# 现代信息检索

# Modern Information Retrieval

## 第12讲 支持向量机和排序学习

## SVM and Learning to Rank

*改编自"An introduction to Information retrieval"网上公开的课件，地址 http://nlp.stanford.edu/IR-book/

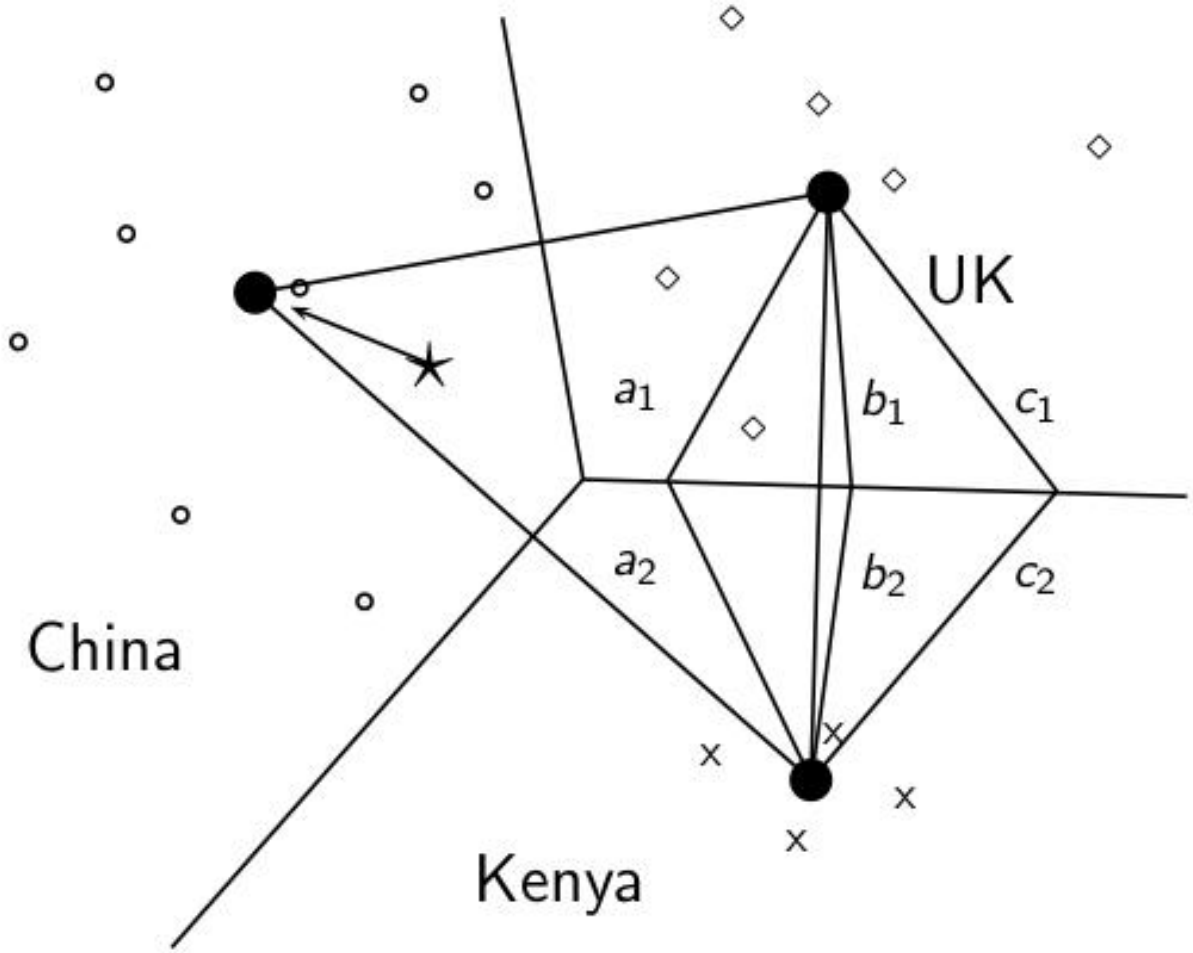
# 提纲

- 上一讲回顾
- 支持向量机
- 排序学习

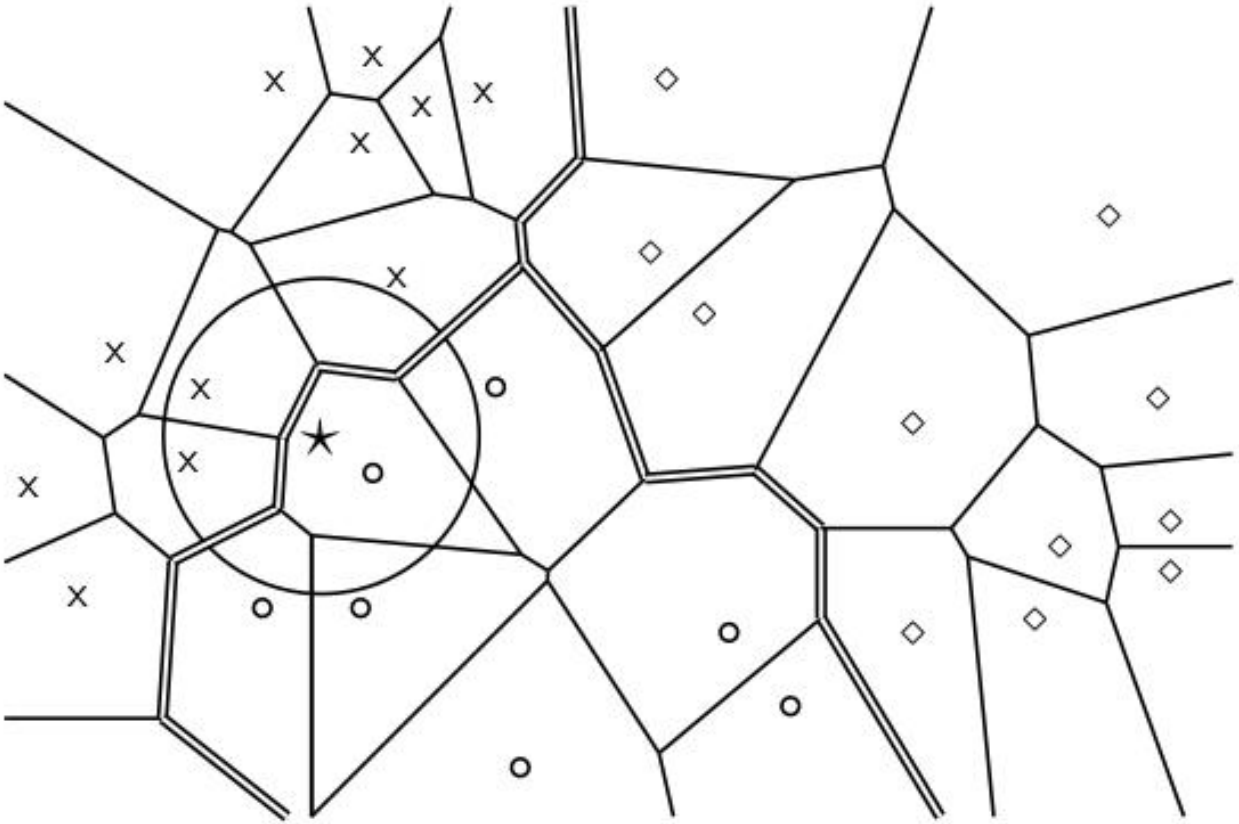
# 向量空间分类

- 同前面一样，训练集包含一系列文档，每篇都标记着它的类别
- 在向量空间分类中，该集合对应着空间中一系列标记的点或向量。
- 假设 1: 同一类中的文档会构成一片连续区域（contiguous region）
- 假设2: 来自不同类别的文档没有交集
- 接下来我们定义直线、平面、超平面来将上述不同区域分开

