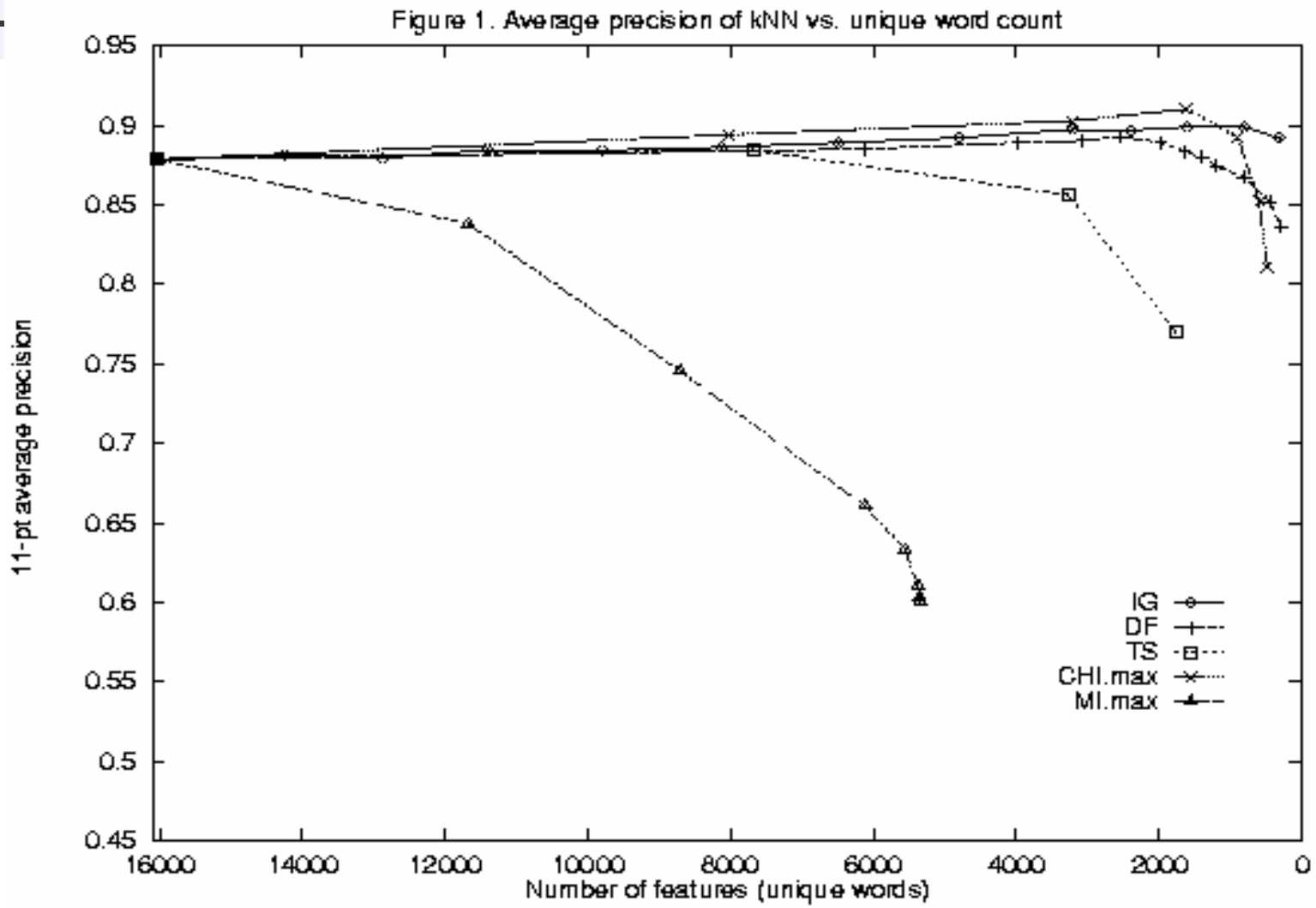
几率比（Odds Ratio）

- 几率比是一种在信息检索中广泛使用的方法，它的定义是：

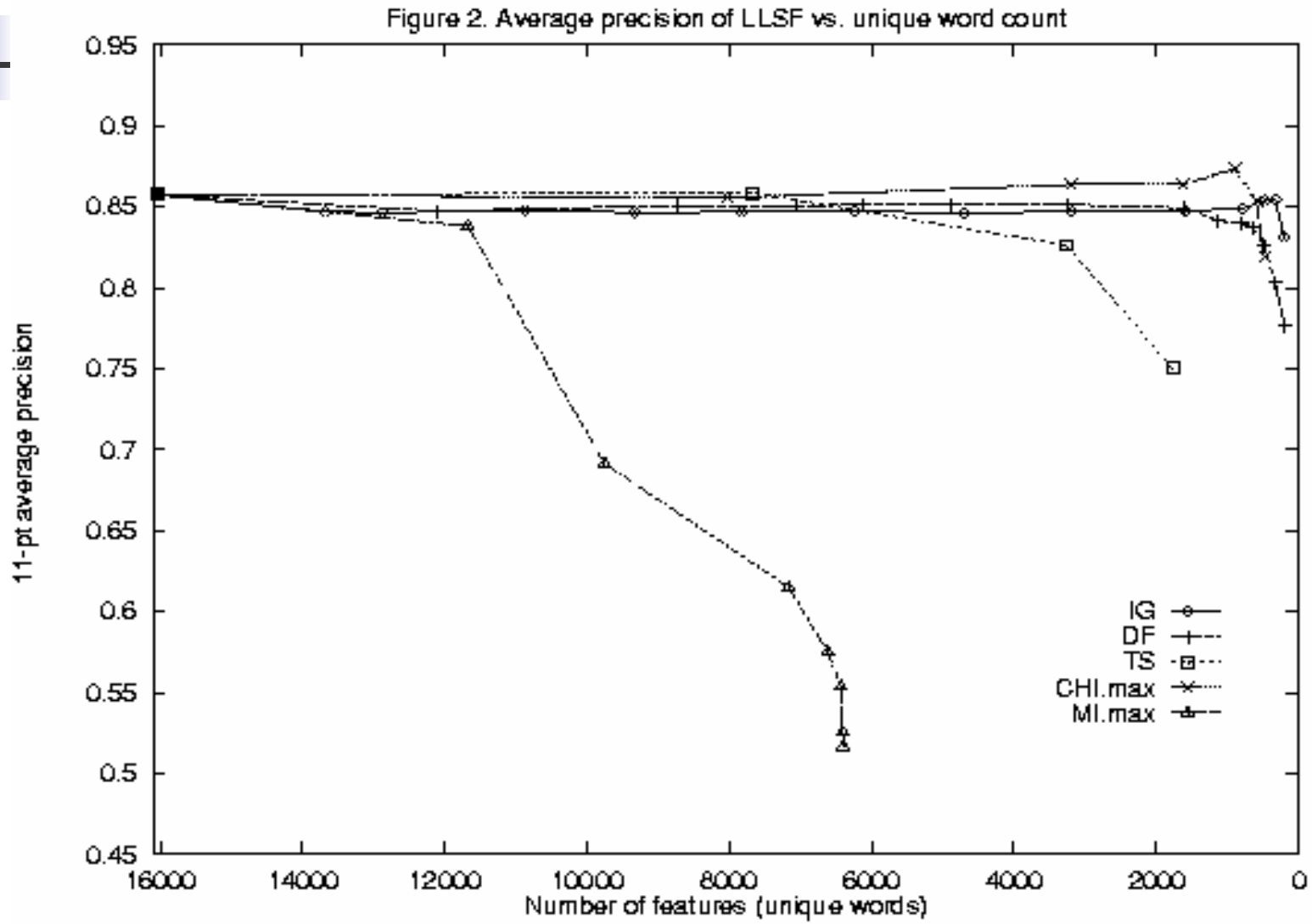
$$OddsRatio(t) = \log \frac{\Pr(t \mid pos)(1 - \Pr(t \mid neg))}{(1 - \Pr(t \mid pos))\Pr(t \mid neg)}$$

  - *Pr(t/pos)*为已知目录值为正“positive”时*t*出现的条件概率
  - *Pr(t/neg)*为已知目录值为负“negative”时*t*出现的条件概率。

特征提取方法的性能比较(1)



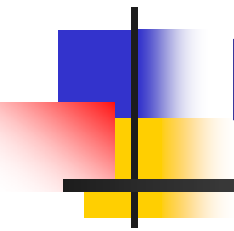
特征提取方法的性能比较(2)



特征提取方法的性能比较(3)

YangYi-ming, CMU, USA

Method	DF	IG	CHI	MI	TS
favoring common terms	Y	Y	Y	N	Y/N
using categories	N	Y	Y	Y	N
using term absence	N	Y	Y	N	N
performance in kNN/LLSF	excellent	excellent	excellent	poor	ok



Rocchio 法

回忆相关反馈

以上的query修改公式是建立在已知全部相关文档集和不相关文档记得基础上，但是相关反馈只告诉了你“一些”相关或不相关的文档

- 将用户提示的相关文档集*DR'* 和不相关文档集作为对*DR* 和*DN* 的估计,重复地修改query达到优化的目的
- 从初始query开始

$$Q' = \alpha Q + \beta\left(\frac{1}{R'}\sum_{i \in D_{R'}} D_i\right) - \gamma\left(\frac{1}{N'}\sum_{i \in D_{N'}} D_i\right)$$

- *Q* 是初始的query， α, β和γ 是一个合适的常数
- Q, Q',Di均为加权向量

举例

- Q:初始query
- D1: 相关文档
- D2:不相关文档
- α = 1, β = 1/2, γ = 1/4
- 假设:

	T1	T2	T3	T4	T5
Q	5,	0,	3,	0,	1)
D1	2,	1,	2,	0,	0)
D2	1,	0,	0,	0,	2)

$$Q' = Q + \frac{1}{2}\left(\sum_{i \in D_{R'}} D_i\right) - \frac{1}{4}\left(\frac{1}{N'}\sum_{i \in D_{N'}} D_i\right)$$

$$S(Q, D_i) = \sum_{j=1}^{t} (Q_j \times D_{ij})$$

$$Q' = (5,0,3,0,1) + \frac{1}{2}(2,1,2,0,0) - \frac{1}{4}(1,0,0,0,2)$$

$$Q' = (5.75, 0.5, 4, 0, 0.5)$$

S(Q,D1) = (5•2)+(0 • 1)+(3 • 2)+(0 • 0)+(1 • 0) = 16
S(Q'D1)=(5.75 • 2)+(0.5 • 1)+(4 • 2)+(0 • 0)+(0.5 • 0)=20

S(Q,D2) = (5•1)+(0 • 0)+(3 • 0)+(0 • 0)+(1 • 2) = 7
S(Q'D2)=(5.75 • 1)+(0.5 • 0)+(4 • 0)+(0 • 0)+(0.5 • 2)=6.75

Rocchio方法

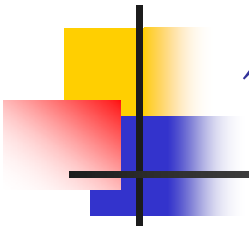
- 可以认为类中心向量法是它的特例
  - Rocchio公式

$$w'_{jc} = \alpha w_{jc} + \beta \frac{\sum_{i \in C} x_{ij}}{n_C} - \gamma \frac{\sum_{i \notin C} x_{ij}}{n - n_C}$$

