基于向量空间模型的文本分类

内容

- 基于向量空间的分类方法
- Rocchio方法
  - 基于质心或原型(*prototype*)将整个向量空间划分成多个区域
- *k*NN(*k* 近邻)方法
  - 将 *k* 个最近邻文档所属的主类别赋给测试文档
- 线性分类器
  - 指基于特征的简单线性组合就可以对文档进行分类的分类器

内容

- 基于向量空间的分类方法
- Rocchio方法
- *k*NN(*k* 近邻)方法
- 线性分类器

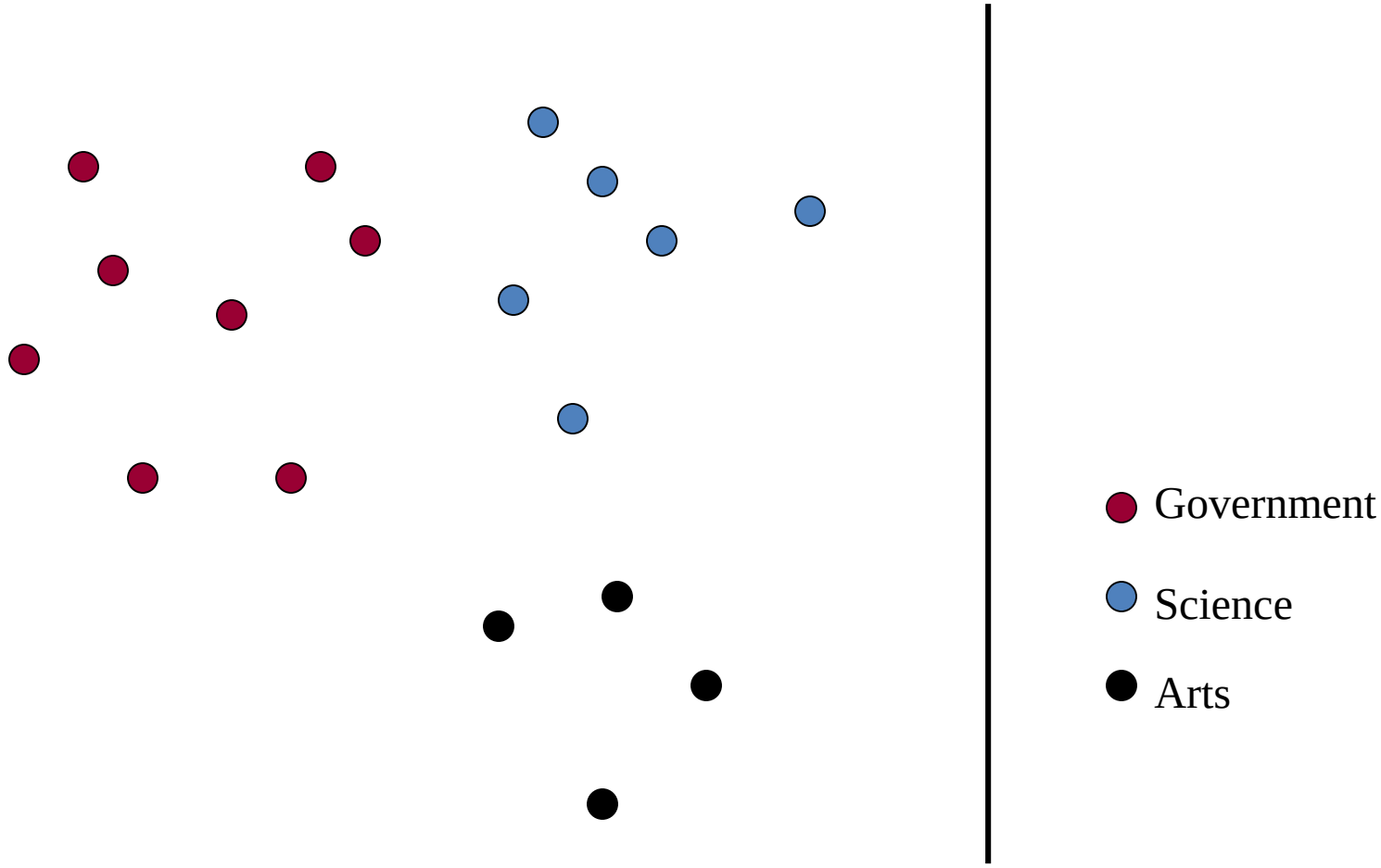
向量空间表示

- 每个文档表示成一个向量，向量的每一维表示一个term
- 向量可以归一化(normalize)成单位长度
- 高维向量空间
  - 维度非常高(term的数量)
  - 每个term就是一个坐标轴
  - 文档表示为空间的向量