# Rocchio算法示意图 : a1 = a2, b1 = b2, c1 = c2



# kNN算法



对于★ 对应的文档，在1NN和 3NN下，分别应该属于哪个类？

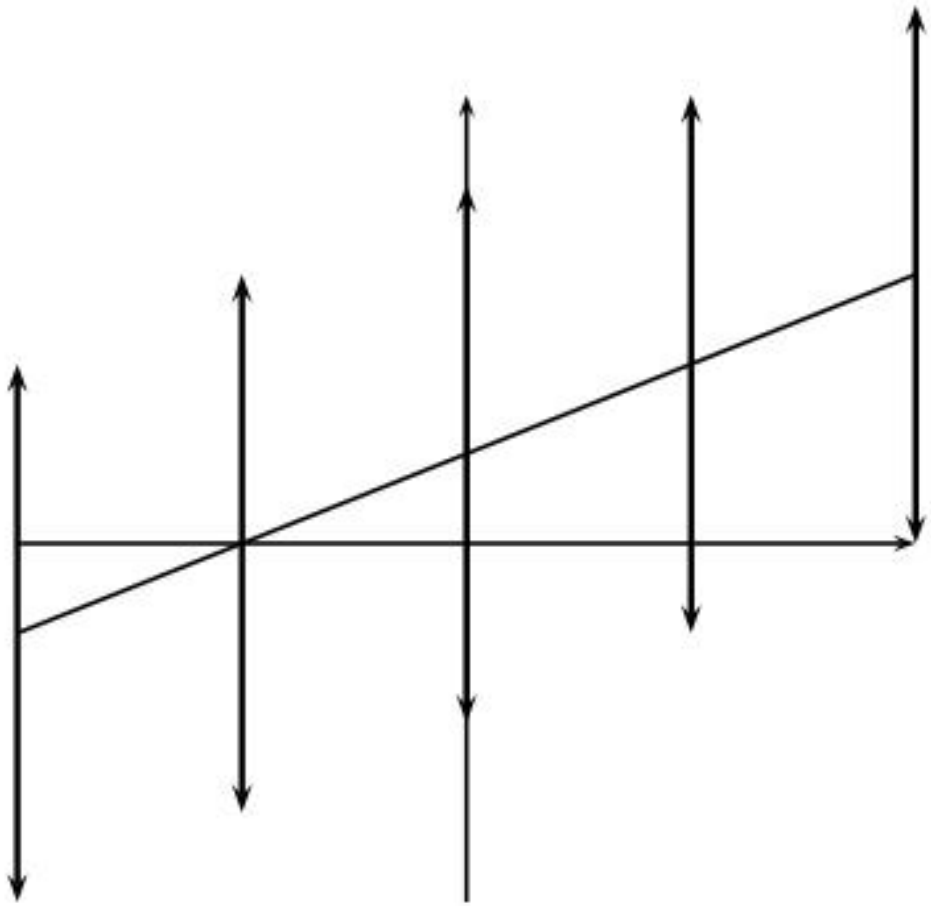
# 线性分类器

- 定义：
  - 线性分类器计算特征值的一个线性加权和 $\sum_i w_i x_i$
  - 决策规则：$\sum_i w_i x_i > \theta$?
  - 其中，$\theta$ 是一个参数
- 首先，我们仅考虑二元分类器
- 从几何上说，二元分类器相当于二维平面上的一条直线、三维空间中的一个平面或者更高维下的超平面，称为分类面
- 基于训练集来寻找该分类面
- 寻找分类面的方法：感知机(Perceptron)、 Rocchio, Naïve Bayes – 我们将解释为什么后两种方法也是二元分类器
- 假设：分类是线性可分的

# 一维下的线性分类器

- 一维下的分类器是方程 $w_1 d_1 = \theta$对应的点
- 点的位置是$\theta / w_1$
- 那些满足$w_1 d_1 \geq \theta$ 的点 $d_1$ 属于类别$c$
- 而那些$w_1 d_1 < \theta$的点$d_1$ 属于类别 $\overline{c}$.

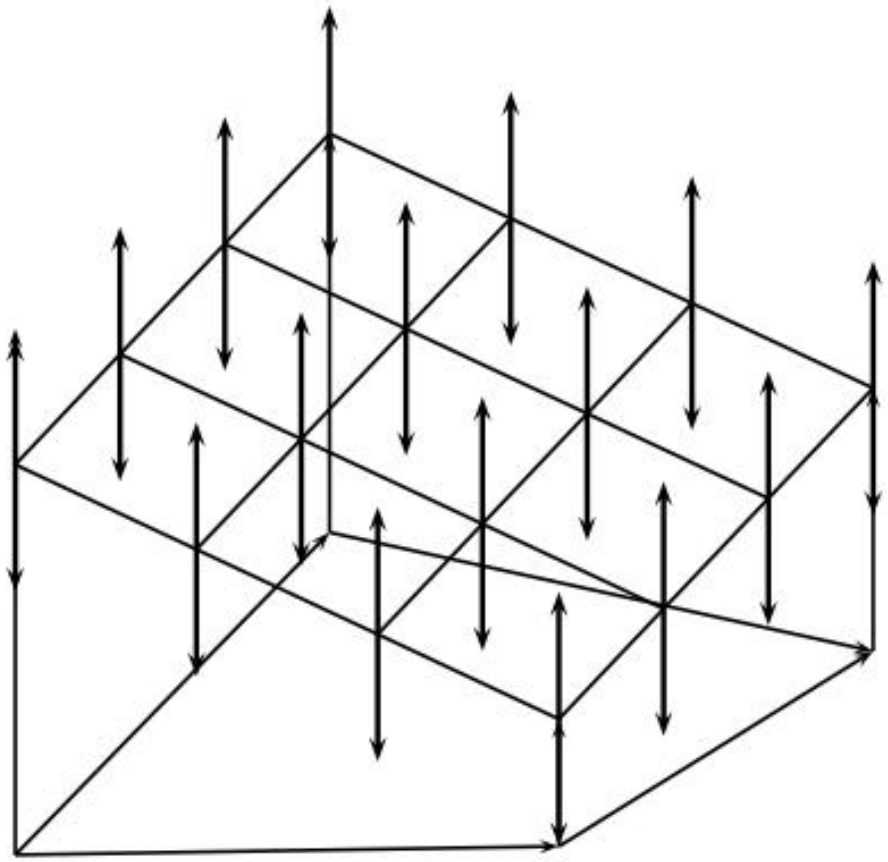
# 二维平面下的线性分类器



- 二维下的分类器是方程$w_1d_1 + w_2d_2 = \theta$对应的直线
- 那些满足$w_1d_1 + w_2d_2 \geq \theta$的点$(d_1\ d_2)$属于类别$c$
- 那些满足$w_1d_1 + w_2d_2 < \theta$的点$(d_1\ d_2)$属于类别