- w'_{jc}：类C中心向量的权重
- n_C：训练样本中正例个数
- x_{ij}：文档向量的权重

  - 分类

$$CSV_c(d_i) = \mathrm{w_c} \cdot \mathrm{x}_i = \frac{\sum w_{cj} \cdot x_{ij}}{\sqrt{\sum w_{cj}^2} \sqrt{\sum x_{ij}^2}}$$



使用相关反馈 (Rocchio)

- 相关反馈方法可以用于文本分类
- 使用标准的TFIDF权重向量来表示文档 (用最大term频率做归一化)
- 对每个类别，通过累加该类中训练样本
- 计算对一个类别中的各训练样本的特征向量进行累加，构成一个*prototype*向量
- 用cosine 计算相似度，将测试文档分配到与prototype向量的相似度最接近的类别中去

Rocchio 文本分类算法(训练)

- 假设类别集合为 $\{c1, c2, \ldots cn\}$
- For i from 1 to n let $\mathbf{p}_i = <0, 0, \ldots, 0>$ (*init. prototype vectors*)
- For each training example $<x, c(x)> \in D$
- Let **d** be the frequency normalized TF/IDF term vector for doc x
- Let $i = j: (cj = c(x))$
- (*sum all the document vectors in ci to get* ***p****i*)
- Let $\mathbf{p}_i = \mathbf{p}_i + \mathbf{d}$

Rocchio 文本分类算法(测试)

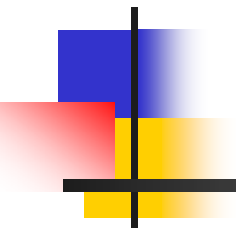
- Given test document *x*
- Let **d** be the TF/IDF weighted term vector for *x*
- Let *m* = –2 (*init. maximum cosSim*)
- For *i* from 1 to *n*:
- (*compute similarity to prototype vector*)
- Let *s* = cosSim(**d**, **p***i*)
- if *s* > *m*
- let *m* = *s*
- let *r* = *ci* (*update most similar class prototype*)
- Return class *r*

Rocchio 的特点

- 在每一个类中，对各个实例形成一个简单的“泛化generalization”，即*prototype*
- Prototype向量不需要用长度进行归一化，因为 cosine相似度计算对向量的长度不敏感

Rocchio 事件复杂度

- Note: 将两个稀疏的向量相加的时间是和两个向量中的非零项的个数成正比的
- 训练时间: $O(|D|(L_d + |V_d|)) = O(|D| L_d)$
  - D: 文档集合
  - L_d : D中文档的平均长度
  - V_d : D中文档的平均词表大小
- 测试时间: $O(L_t + |C||V_t|)$
  - L_t : 测试文档的平均长度
  - $|V_t|$: 测试文档的平均词表长度
  - 假设$\mathbf{p}_i$ 向量的长度在训练的过程中被计算和存储, cosSim(**d**, $\mathbf{p}_i$) 的计算在时间上和d中的非零项的数量成比例



贝叶斯分类

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

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

- MAP (最大后验概率) 假定

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

$$P(D)=\sum_{i=1}^{n}P(h_i)P(D|h_i)$$

- 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

参数计算

$$P(c_j)=\frac{c_j\text{的文档个数}}{\text{总文档个数}}=\frac{N(c_j)}{\sum_k N(c_k)}\approx\frac{1+N(c_j)}{|c|+\sum_k N(c_k)}$$

$$P(w_i\mid c_j)=\frac{w_i\text{在}c_j\text{类别文档中出现的次数}}{\text{在}c_j\text{类所有文档中出现的词的次数}}\approx\frac{1+N_{ij}}{\text{不同词个数}+\sum_k N_{kj}}$$

防止下溢

- 大量概率值（0和1之间）相乘，结果太小，将导致浮点下溢
- 由于$\log(xy) = \log(x) + \log(y)$，因此用概率的对数累加的方式，比用概率累乘的方式更好
- 对数是正比例函数，因此概率对数最大的类，概率一定也是最大的

概率估计

- 概率通常是基于训练数据中观察到的频率来估计的
- 如果 D 中有 n_i 个文本属于 c_i 类，并且这 n_i 个文本有 n_{ij} 个文本包含了特征 e_j，那么：

$$P(e_j \mid c_i) = \frac{n_{ij}}{n_i}$$

- 对于小的训练集合来说，这样估计概率容易出错
- 如果一个特征在训练集中从未出现过 e_k，则：
  $\forall c_i : P(e_k \mid c_i) = 0.$
- 如果 e_k 又恰好在测试集中出现，则：
  $\forall c_i : P(E \mid c_i) = 0$ 且 $\forall c_i : P(c_i \mid E) = 0$

平滑Smoothing

- 为了从小样本中估计概率，需要对概率值进行调节和平滑
- Laplace 平滑算法假设已经观察到m个“虚拟”文本，并且特征e_j的先验概率为p

$$P(e_j \mid c_i) = \frac{n_{ij} + mp}{n_i + m}$$

- 对于二值化特征，p 简单地设为0.5.

文本分类 Naïve Bayes算法(训练)

- Let V be the vocabulary of all words in the documents in D
- For each category $ci \in C$
  - Let Di be the subset of documents in D in category ci
  - $P(ci) = |Di| / |D|$
  - Let Ti be the concatenation of all the documents in Di
    - Let ni be the total number of word occurrences in Ti
    - For each word $wj \in V$
      - Let nij be the number of occurrences of wj in Ti
        - Let $P(wi \mid ci) = (nij + 1) / (ni + |V|)$

文本分类 Naïve Bayes算法(测试)

- Given a test document *X*
- Let *n* be the number of word occurrences in *X*
- Return the category:

$$\underset{c_i \in C}{\operatorname{argmax}} P(c_i) \prod_{i=1}^{n} P(a_i \mid c_i)$$

- where *ai* is the word occurring the i-th position in *X*

Naïve Bayes举例

过敏

打喷嚏

- C = {allergy, cold, well}
- e_1 = sneeze; e_2 = cough; e_3 = fever
- 当前实例是：E = {sneeze, cough, ¬fever}

Prob	**Well**	**Cold**	**Allergy**
P(c_i)	0.9	0.05	0.05
P(sneeze|c_i)	0.1	0.9	0.9
P(cough|c_i)	0.1	0.8	0.7
P(fever|c_i)	0.01	0.7	0.4

Naïve Bayes 举例 (cont.)

- 参数计算：
  - P(well | E) = (0.9)(0.1)(0.1)(0.99)/P(E)=0.0089/P(E)
  - P(cold | E) = (0.05)(0.9)(0.8)(0.3)/P(E)=0.01/P(E)
  - P(allergy | E) = (0.05)(0.9)(0.7)(0.6)/P(E)=0.019/P(E)

- 最大概率类: allergy
  - P(E) = 0.089 + 0.01 + 0.019 = 0.0379
  - P(well | E) = 0.23
  - P(cold | E) = 0.26
  - P(allergy | E) = 0.50

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

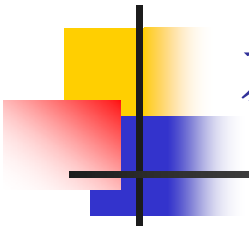
P(n) = 5/14

湿度

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

Play-tennis例子: 分类 X

- 例子 X = <rain, hot, high, false>
  - P(X|p)·P(p) = P(rain|p)·P(hot|p)·P(high|p)·P(false|p)·P(p) = 3/9·2/9·3/9·6/9·9/14 = 0.010582
  - P(X|n)·P(n) = P(rain|n)·P(hot|n)·P(high|n)·P(false|n)·P(n) = 2/5·2/5·4/5·2/5·5/14 = 0.018286
- 样本 X 被分到 n类，即"不适合打网球"



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%

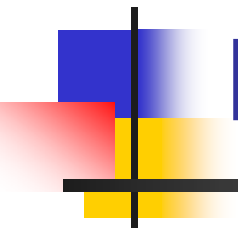
Naïve Bayes时间复杂度

- 训练时间: $O(|D|L_d + |C||V|))$
  - L_d is 是文档集合D中文档的平均长度
  - 一般来说 $|C||V| < |D|L_d$

- 测试时间: $O(/C/ L_t)$
  - L_t 是测试文档的平均长度
  - 和 Rocchio 的时间复杂度相似

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



K近邻

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

$$p(c_1 \mid y) = \frac{n1}{n1+n2} \qquad p(c_2 \mid y) = \frac{n2}{n1+n2}$$

- 如果p(c1|y)>p(c2|y),判为c1,否则判为c2

K 近邻算法

- **Training:**
- For each each training example $\langle x, c(x) \rangle \in D$
- Compute the corresponding TF-IDF vector, $\mathbf{d}_x$, for document x