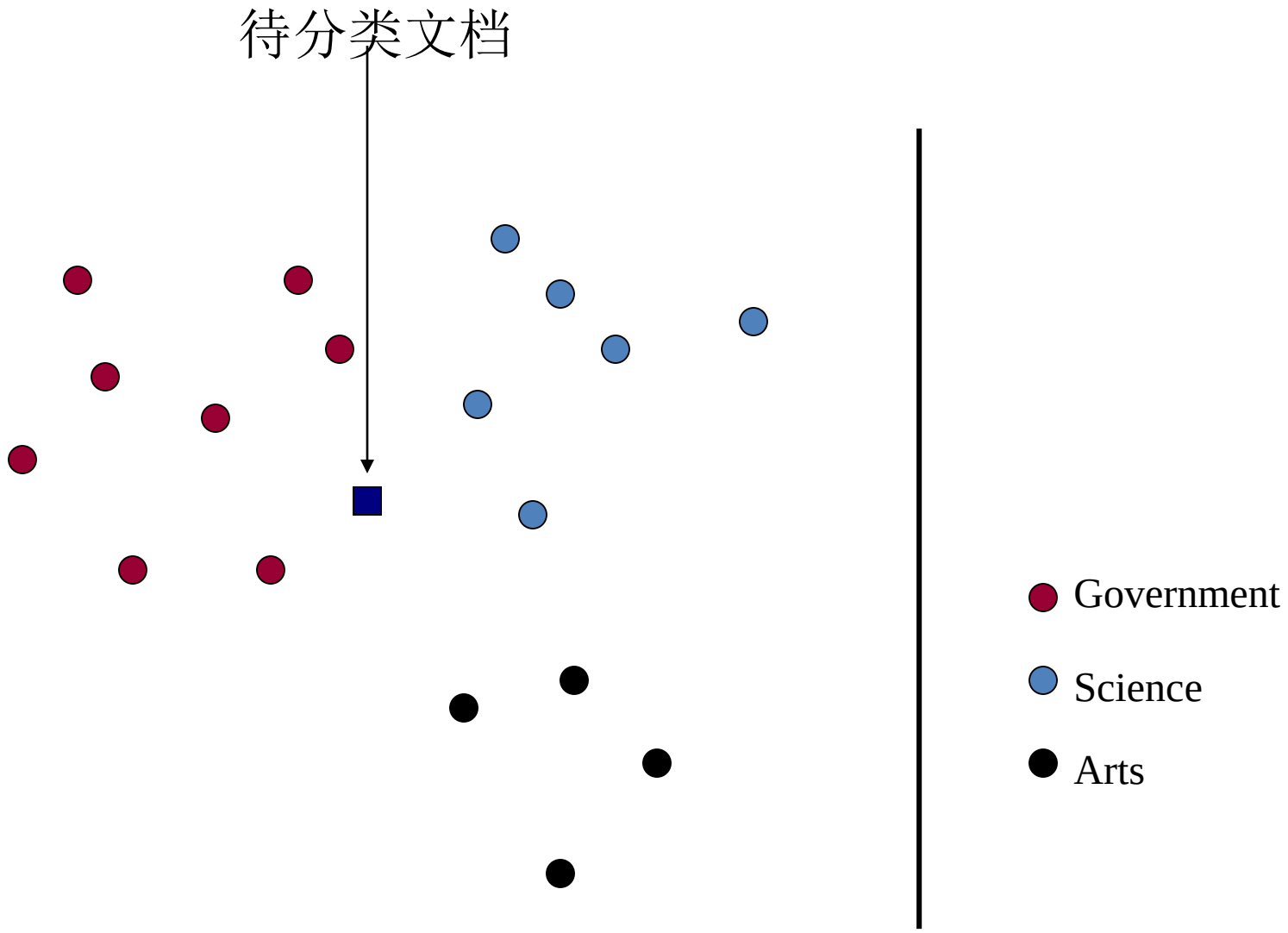
思路

- 向量空间模型
  - 词项-文档矩阵：二值→ 频数 → 权重矩阵(tf-idf值)
  - 相关性=向量距离：欧氏距离→夹角→余弦相似度
- 利用向量空间模型进行文本分类的思路主要基于邻近假设(contiguity hypothesis)
  - ①同一类的文档会构成一个邻近区域，
  - ②而不同类的邻近区域之间是互不重叠的。
- 核心问题:
  - 如何找到分类面决策边界(**decision boundary**)