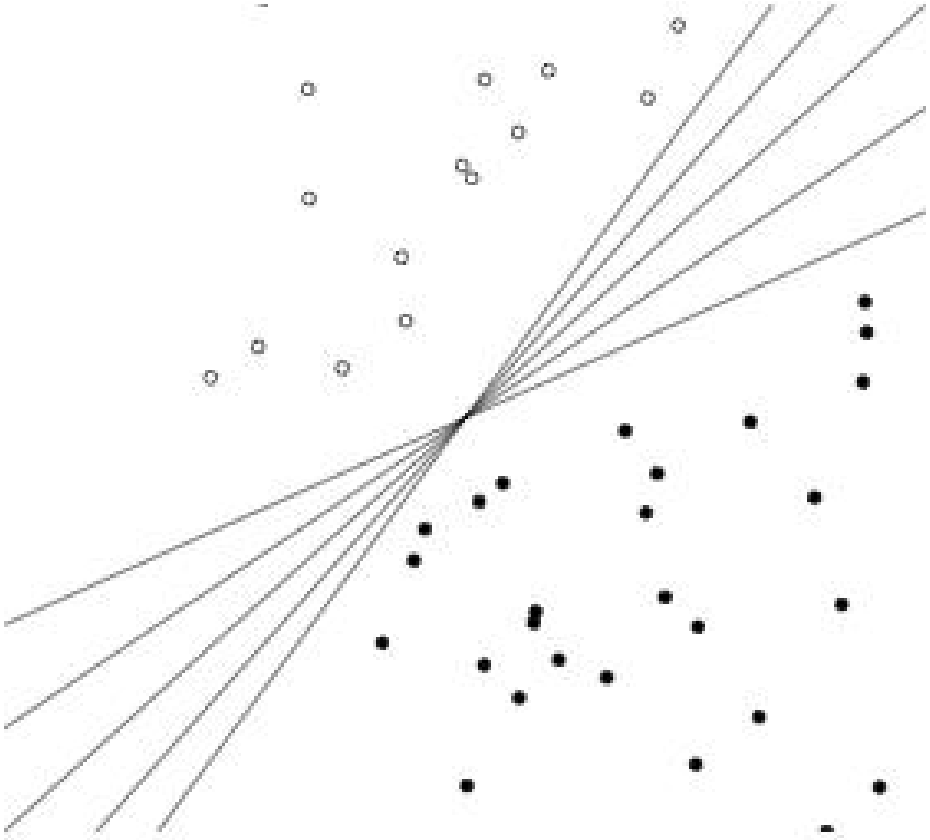
$\overline{c}$.

# 三维空间下的线性分类器



- 三维空间下的分类器是方程$w_1d_1 + w_2d_2 + w_3d_3 = \theta$对应的平面
- 那些满足$w_1d_1 + w_2d_2 + w_3d_3 \geq \theta$的点$(d_1\ d_2\ d_3)$属于类别$c$
- 那些满足$w_1d_1 + w_2d_2 + w_3d_3 < \theta$的点$(d_1\ d_2\ d_3)$属于类别 $\overline{c}$.

# 应该选哪个超平面?



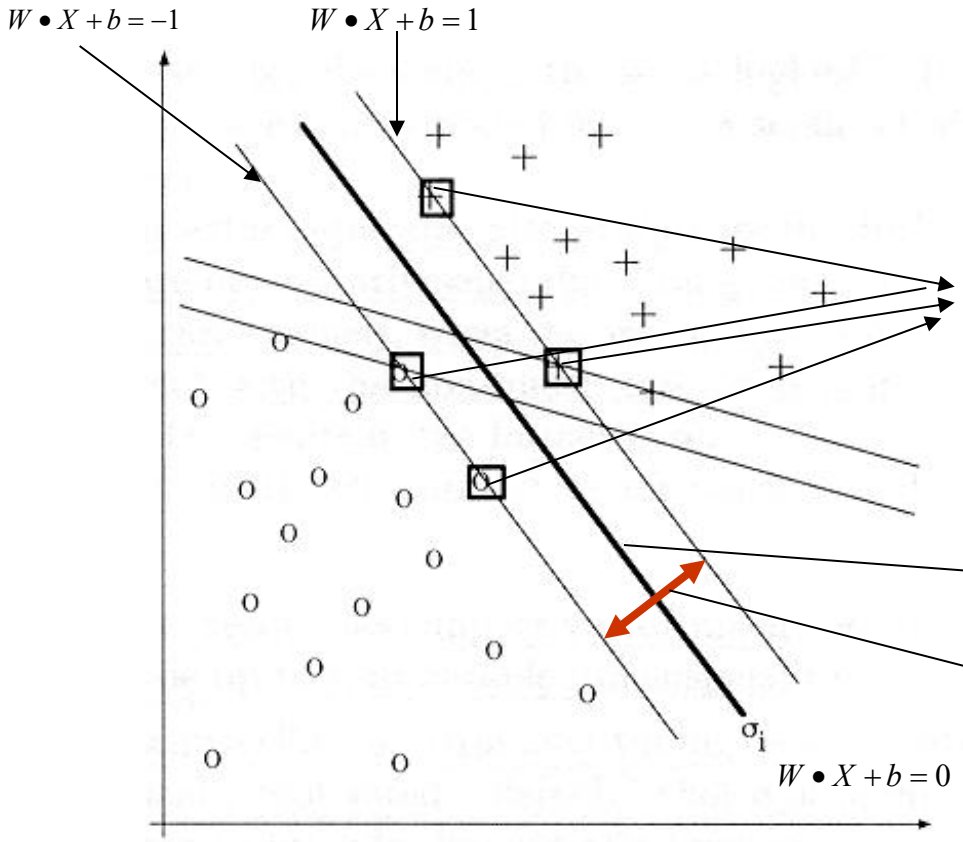
# 本讲内容

- 支持向量机
  - 线性可分：最大间隔
  - 非线性可分：最大间隔+最小错误
  - 空间转换：核函数及核技巧

# 提纲

- 上一讲回顾
- 支持向量机
- 排序学习

# 支持向量机(Support Vector Machines, SVM)




*Support Vector*
*（支持向量）*

***Optimal Separating Hyper plane （最优分隔超平面）***

*Margin*

线性可分情况下，不仅要区分开，而且要使得区分间隔Margin最大。

如上图的训练样本,在线性可分的情况下,存在多个超平面(Hyperplane) (如：H1,H2….)使得这两类被无误差的完全分开。这个超平面被定义为：

$$W \bullet X + b = 0$$

其中,
W:权重向量;
X: 特征向量;
b:截距参数

Optimal Separating Hyperplane（最优超平面）是使得两类的分类间隔(Margin)最大的超平面，即每类中离超平面最近的样本到超平面的距离最大。距离这个最优超平面最近的样本被称为支持向量（Support Vector）。

$$\text{Margin} = \frac{2}{\|W\|}$$

H1平面: $W \bullet X_1 + b \geq 1$

H2平面: $W \bullet X_2 + b \leq -1$

$$y_i[(W \bullet X_i) + b] - 1 \geq 0$$

求解最优超平面就相当于，在上述约束条件下,求2/||W||的最大值 ，即以下损失函数最小值

Minimum: $$\phi(W) = \frac{1}{2}\|W\|^2 = \frac{1}{2}(W \bullet W)$$

Subject to: $$y_i[(W \bullet X_i) + b] - 1 \geq 0$$

- 上述二次优化问题，采用Lagrange方法求解，可得

$$f(X) = \text{sgn}\left( \sum_{i=1}^{n} \alpha_i^* y_i X \bullet X_i + b^* \right)$$

支持向量(Support Vector)

# 非线性可分情况下的处理(方法1)

- 广义最优分类面方法：在线性不可分的情况下，就是某些训练样本不能满足约束条件，因此可以在条件中增加一个松弛项ζ(发音Zeta，也称引入Soft Margin，软边界)，约束条件变成：

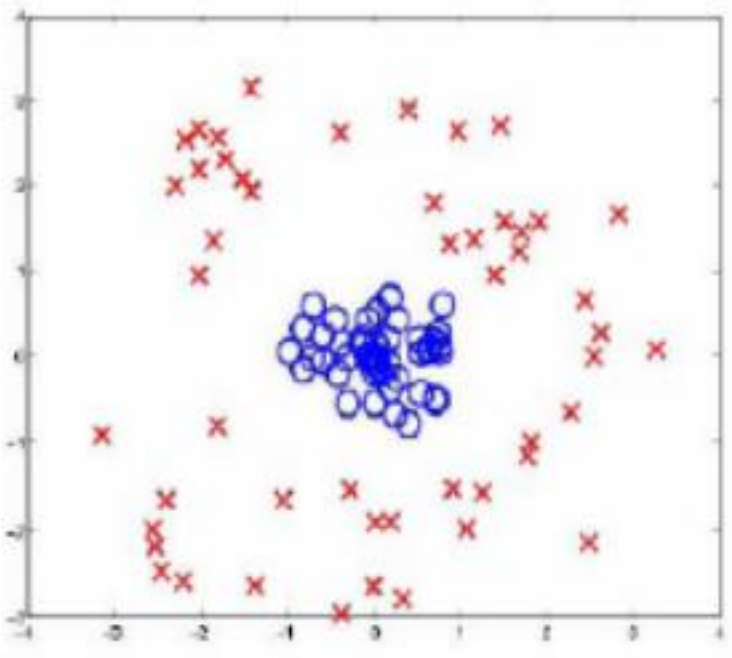
$$y_i[(W \bullet X_i) + b] - 1 + \varsigma_i \geq 0$$