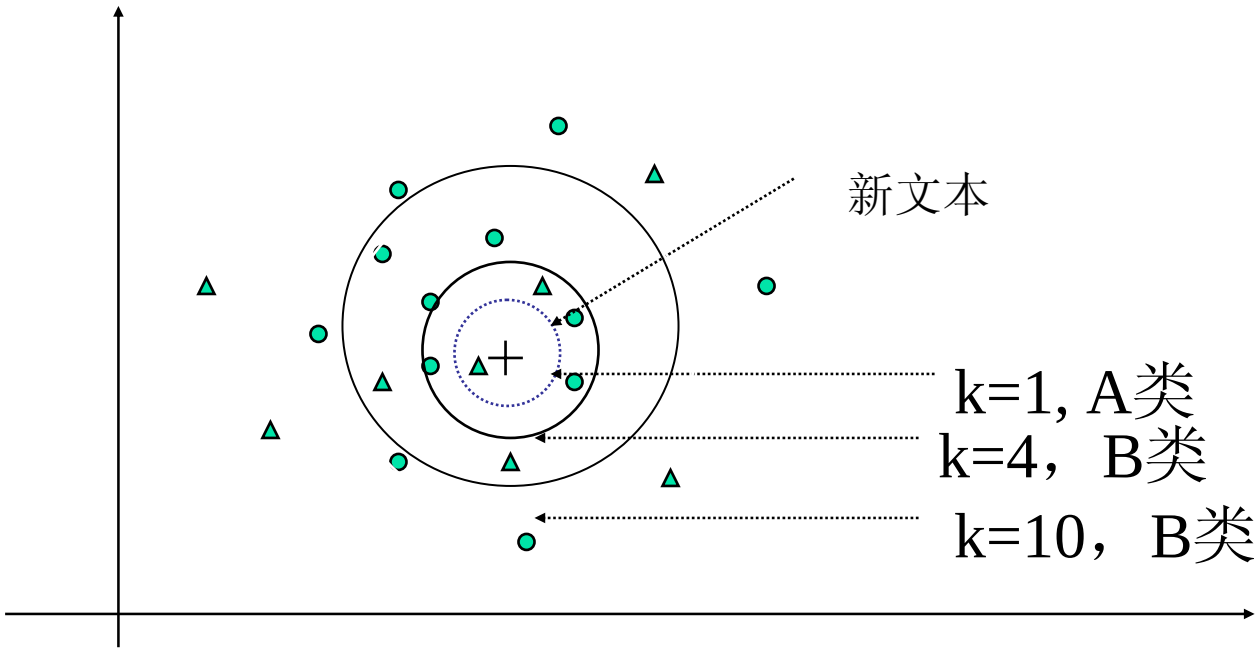
- **Test instance y:**
- Compute TF-IDF vector $\mathbf{d}$ for document y
- For each $\langle x, c(x) \rangle \in D$
- Let $s_x = \text{cosSim}(\mathbf{d}, \mathbf{d}_x)$
- Sort examples, x, in D by decreasing value of s_x
- Let N be the first k examples in D. (*get most similar neighbors*)
- Return the majority class of examples in N

相似度度量

- 最简单的是欧式距离
- 还有汉明距离（特征值不同的特征数）
- 最常用的还是用TF-IDF 计算权重，用Cosine计算相似度的方法

kNN方法

- 一种基于实例的学习方法



带权重计算，计算权重和最大的类。k常取3或者5。

KNN在文本分类中的应用

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

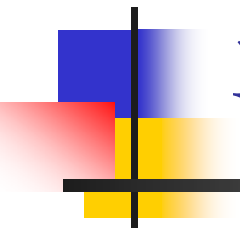
Table 16.12 Classification results for an 1NN categorizer for the "earnings" category. Classification accuracy is 95.3%.

Nearest Neighbor时间复杂度

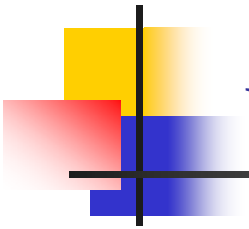
- 训练时间: $O(|D| L_d)$
  - 用来构成TF-IDF 向量
- 测试时间: $O(L_t + |D||V_t|)$ 将测试文本和全部训练文本进行比较
  - 假设用向量空间模型表示文档
  - 用cosine计算相似度
- 对于训练集非常大的情况，测试时间可能非常长

用倒排表实现最近邻

- 确定 k 个最近邻的问题类似于：
  - 以测试文本为查询
  - 从训练文本集合中找出 k 个最相似的文本的问题
- 因此可以采用基于向量空间模型的倒排文档实现
- 测试时间：$O(B|V_t|)$
  - B 是包含测试文档中的词汇的训练文档集合中的文档数量
- 因此，全部分类时间为：$O(L_t + B|V_t|)$
  - 一般来说 $B << |D|$

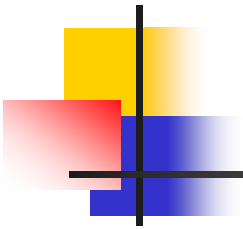


决策树



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



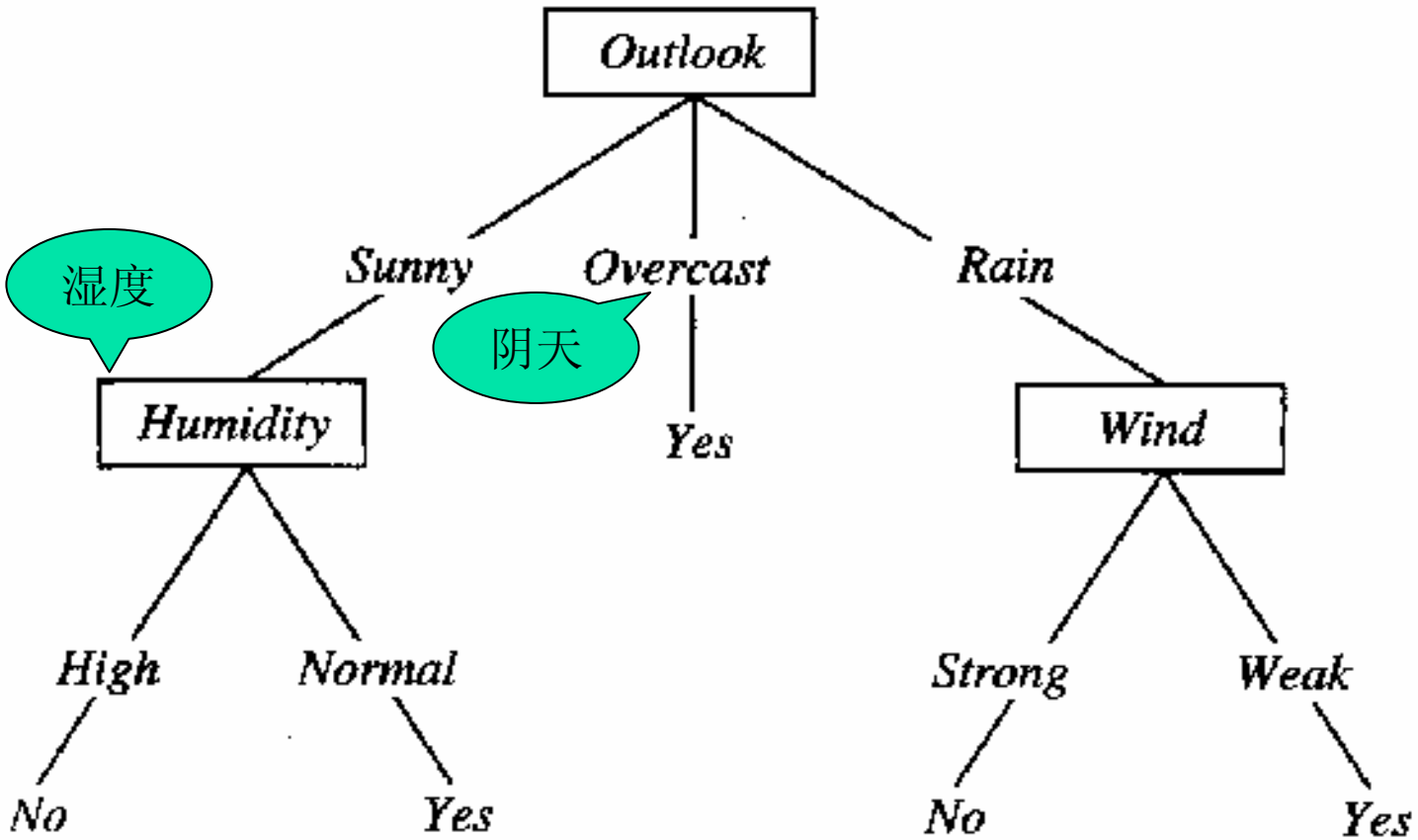
简介

- 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


Outlook
Sunny
Overcast
Rain
湿度
阴天
Humidity
Yes
Wind
High
Normal
Strong
Weak
No
Yes
No
Yes

表达式

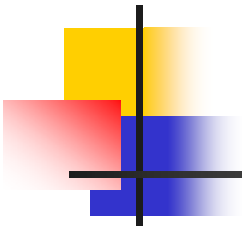
$$(Outlook = Sunny \wedge Humidity = Normal)$$

$$\vee \quad (Outlook = Overcast)$$

$$\vee \quad (Outlook = Rain \wedge Wind = Weak)$$

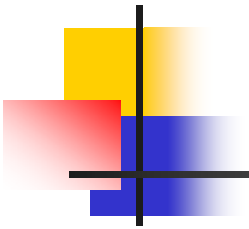
决策树学习的适用问题

- 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



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

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

$$Entropy([9+, 5-]) = -(9/14) \log_2(9/14) - (5/14) \log_2(5/14)$$
$$= 0.940$$

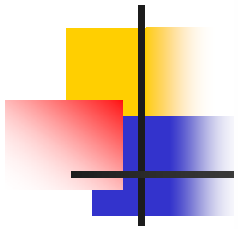
用信息增益度量期望熵最低

- 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

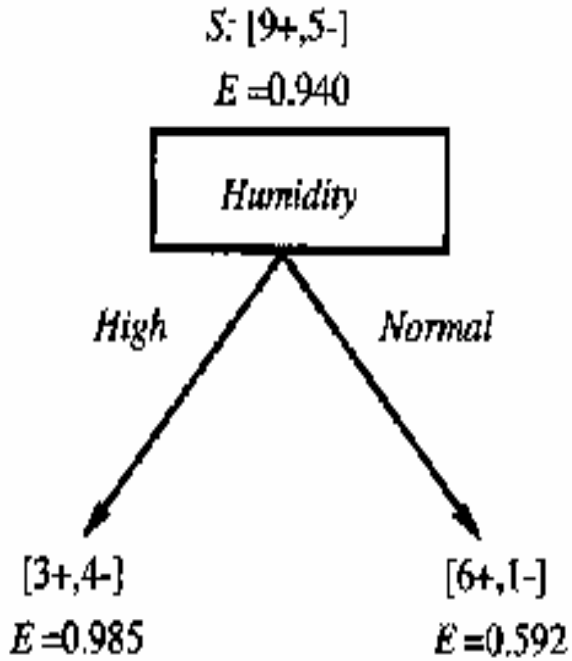
$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

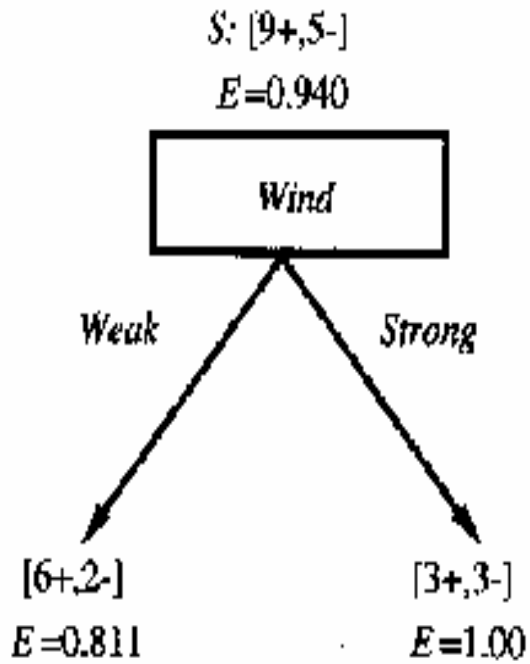
举例

$$
\begin{aligned}
Values(Wind) &= Weak, Strong \\
S &= [9+, 5-] \\
S_{Weak} &\leftarrow [6+, 2-] \\
S_{Strong} &\leftarrow [3+, 3-] \\
Gain(S, Wind) &= Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v) \\
&= Entropy(S) - (8/14) Entropy(S_{Weak}) \\
&\quad - (6/14) Entropy(S_{Strong}) \\
&= 0.940 - (8/14)0.811 - (6/14)1.00 \\
&= 0.048
\end{aligned}
$$



Which attribute is the best classifier?