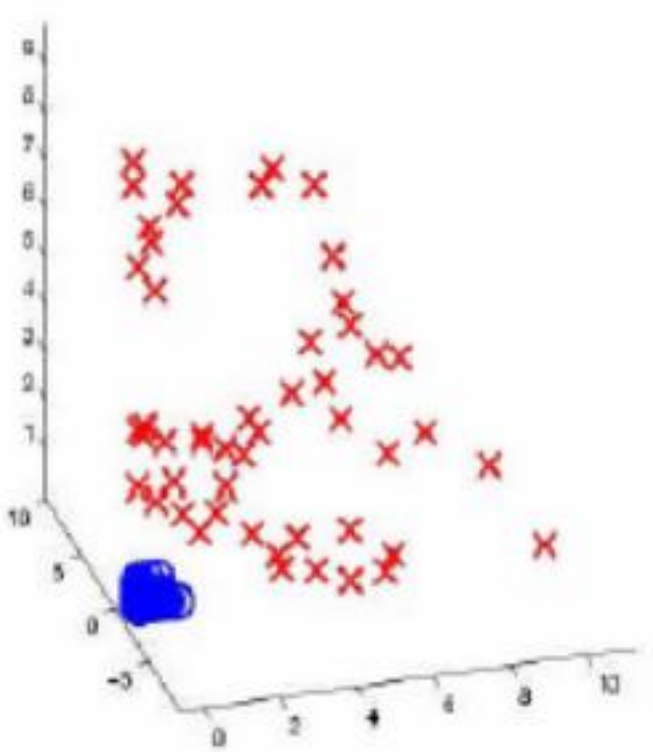
此时的目标函数是求下式的最小值：

$$\Phi(W,\varsigma_i)=\frac{1}{2}(W\bullet W)+C\left(\sum_{i=1}^{n}\varsigma_i\right)$$

这个二次优化问题，同样可以应用拉格朗日法求解

# 非线性可分情况下的处理(方法2)





# 变换到高维空间的支持向量机

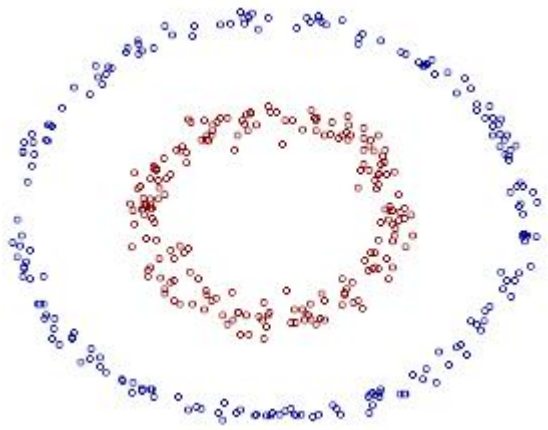
- 采用如下的内积函数(核函数):

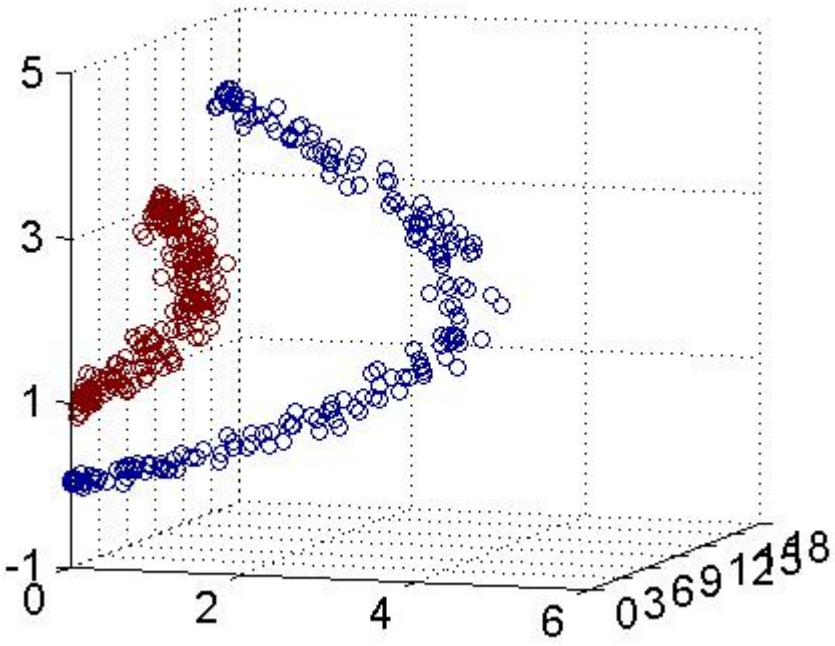
$$K(X, X_i) = [(X \bullet X_i) + 1]^q$$

$$K(X, Xi) = \exp\left\{-\frac{|X - Xi|^2}{\sigma^2}\right\}$$

$$K(X, X_i) = \tanh(\upsilon(X \bullet X_i) + c)$$

将数据映射到高维空间，从而线性可分





判别函数成为：

$$f(X)=\operatorname{sgn}\left(\sum_{i=1}^{n}\alpha_i^{*}y_iK(X,X_i)+b^{*}\right)$$

# 支持向量机

- SVM训练相对较慢，分类速度一般。但是分类效果较好。
- 在面对非线性可分情况时，可以引入松弛变量进行处理或者通过空间变换到另一个线性可分空间进行处理。
- SVM有很多实现工具，SMO/SVM light/SVM torch/LibSVM等等。

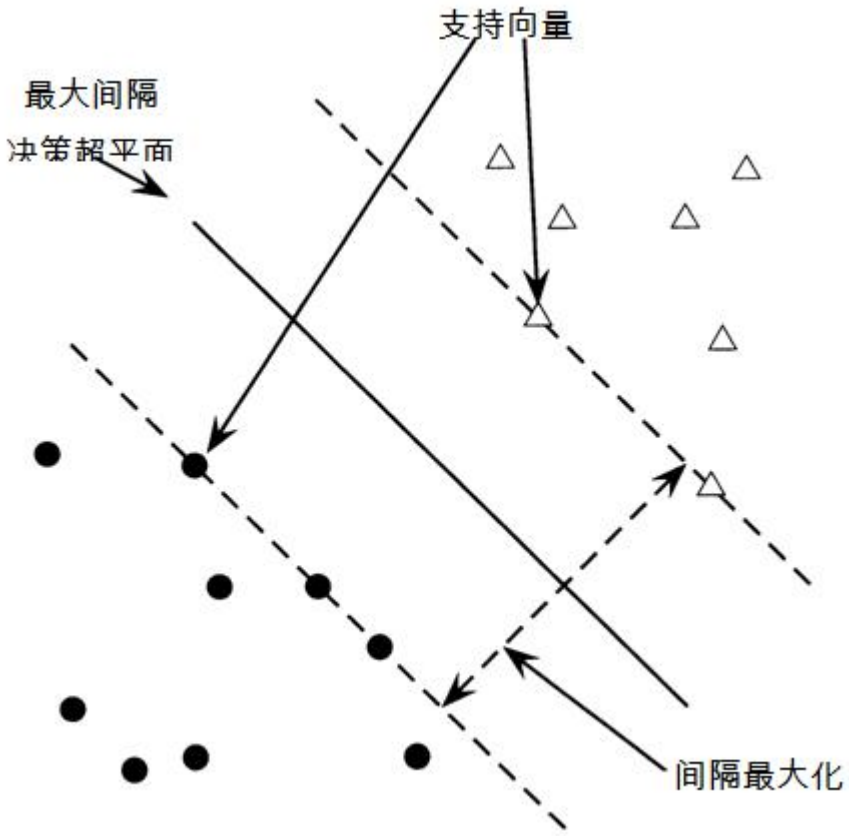
# 支持向量机

- 2-类训练数据
- 决策面

  → 线性分类面
- 准则:离任何数据点最远

  → 确定分类器

  **间隔(margin)**
- 线性分类面的位置基于支持向量(support vector)来定义

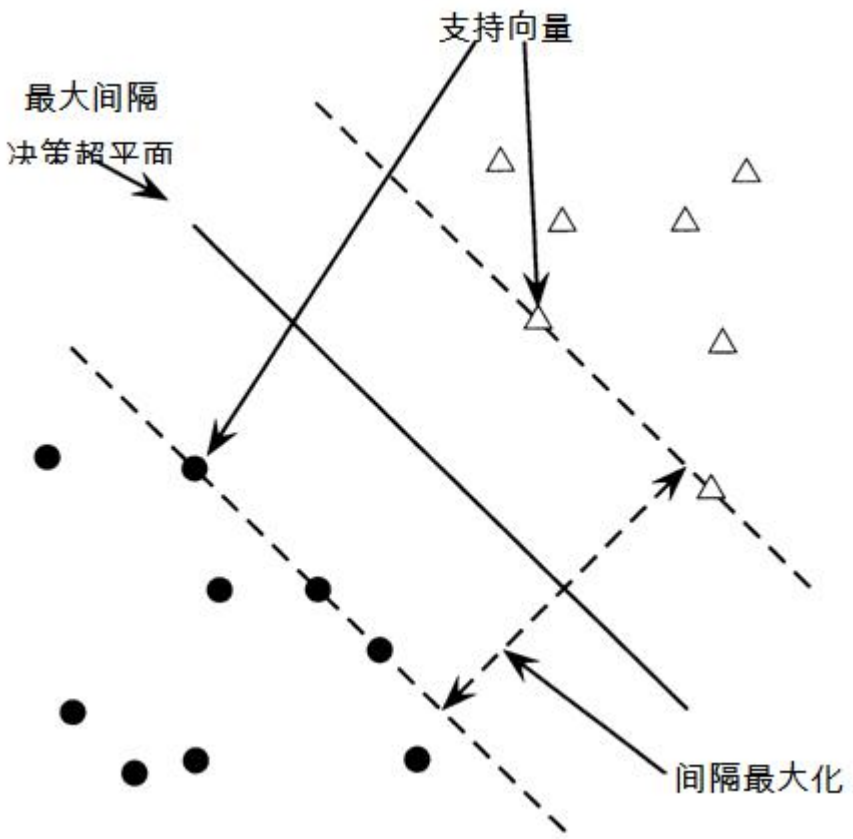


# 为什么要使间隔最大化？

分界面附近的点代表了不确定的分类决策，分类器会以两边各50%的概率做出决策

具有很大分类间隔的分类器不会做出确定性很低的决策，它给出了一个分类的安全间隔

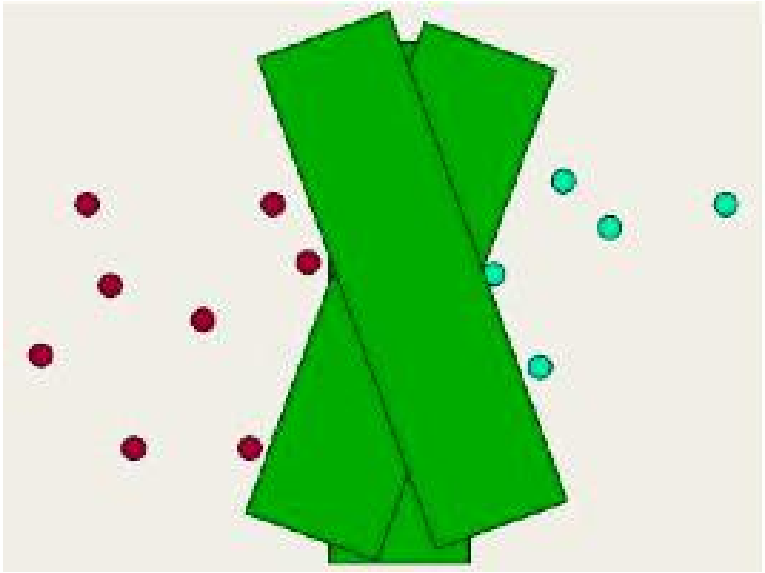
度量中的微小错误和文档中的轻微变化不会导致错误分类



# 为什么要使间隔最大化？

SVM 分类器：在决策面周围有大的间隔

- 与放置(无穷的)决策超平面相比，如果要在类别间放置一个宽间隔，那么选择会少很多
- 减少记忆容量
- 增加测试文档分类泛化能力



# 支持多类的支持向量机

SVMs: 是一个天生的二类分类器

- 实际中存在一些常用的技巧： 构造 |C|个一对多( one-versus-rest 或one-versus-all或OVA)的二类分类器， 选择那个在测试数据上具有最大间隔的分类器所给出的类别
- 另一种方法：建立一系列一对一（one-versus-one）的分类器，选择这些分类器中给出的最多的那个类别。虽然包含 |C|(|C| − 1)/2 个分类器的构建过程，但是分类器的训练时间可能实际上会降低，这是因为每个分类器训练的语料都小很多

# SVM用于支持多类问题

更好的解决方法：结构化SVM

- 将分类问题一般化为如下问题：类别不再只是多个独立的类别标签集合，而可以是相互之间具有关系定义的任意结构化对象的集合。
- 在基于机器学习的排序中将进一步介绍该方法

# 提纲

- 上一讲回顾
- 支持向量机
- 排序学习

# 用于检索排序的机器学习

- 本课程已学习过一系列检索排序的方法
  - Cosine相似度，逆文档频率，BM25，邻近度，回转文档长度归一，（将要学习）Pagerank，…
- 已学习过基于有监督机器学习的文档分类的方法
  - Rocchio，kNN，决策树，等
- 我们也可以使用*机器学习*的方法来对检索到的文档排序？
  - 听上去是一个不错的想法
  - 即“机器学习相关性”（machine-learned relevance）或“排序学习”（learning to rank）

# 用于检索排序的机器学习

- 在过去的10-15年中，这个“不错的想法”已经被主流网络搜索引擎积极研究和部署

- 为什么没有更早的发生?
  - 现代有监督机器学习大约从25年前开始流行...
  - 朴素贝叶斯大约从60年前开始得到应用...