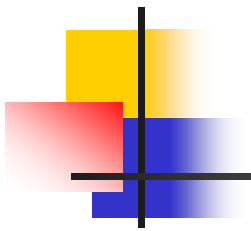


Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048

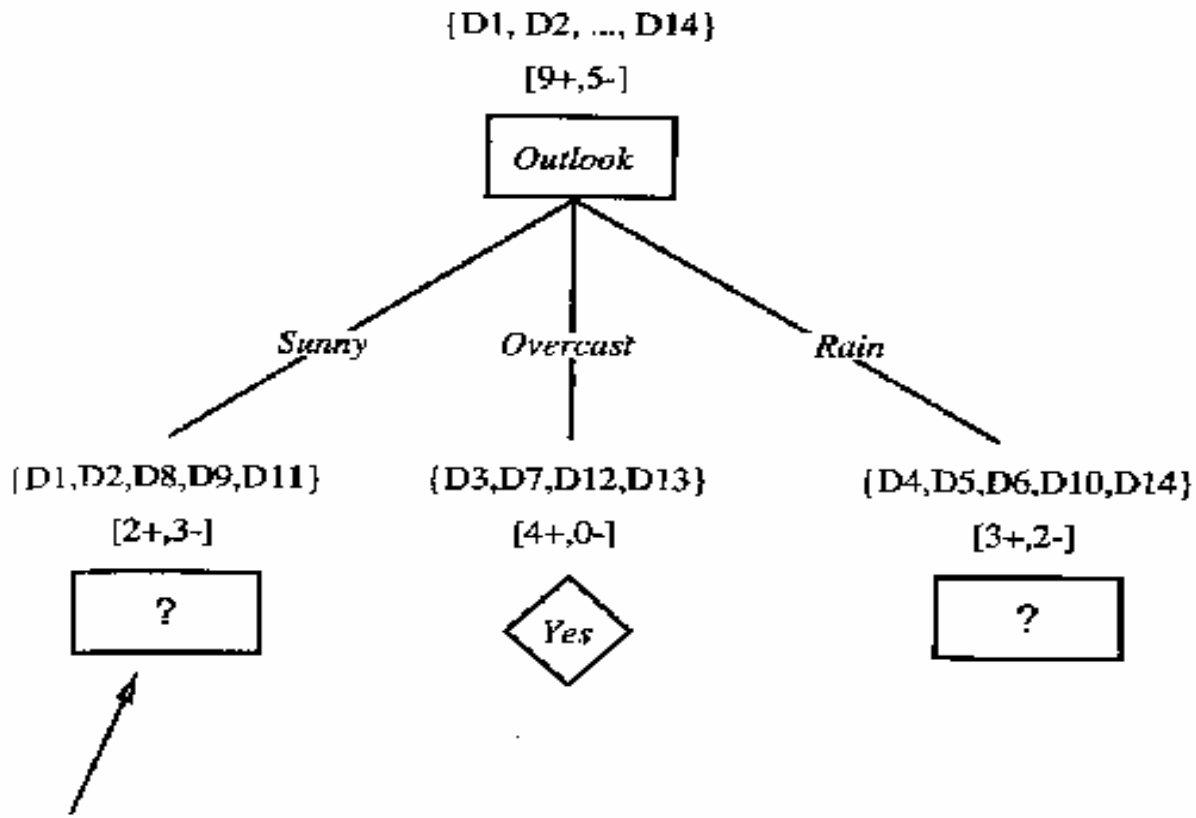


$$Gain(S, Outlook) = 0.246$$

$$Gain(S, Humidity) = 0.151$$

$$Gain(S, Wind) = 0.048$$

$$Gain(S, Temperature) = 0.029$$



Which attribute should be tested here?

S_{sunny} = {D1,D2,D8,D9,D11}

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

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

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

ID3算法

创建树的Root结点

开始

A←Attributes中分类能力最好的属性

Root的决策属性←A

对于每个可能值v_i

在Root下加一个新的分支对应测试$A=v_i$

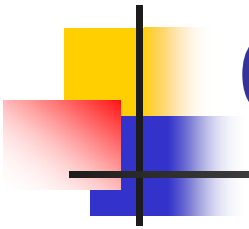
令$Examples_{vi}$为Examples中满足A属性值为v_i的子集

如果$Examples_{vi}$为空

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

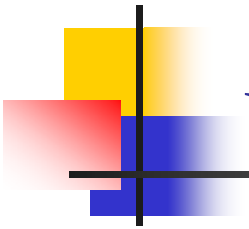
否则在这个新分支下加一个子树ID3($example_{vi}$,target-attribute,attributes-{A})

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)

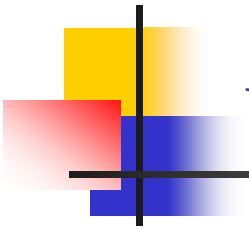
- 避免过度拟合数据
  - 基本的决策树构造算法没有考虑噪声，生成的决策树完全与训练例子拟合
  - 有噪声情况下，完全拟合将导致过分拟合（Overfitting），即对训练数据的完全拟合反而不具有很好的预测性能。

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。

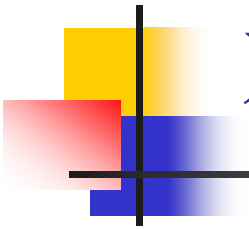
$$SplitInformation(S, A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

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



决策树的优点

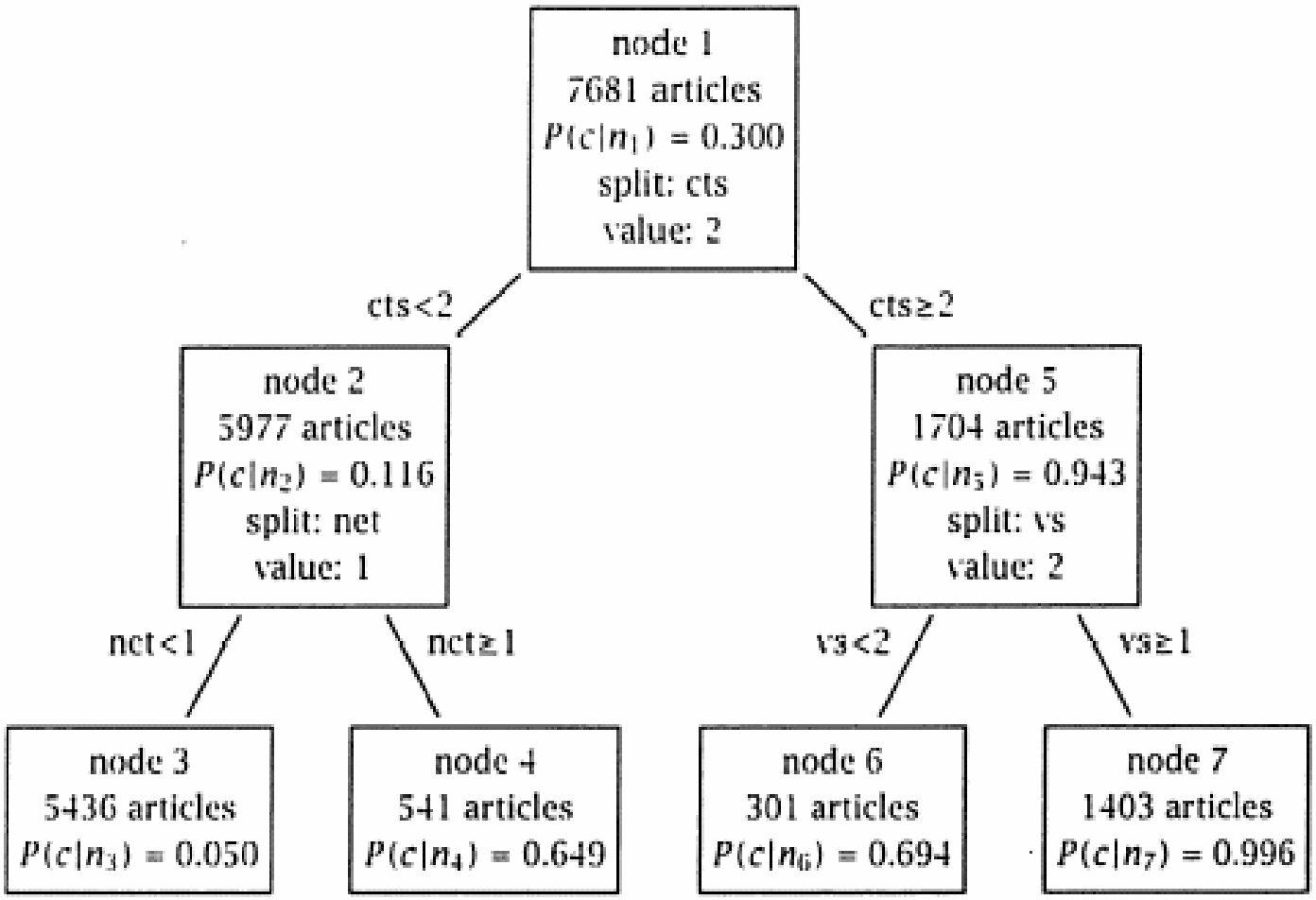
- 可以生成可以理解的规则
- 计算量相对来说不是很大
- 可以处理连续和离散字段
- 决策树可以清晰的显示哪些字段比较重要
- 过程可见



不足之处

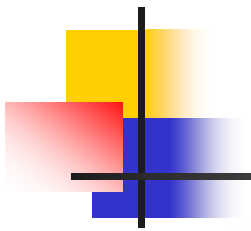
- 对连续性的字段比较难预测
- 当类别太多时，错误可能会增加的比较快
- 一般的算法分类的时候，只是根据一个属性来分类。
- 不是全局最优

举例：利用决策树进行文本分类



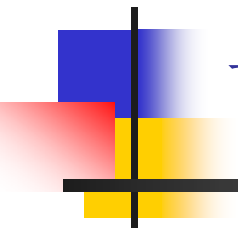
entropy at node 1, $P(C\|N) = 0.300$	0.611
entropy at node 2, $P(C\|N) = 0.116$	0.359
entropy at node 5, $P(C\|N) = 0.943$	0.219
weighted sum of 2 and 5	$\frac{5977}{7681} \times 0.359 + \frac{1704}{7681} \times 0.219 = 0.328$
information gain	$0.611 - 0.328 = 0.283$

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.

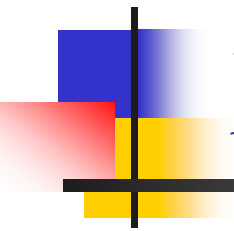


"earnings" assigned?	"earnings" correct? YES	NO
YES	1024	69
NO	63	2143

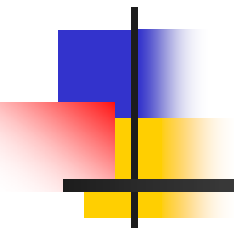
Table 16.5 Contingency table for a decision tree for the Reuters category "earnings." Classification accuracy on the test set is 96.0%.



谢谢！

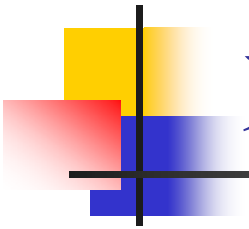


文本分类系统演示



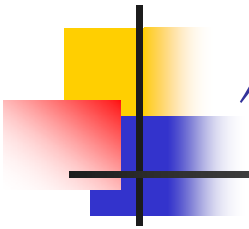
文本聚类
(Text Clustering)

刘挺
哈工大信息检索研究室
2004年秋



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
  - 自顶向下聚类
- 非层次聚类
  - K-均值

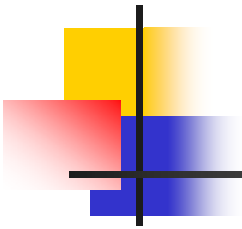


什么是聚类分析?

- 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）

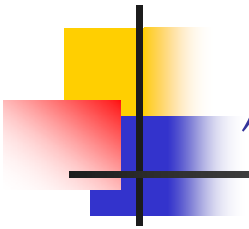


聚类算法类型

- 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

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

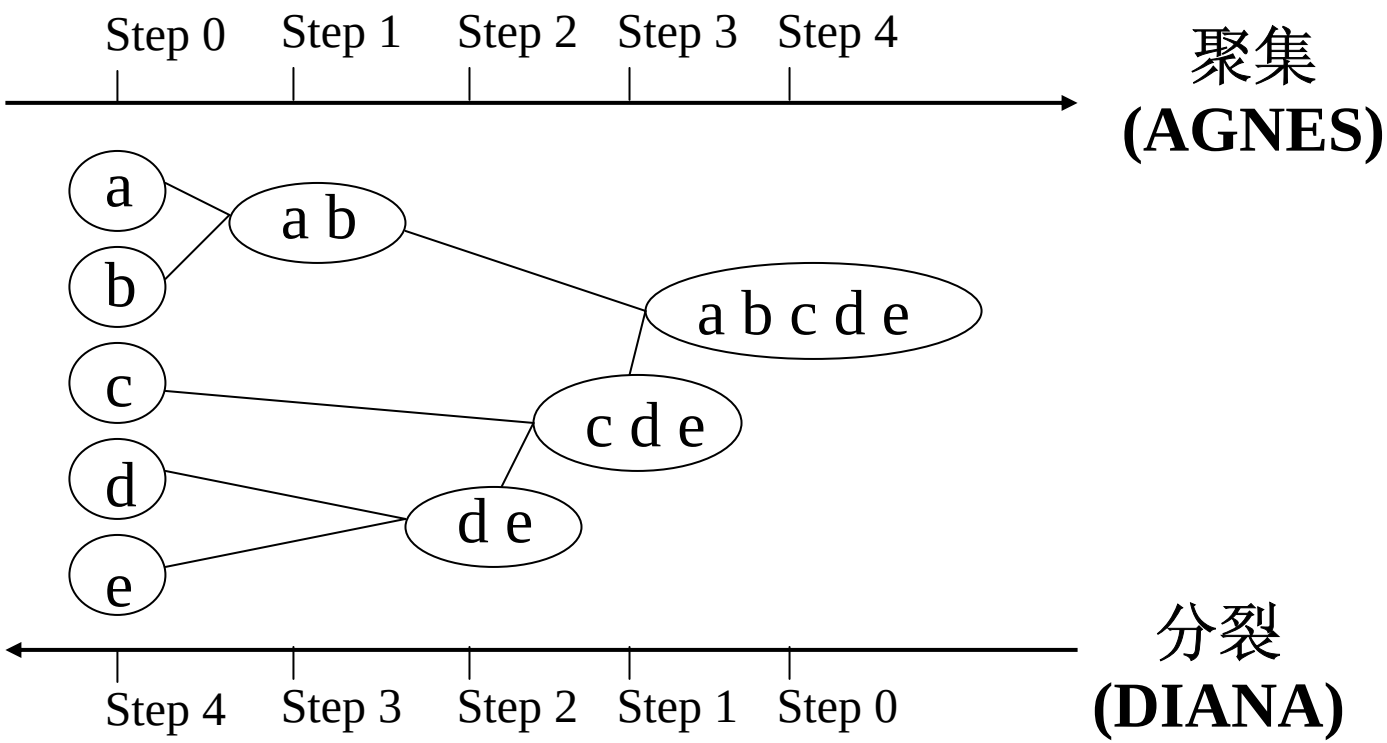


层次聚类

- 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

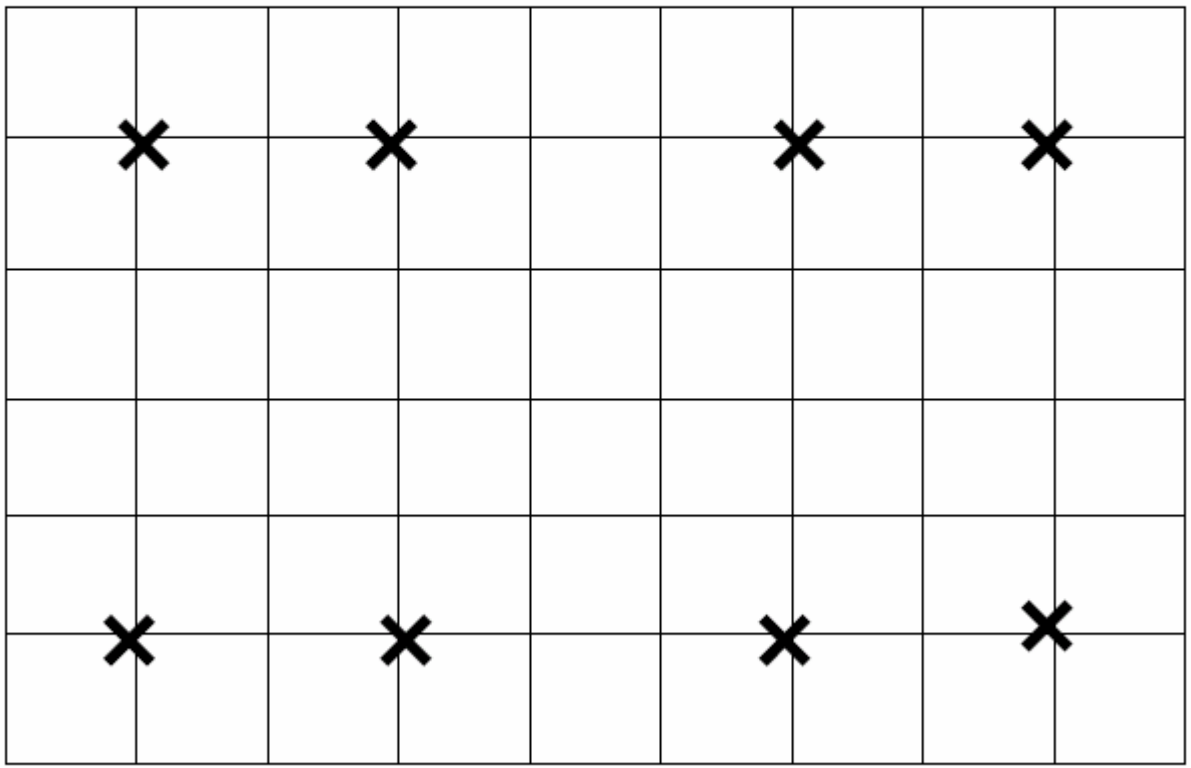
- 这种方法不需要输入参数K，但需要一个终止条件。例如：相似度阈值



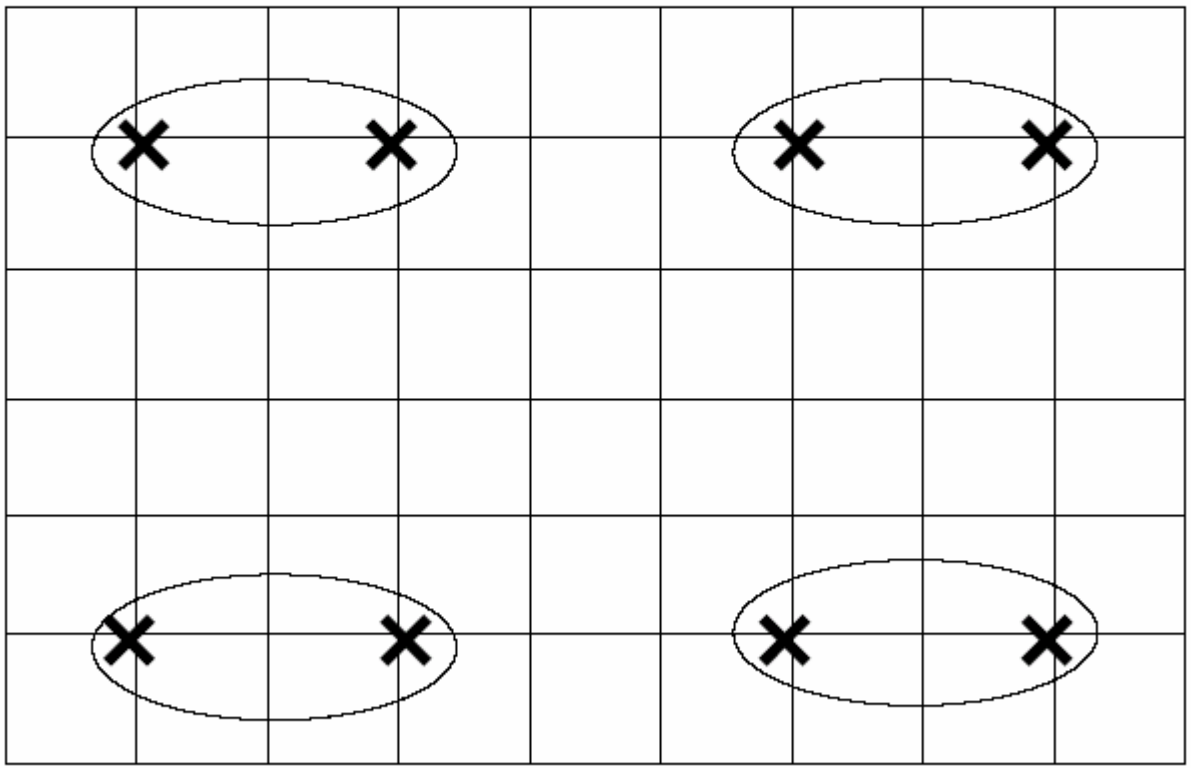
类的相似度度量

- 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
- 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