# 用于检索排序的机器学习

- IR社区与ML社区之间的联系并不紧密
- 但是这方面的研究也有很多先驱:
  - Wong, S.K. et al. 1988. Linear structure in information retrieval. *SIGIR 1988.*
  - Fuhr, N. 1992. Probabilistic methods in information retrieval. *Computer Journal.*
  - Gey, F. C. 1994. Inferring probability of relevance using the method of logistic regression. *SIGIR 1994.*
  - Herbrich, R. et al. 2000. Large Margin Rank Boundaries for Ordinal Regression. *Advances in Large Margin Classifiers.*

# 为什么早期尝试并不是很成功/有影响力？

- 有的时候想法也需要时间让人们接受…
- **有限的训练数据**
  - 特别是在现实世界中使用（相对于撰写学术论文而言），很难收集代表真实用户需求的测试收集查询和相关判断以及对返回文档的判断
    - 现在学术界和工业界都已经不在受到这个问题的制约
- 不良的机器学习技术
- 对IR问题的定制不足
- 没有足够的特征来让机器学习方法显示价值

# 为什么以前不需要机器学习？

- 传统IR排序函数仅使用非常少量的特征，例如
  - 词频
  - 逆文档频率
  - 文档长度
- 可以手动调节权重系数
  - 并且有人这样做

# 现代互联网系统使用机器学习

- 现代互联网系统使用大量特征：
  - 直观上有用的特征 - 并非一个统一的模型
  - 锚文本中查询词的对数词频?
  - 页面上彩色字体的查询词?
  - 页面上的图片数量?
  - 页面上的出链(outline)数量?
  - 页面PageRank值?
  - URL 长度?
  - URL 包含 “~”？
  - 页面上一次更新时间?
  - 页面加载速度
- 2008年6月3日的《纽约时报》引述Amit Singhal的话说，谷歌正在使用200多种此类特征（“信号”），据此推论，今天(注：2019年)应该会超过500种

基于布尔权重的学习

基于实数权重的学习

基于序回归的排序学习

# 基本思路

- 词项权重(如tfidf)的目标是为了度量词项的重要性
  - 将一篇文档中所有词项的权重加起来便可以计算文档和查询的相关度，基于该相关度可以对所有文档排序
- 上述过程可以想象成一个文本分类问题
  - 词项权重可以从已判定的训练集合中学习得到
- 上述研究方法被归入一类称为机器学习的相关度(machine learned relevance )或排序学习(learning to rank)

# 权重学习

主要方法：

- 给定训练样例集合，每个样例表示为三元组<q, d, R(d,q)>
  - 最简单的情况： 相关性判定结果$R(d, q)$要么为1 (相关)，要么为0 （不相关）
  - 更复杂的情况：多级相关
- 从上述样例中学习权重，使得学到的评分接近训练集中的相关性判定结果。

- 下面以域加权评分(*Weighted zone scoring*)为例来介绍

# 域加权评分

- 给定查询以及具有3个域(author、title、body)的文档集合

- 域加权评分对每个域都有个独立的权重，比如 $g_1, g_2, g_3$
- 并非所有域的重要性都完全一样:

  比如： author<title<body

  $\rightarrow g_1 = 0.2, g_2 = 0.3, g_3 = 0.5$ (系数总和为1)
- 如果查询词项出现在某个域中，那么该域上的得分为1，否则为0 (布尔权重)

**例子**

查询词项仅仅在title和body域中出现
于是文档得分为： $(0.3 \cdot 1) + (0.5 \cdot 1) = 0.8.$

# 域加权评分的一般化

给定 $q$ 和 $d$, 域加权评分方法通过计算所有文档域得分的线性组合，赋予(q,d)一个[0,1]内的得分

- 考虑一系列文档，每篇文档包含 $l$ 个域
- 令 $g_1, \ldots, g_l \in [0, 1]$, 且有 $\sum_{i=1}^{l} g_i = 1$
- 对于 $1 \leq i \leq l$，令 $s_i$ 为q和文档第i个域的 布尔匹配得分
  - 比如， $s_i$可以是将域当中查询词项出现与否映射为0或1的任意布尔函数。

**域加权评分也称为排序式布尔检索**

$$\sum_{i=1}^{l} g_i s_i$$

# 域加权评分及权重学习

- 域加权评分可以看成基于布尔匹配值的线性函数学习，每个布尔匹配值对应一个域
- 坏消息：权重学习需要训练集，而人工产生训练集中的相关性判断费时费力
  - 特别是在动态语料库环境下 (比如Web)
- 好消息：将权重$g_i$的学习简化为一个简单的优化问题

# 域加权评分中的权重学习

- 最简单的情况：每篇文档有两个域---- title、body
- 域加权评分的公式如下：

$$\sum_{i=1}^{l} g_i s_i \qquad (2)$$

- 给定 $q$、$d$, 要根据$d$的不同域与查询$q$的匹配情况计算 $s_T(d, q)$ 及$s_B(d, q)$
- 利用 $s_T(d, q)$ 、 $s_B(d, q)$ 及常数 $g \in [0, 1]$我们可以得到$(q,d)$的匹配得分：

$$score(d, q) = g \times s_T(d, q) + (1 - g) \times s_B(d, q) \qquad (3)$$

# 基于训练样例学习权重$g$

## 例子

| Example | DocID | Query | $s_T$ | $s_B$ | Judgment |
|---|---|---|---|---|---|
| $\Phi_1$ | 37 | linux | 1 | 1 | Relevant |
| $\Phi_2$ | 37 | penguin | 0 | 1 | Nonrelevant |
| $\Phi_3$ | 238 | system | 0 | 1 | Relevant |
| $\Phi_4$ | 238 | penguin | 0 | 0 | Nonrelevant |
| $\Phi_5$ | 1741 | kernel | 1 | 1 | Relevant |
| $\Phi_6$ | 2094 | driver | 0 | 1 | Relevant |
| $\Phi_7$ | 3194 | driver | 1 | 0 | Nonrelevant |

- 训练样例: 三元组形式 $\Phi_j = (d_j, q_j, r(d_j, q_j))$
- 给定训练文档 $d_j$ 和训练查询$q_j$，通过人工判定得到$r(d_j, q_j)$ (要么相关要么不相关)

# 基于训练样例学习权重$g$

## 样例

| Example | DocID | Query | $s_T$ | $s_B$ | Judgment |
|---|---|---|---|---|---|
| $\Phi_1$ | 37 | linux | 1 | 1 | Relevant |
| $\Phi_2$ | 37 | penguin | 0 | 1 | Nonrelevant |
| $\Phi_3$ | 238 | system | 0 | 1 | Relevant |
| $\Phi_4$ | 238 | penguin | 0 | 0 | Nonrelevant |
| $\Phi_5$ | 1741 | kernel | 1 | 1 | Relevant |
| $\Phi_6$ | 2094 | driver | 0 | 1 | Relevant |
| $\Phi_7$ | 3194 | driver | 1 | 0 | Nonrelevant |

- 对每个训练样例 $\Phi_j$ ，我们有布尔值 $s_T(d_j, q_j)$和$s_B(d_j, q_j)$，利用这些布尔值可以计算：

$$score(d_j, q_j) = g \cdot s_T(d_j, q_j) + (1 - g) \cdot s_B(d_j, q_j) \qquad (4)$$

# 权重学习

- 比较该得分和人工判定结果的差异
  - 人工判定结果中，相关记为1，不相关记为0
- 评分函数的错误定义为：

$$\epsilon(g,\Phi_j) = (r(d_j, q_j) - \text{score}(d_j, q_j))^2 \quad (5)$$

- 于是，整个训练集上的总错误为：

$$\sum_j \epsilon(g,\Phi_j) \quad (6)$$

- 权重$g$的学习归结为选择使得上述总错误率最小的那个$g$

# 课堂练习: 寻找使总错误$\epsilon$ 最小的$g$值

## 训练样例

| Example | DocID | Query | $s_T$ | $s_B$ | Judgment |
|---|---|---|---|---|---|
| $\Phi_1$ | 37 | linux | 1 | 1 | Relevant |
| $\Phi_2$ | 37 | penguin | 0 | 1 | Nonrelevant |
| $\Phi_3$ | 238 | system | 0 | 1 | Relevant |
| $\Phi_4$ | 238 | penguin | 0 | 0 | Nonrelevant |
| $\Phi_5$ | 1741 | kernel | 1 | 1 | Relevant |
| $\Phi_6$ | 2094 | driver | 0 | 1 | Relevant |
| $\Phi_7$ | 3194 | driver | 1 | 0 | Nonrelevant |

1. 相关记为 1，不相关记为0
2. 计算得分:

$$score(d_j, q_j) = g \cdot s_T(d_j, q_j) + (1 - g) \cdot s_B(d_j, q_j)$$

3. 计算总错误: $\sum_j \epsilon(g, \Phi_j)$，其中

$$\epsilon(g, \Phi_j) = (r(d_j, q_j) - \text{score}(d_j, q_j))^2$$

4. 选择使得总错误最小的$g$值

# 习题解答

1. 计算评分 score($d_j$ , $q_j$ )

$score(d_1, q_1) = g \cdot 1 + (1-g) \cdot 1 = g + 1 - g = 1$

$score(d_2, q_2) = g \cdot 0 + (1-g) \cdot 1 = 0 + 1 - g = 1 - g$

$score(d_3, q_3) = g \cdot 0 + (1-g) \cdot 1 = 0 + 1 - g = 1 - g$

$score(d_4, q_4) = g \cdot 0 + (1-g) \cdot 0 = 0 + 0 = 0$

$score(d_5, q_5) = g \cdot 1 + (1-g) \cdot 1 = g + 1 - g = 1$

$score(d_6, q_6) = g \cdot 0 + (1-g) \cdot 1 = 0 + 1 - g = 1 - g$

$score(d_7, q_7) = g \cdot 1 + (1-g) \cdot 0 = g + 0 = g$

2. 计算总错误 $\sum_j \epsilon(g, \Phi_j)$

$(1-1)^2 + (0-1+g)^2 + (1-1+g)^2 + (0-0)^2 + (1-1)^2 +$

$(1-1+g)^2 + (0-g)^2 = 3g^2 + (1-g)^2 = 4g^2 - 2g + 1$

3. 选择使得总错误最小的$g$值

求解$g=0.25$

基于布尔权重的学习

基于实数权重的学习

基于序回归的排序学习

# 一个简单的机器学习评分的例子

- 迄今为止，我们都是考虑一种非常简单的情况，即我们考虑的是布尔相关值的组合
- 现在考虑更一般的情况

# 一个简单的机器学习评分的例子

- 设置: 评分函数是两个因子的线性组合:
  1. 查询和文档的向量空间相似度评分 (记为 $\alpha$)
  2. 查询词项在文档中存在的最小窗口宽度 (记为 $\omega$)
     - 查询词项的邻近度(proximity)往往对文档的主题相关性具有很强的指示作用
     - 查询词项的邻近度给出了隐式短语的实现

- 因此，我们的一个因子取决于查询词项在文档中的词袋统计量，另一个因子取决于邻近度权重

# 一个简单的机器学习评分的例子

给定训练集，对每个样例计算

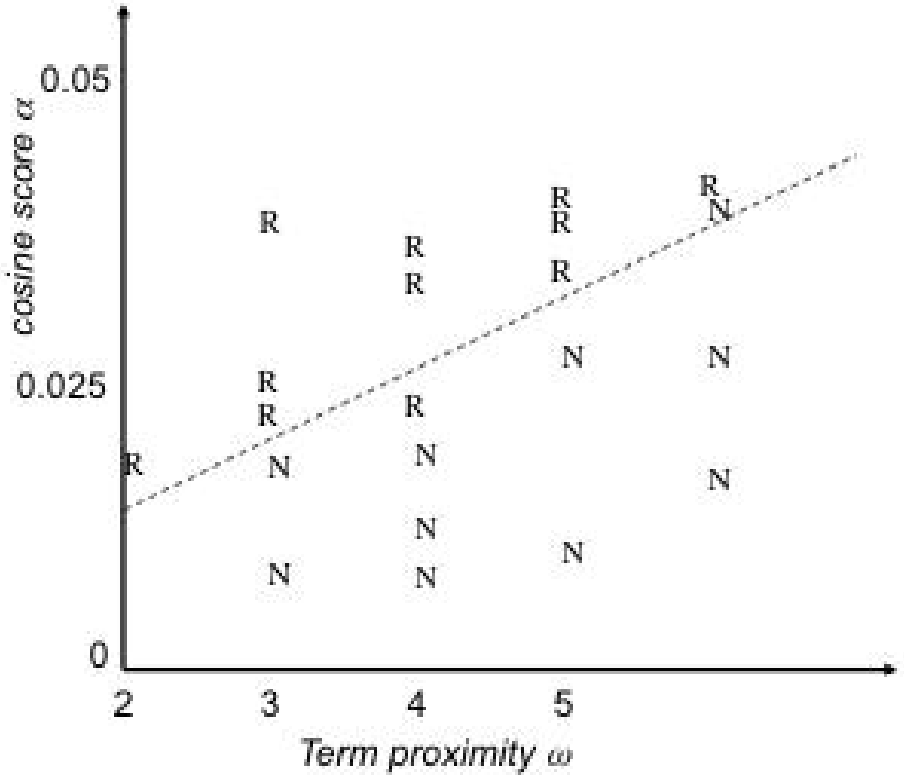
- 向量空间余弦相似度 $\alpha$
- 窗口宽度 $\omega$

上述结果构成训练集，与前面不同的是，我们引入的是两个实数特征因子 $(\alpha, \omega)$

## 例子

| Example | DocID | Query | Cosine score | $\omega$ | Judgment |
|---|---|---|---|---|---|
| $\Phi_1$ | 37 | linux operating system | 0.032 | 3 | *relevant* |
| $\Phi_2$ | 37 | penguin logo | 0.02 | 4 | *nonrelevant* |
| $\Phi_3$ | 238 | operating system | 0.043 | 2 | *relevant* |
| $\Phi_4$ | 238 | runtime environment | 0.004 | 2 | *nonrelevant* |
| $\Phi_5$ | 1741 | kernel layer | 0.022 | 3 | *relevant* |
| $\Phi_6$ | 2094 | device driver | 0.03 | 2 | *relevant* |
| $\Phi_7$ | 3191 | device driver | 0.027 | 5 | *nonrelevant* |
| ... | ... | ... | ... | ... | ... |

# 2-D 平面上的图展示



# 一个简单的机器学习评分的例子

- 同样，相关记为1，不相关记为0
- 我们的目标是寻找一个评分函数，该函数能够组合特征因子的值，并尽量接近0或1
- 希望该函数的结果尽量与训练集上的结果保持一致