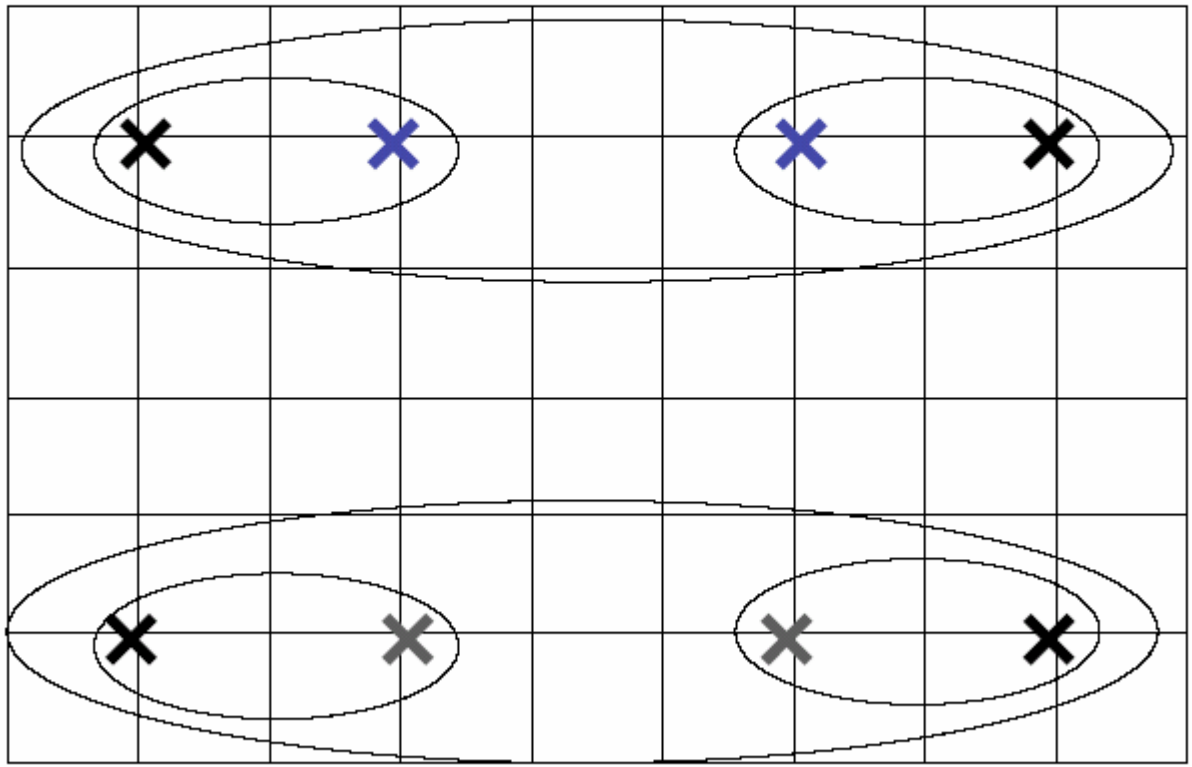
Single vs. Complete link



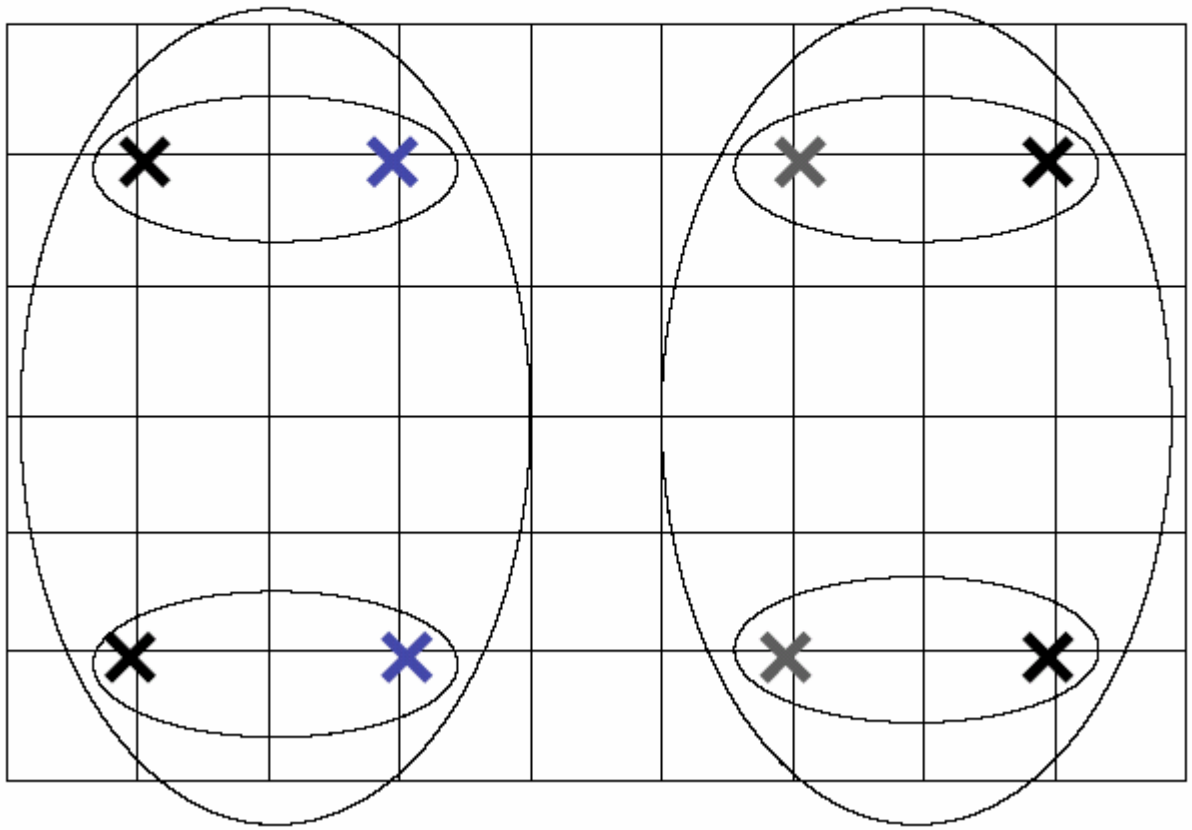
Single vs. Complete link

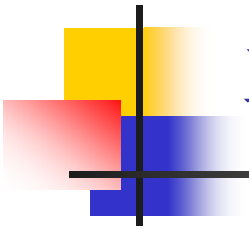


Single link



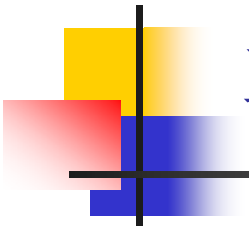
Complete link





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

K-均值(k-means)