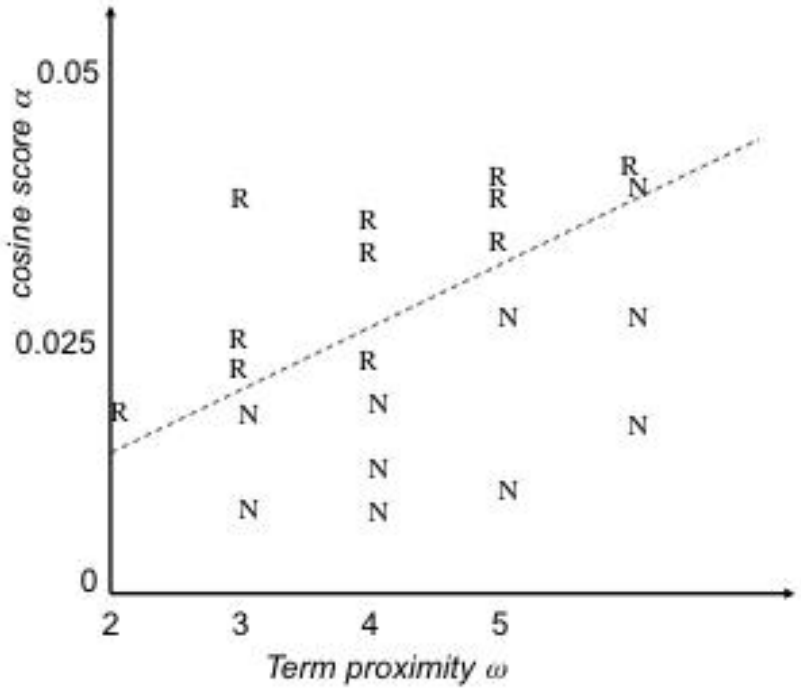
不失一般性，线性分类器可以采用下列通用形式：

$$Score(d, q) = Score(\alpha, \omega) = a\alpha + b\omega + c, \qquad (7)$$

公式中的系数 $a$、$b$、 $c$ 可以从训练语料中学到

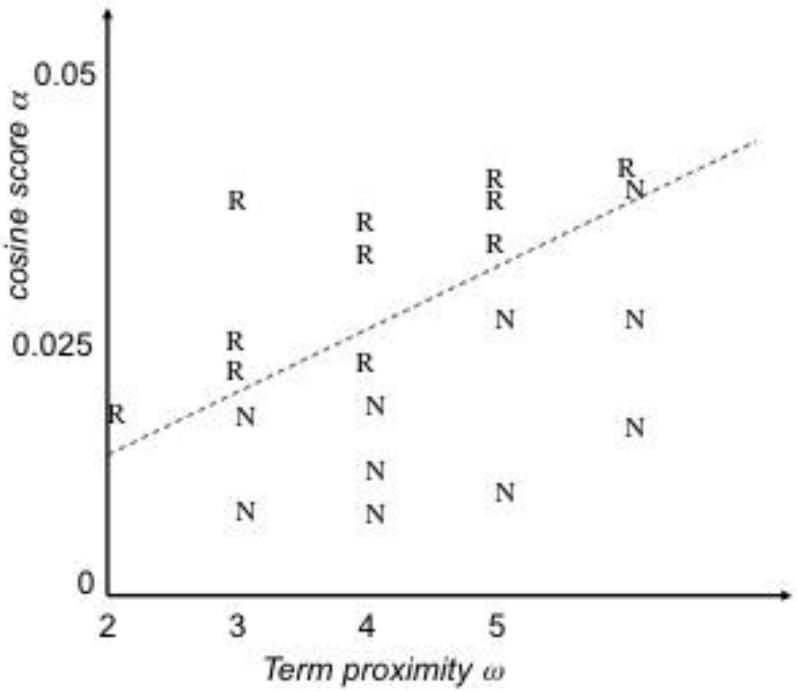
# 一个简单的机器学习评分的例子

- 函数$Score(\alpha,\omega)$可以看成悬挂在右图上空的一个曲面
- 理想情况下，该曲面在R表示的点之上的函数值接近于1，而在N表示的点之上的函数值接近于0



# 一个简单的机器学习评分的例子

- 引入阈值 θ
- 如果$Score(\alpha, \omega) > \theta$, 则认为文档相关，否则认为不相关
- 从前面的 SVM我们可以知道, $Score(\alpha, \omega) = \theta$ 的点构成一条直线 (图中虚线) → 它能将相关和不相关文档线性地分开



# 一个简单的机器学习评分的例子

因此，给定训练集情况下判定相关/不相关的问题就转变成一个学习问题，如前面提到，相当于学习一条能够将训练集中相关文档和不相关文档分开的虚线

- 在$\alpha$-$\omega$ 平面上, 该直线可以写成基于$\alpha$ 和$\omega$ (分别表示斜率和截距)的一个方程
- 前面已经介绍选择直线的线性分类方法
- 如果能够构建大规模训练样例，那么就可以避免手动去调节评分中的参数
- 瓶颈： 维护一个合适的有代表性的训练样例需要较大的开销，而且这些样例的相关性判定需要专家来进行

# 基于机器学习的检索结果排序

- 上面的例子可以很容易地推广到包含多个变量的函数
- 除余弦相似度及查询词项窗口之外，存在大量其他的相关性度量方法，如 PageRank之类的指标、文档时间、域贡献、文档长度等等
- 如果训练文档集中的这些指标可以计算出来，加上相关性判定，就可以利用任意数目的指标来学习分类器。

# Microsoft Research 在2010.6.16公布的134个特征

http://research.microsoft.com/en-us/projects/mslr/feature.aspx

**Zones**: body, anchor, title, url, whole document

**Features**: query term number, query term ratio, stream length, idf, sum of term frequency, min of term frequency, max of term frequency, mean of term frequency, variance of term frequency, sum of stream length normalized term frequency, min of stream length normalized term frequency, max of stream length normalized term frequency, mean of stream length normalized term frequency, variance of stream length normalized term frequency, sum of tf*idf, min of tf*idf, max of tf*idf, mean of tf*idf, variance of tf*idf, boolean model, vector space model, BM25, LMIR.ABS, LMIR.DIR, LMIR.JM, number of slash in url, length of url, inlink number, outlink number, PageRank, SiteRank, QualityScore, QualityScore2, query-url click count, url click count, url dwell time.

# 基于机器学习的检索结果排序

然而，利用上述方法来进行IR的排序未必是正确的问题处理方法

- 统计学家通常将问题分成分类问题 (预测一个类别型变量) 和回归问题 (预测一个实数型变量)
- 在这两者之间，有一个特别的称为序回归(**ordinal regression)的领域，其目标是预测一个序**
- 基于机器学习的Ad hoc检索可以看成是一个序回归问题，这是因为检索的目标是，给定q的情况下，对所有的文档进行排序

基于布尔权重的学习

基于实数权重的学习

基于序回归的排序学习

# 将IR排序问题看成序回归

为什么将IR排序问题看成一个序回归问题？

- 对于同一查询，文档之间可以按照相对得分排序即可，并不一定要求每篇文档有一个全局的绝对得分
- 因此，只需要一个排序，而不要得到相关度的绝对得分，问题空间可以减小

这对于Web检索来说特别重要，Web检索中排名最高的一些结果非常重要

**利用结构化SVM 进行IR排序**

利用结构化SVM框架可以处理IR排序问题，对于一个查询，预测的类别是结果的排序

# 排序SVM的构建

- 给定一些已经判定的查询
- 对训练集中的每条查询$q$, 我们都有针对该查询的一系列文档集合，这些文档已经由人工按照其与查询的相关度排序
- 对每个文档、查询对，构造特征向量 $\psi_j = \psi(d_j, q)$，这里的特征可以采用前面讨论的特征
- 对于两篇文档$d_i$ 和$d_j$ ，可以计算特征向量之间的差异向量：：

$$\Phi(d_i, d_j, q) = \psi(d_i, q) - \psi(d_j, q) \tag{8}$$

# 排序SVM的构建

- 依据假设， $d_i$ 、$d_j$ 中的一个更相关
- 如果 $d_i$ 比 $d_j$更相关，记为$d_i <$ $d_j$ (在检索结果中，$d_i$ 应该出现在 $d_j$ 前面), 那么分配给$\Phi(d_i, d_j, q)$向量的类别为 $y_{ijq} = +1$，否则为 $-1$

- 学习的目标是建立一个分类器，满足：

$$\vec{w}^{\mathrm{T}} \Phi(d_i, d_j, q) > 0 \text{ iff } d_i < d_j \qquad (9)$$

# 排序SVM(Ranking SVM)

- 该方法已经被用于构建排序函数，在标准数据集的IR评测中表现的性能优于普通的人工排序函数
- 参考《信息检索导论》第239页的一些参考文献

# 注意: 线性 vs. 非线性权重计算

- 前面介绍的方法也代表当前研究中的主要做法，即将特征进行线性组合
- 但是很多传统IR方法中还包括对基本量的非线性放缩方法，比如对词项频率或IDF取对数
- 当前来说，机器学习方法很擅长对线性组合的权重进行优化，但是并不擅长基本量的非线性放缩处理。
- 该领域仍然存在大量人工特征工程的方法

# 排序学习总结

- 排序学习算法现在一般分为以下三类
  - Pointwise (即本讲介绍的权重学习方法)
    - 每个文档是一个训练样本，预测文档相关/不相关
  - Pairwise （即本讲介绍的序回归方法）
    - 文档对构成一个训练样本，预测一个文档相关性是否高于另一个文档
  - Listwise（基于列表的排序学习，本讲未介绍）
    - 一个文档排序列表构成一个训练样本，预测最优排序

虽然近年来基于深度学习和大规模预训练语言模型的方法已成功应用于IR，排序学习仍然是一种整合不同文本特征的有效方法