- 将n个向量分到k个类别中去

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

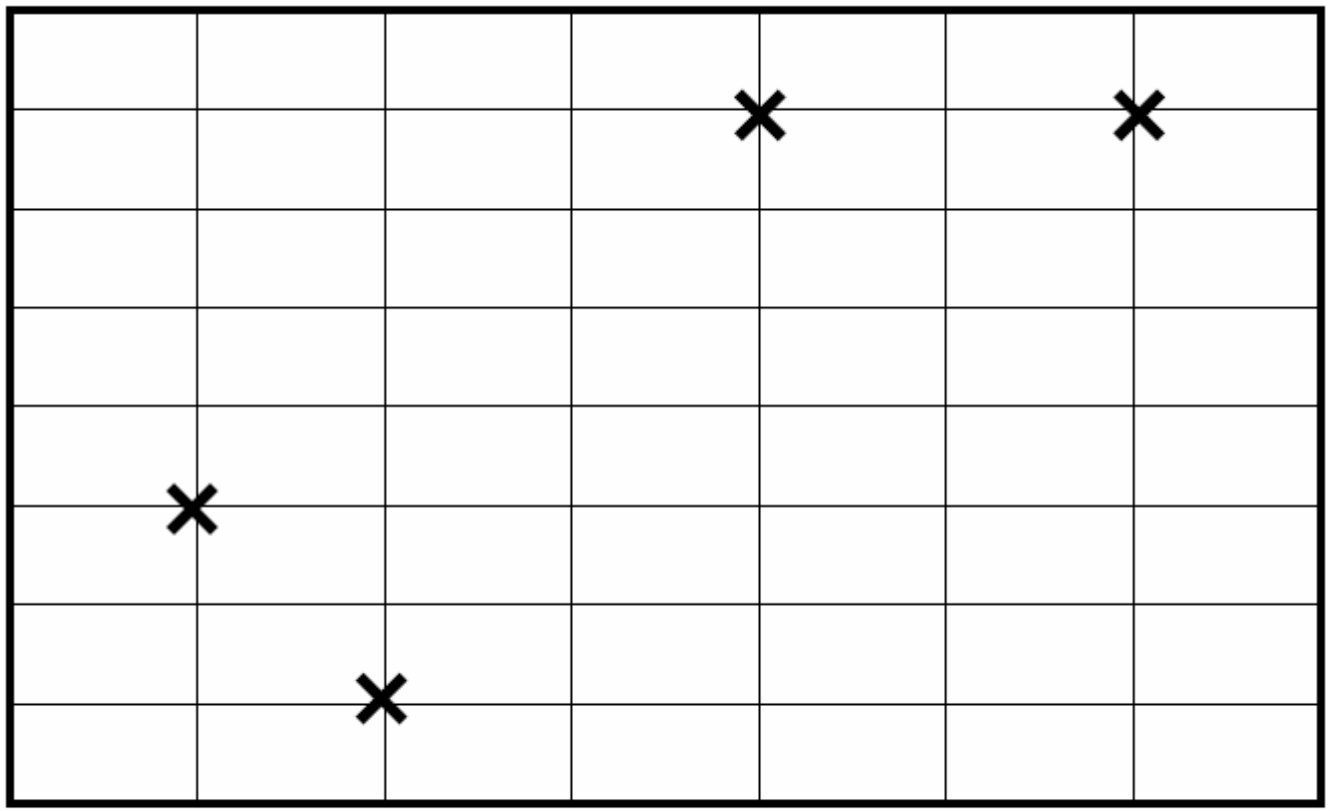
- 选择k个初始中心

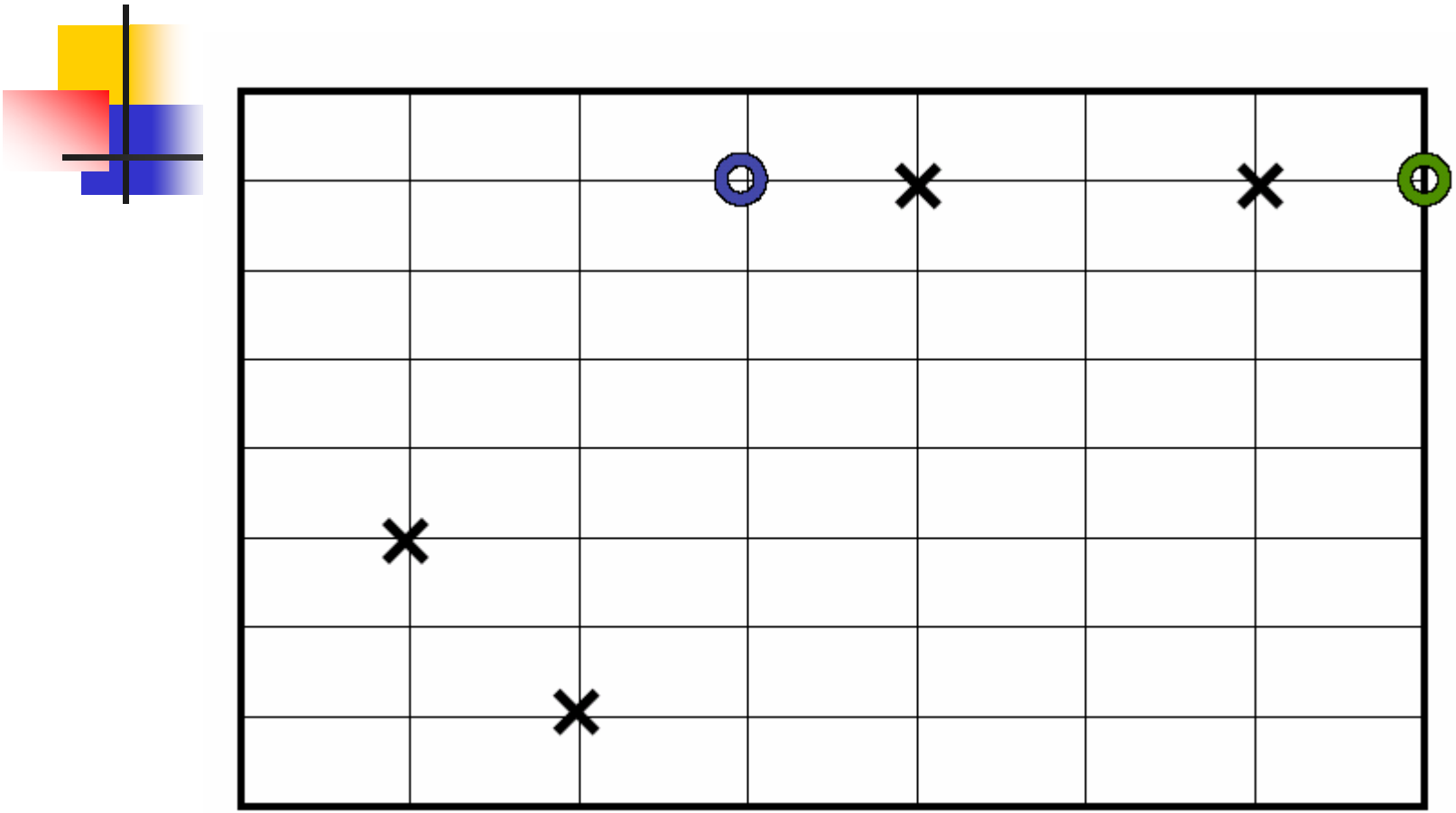
  $f_1=(4,3)\ \ f_2=(5,5)$

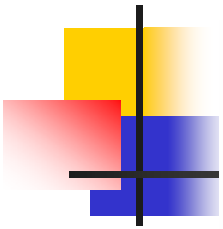
- 计算两项距离 $d(x_i,x_j)=\sqrt{\sum_{k=1}^{n}(x_{i_k}-x_{j_k})^2}$

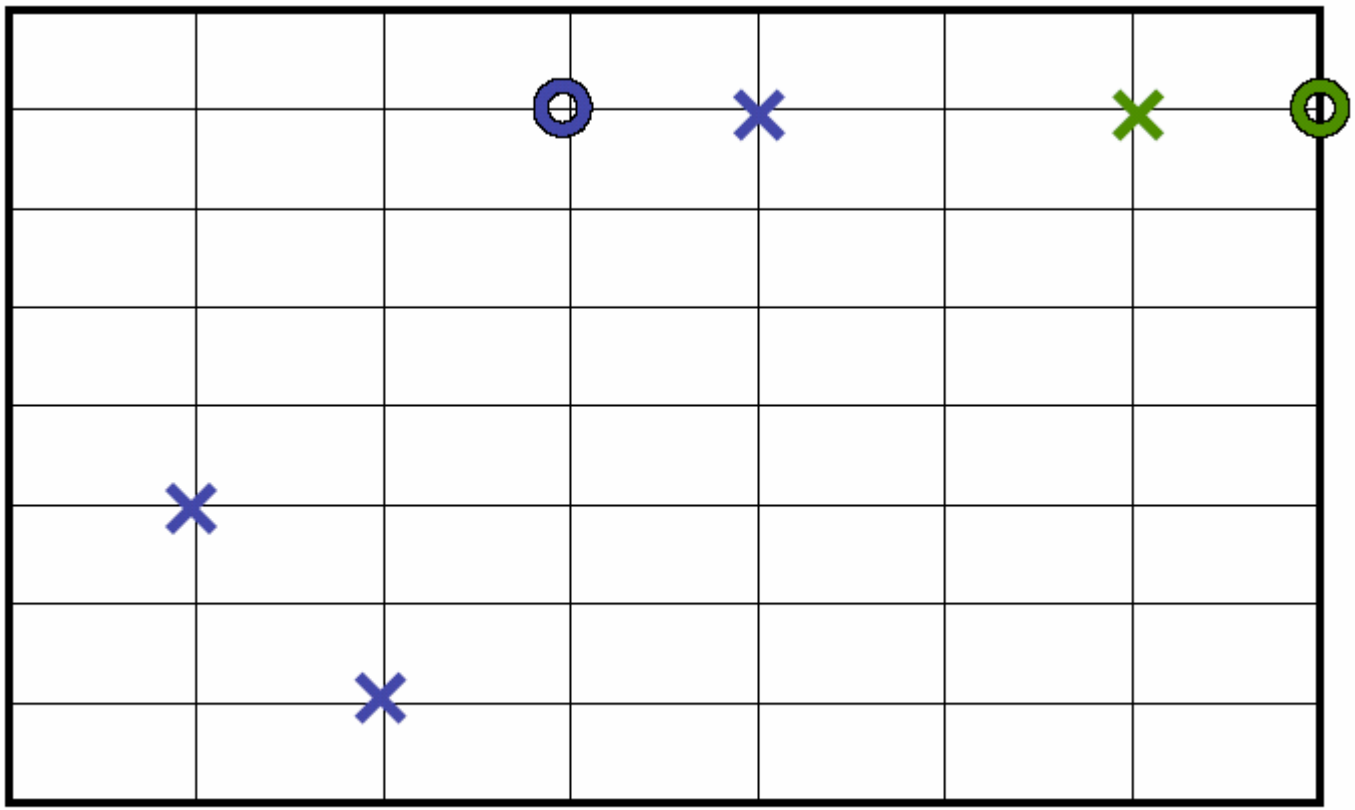
- 计算n个向量均值

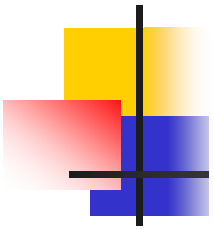
$$\mu(x_1,\ldots,x_n)=\left(\frac{\sum_{i=1}^{n}x_{i_1}}{n},\ldots,\frac{\sum_{i=1}^{n}x_{i_m}}{n}\right)$$

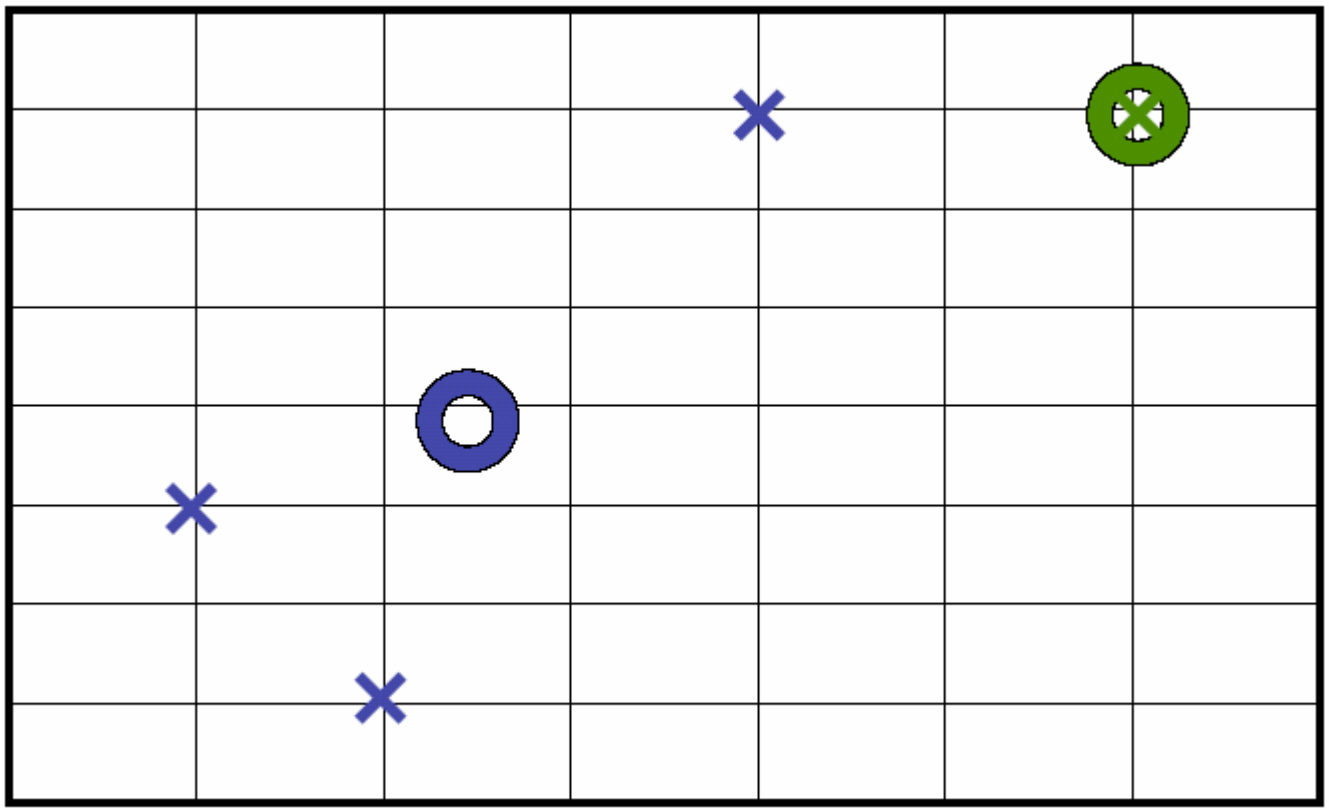


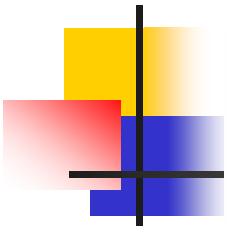


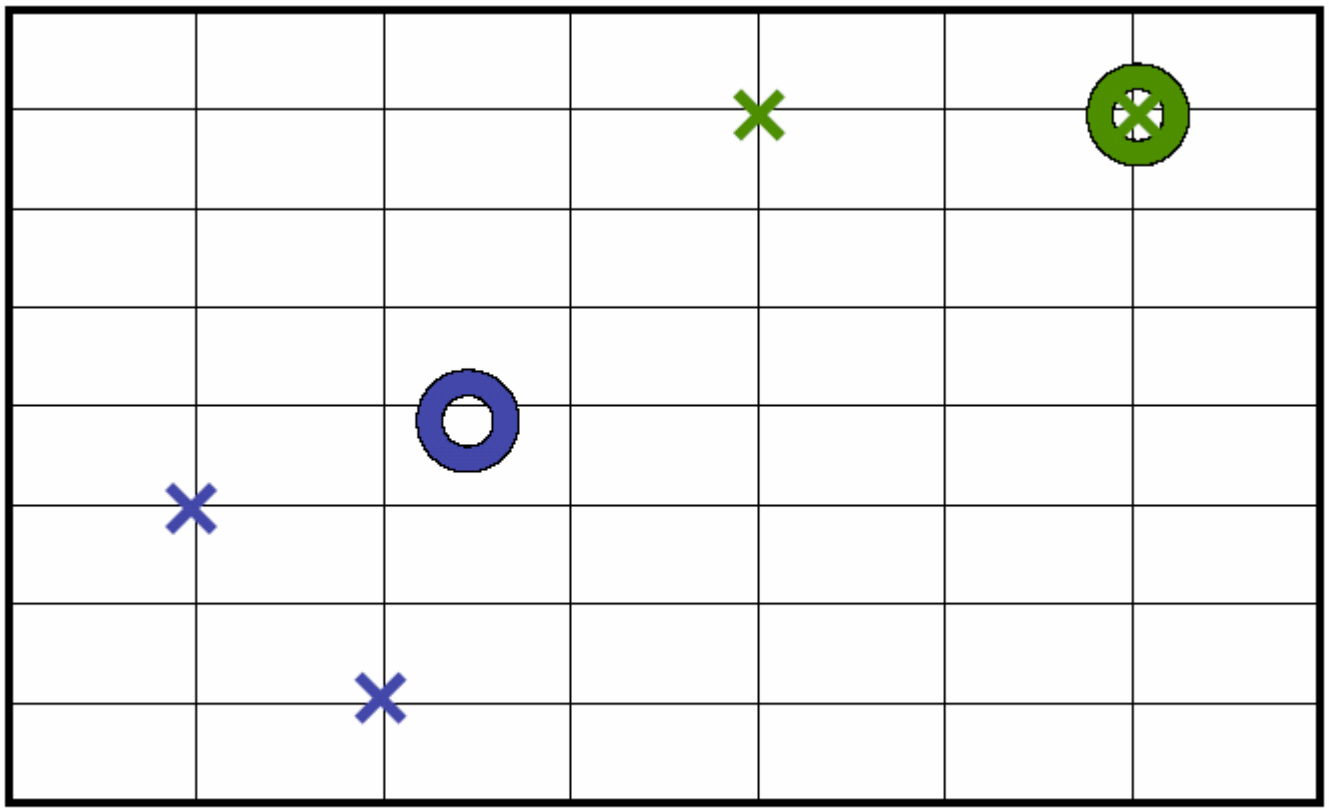


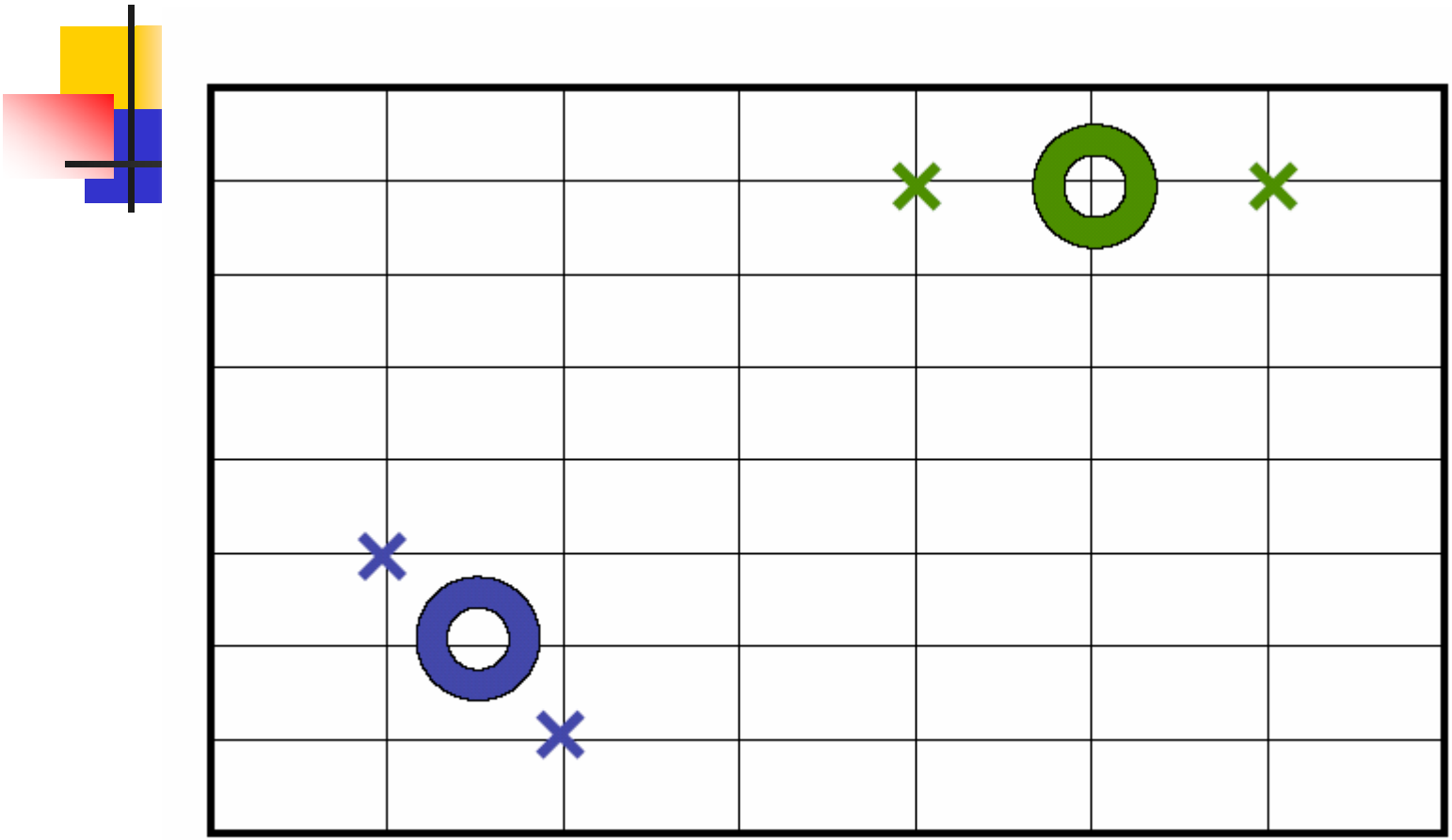






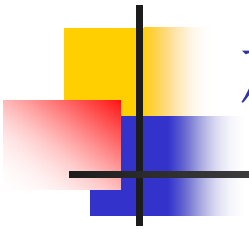






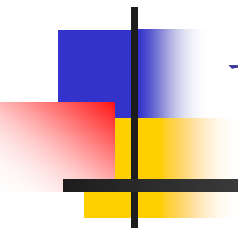
K-均值算法

- 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.

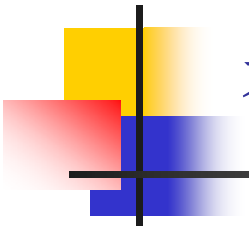


模糊聚类

- 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



谢谢



参考书目

- 现代信息检索
  - Modern Information Retrieval
  - 电子工业出版社
- 金北方有售

测验

- 基础题
  - 简述向量空间模型的概念
  - 简述倒排表的概念和使用方法
  - 简述文本分类的步骤
- 扩展题
  - 你对信息检索的那个方向最感兴趣，说明为什么
  - 你估计信息检索技术能否以及怎样和你未来的硕士论文方向结合
  - 对信息检索课的建议