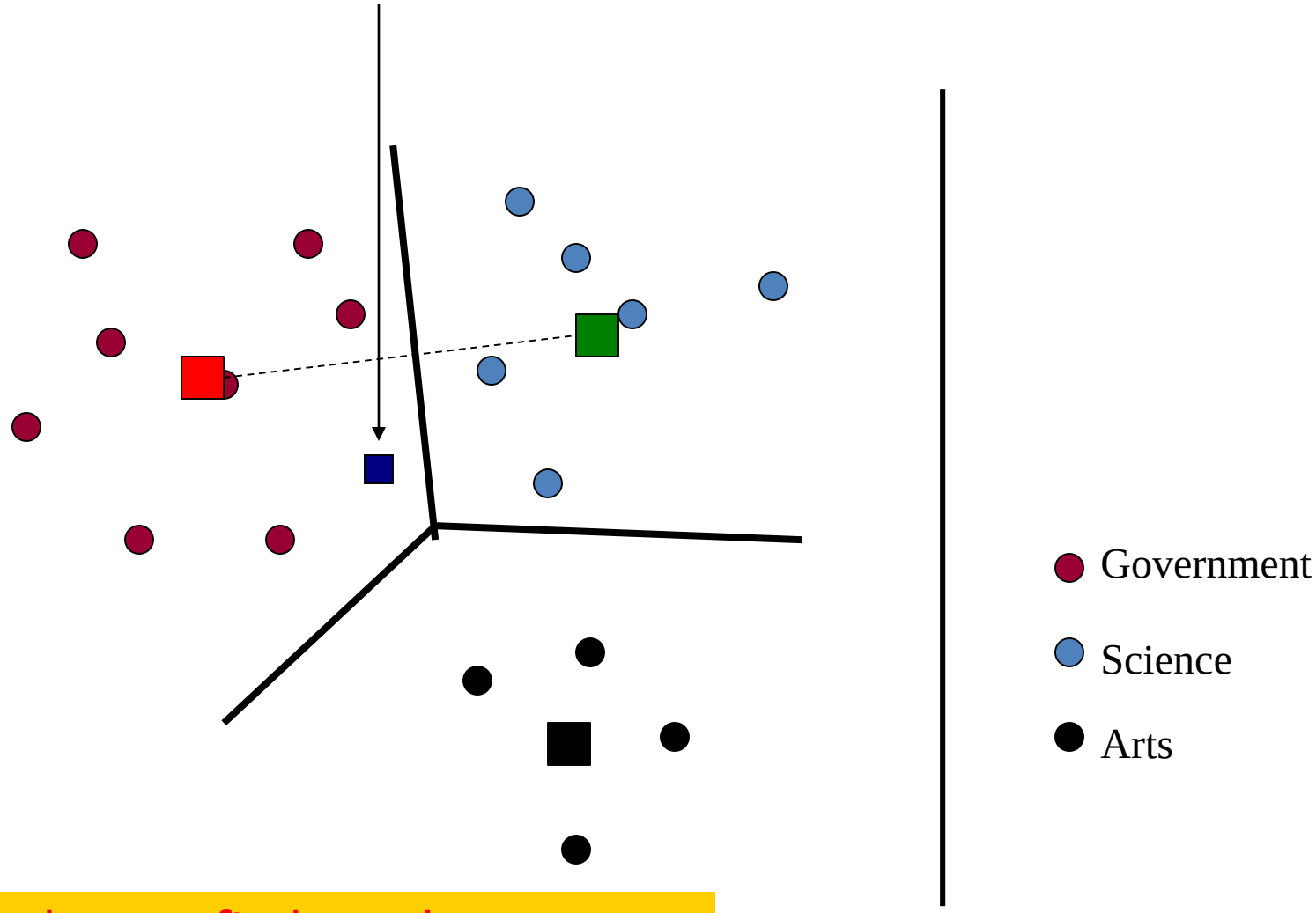
Documents in a Vector Space



Test Document of what class?



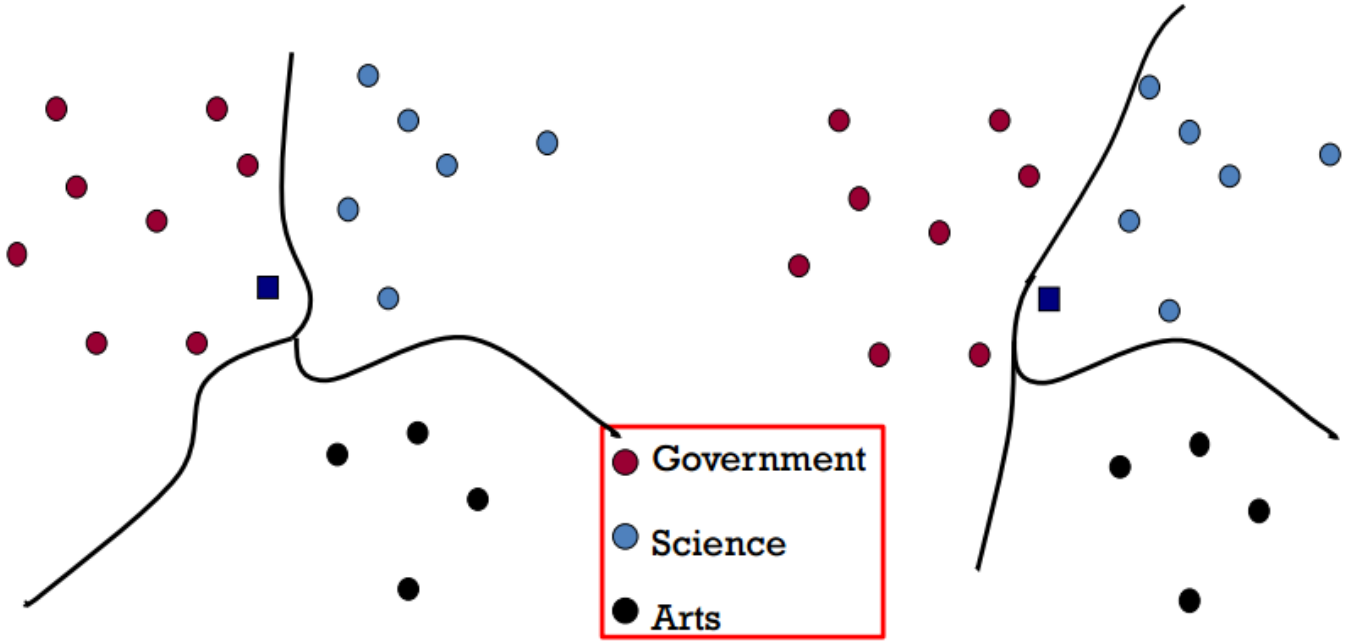
Test Document = Government



Our focus: how to find good separators

分类面选择

- Test Document = Government?
- Test Document = Science?
- 给定训练集可能存在多种分类面方案
- 选定的分类面方案有可能将测试文档归入错误的类中



内容

- 基于向量空间的分类方法
- Rocchio方法
- *k*NN(*k* 近邻)方法
- 线性分类器

Rocchio方法进行向量空间分类的思路

- 利用质心(centroid)来定义分类边界。
- 一个类别c的质心可以通过类中文档向量的平均向量或者质心向量来计算，即

$$\vec{\mu}(c)=\frac{1}{|D_c|}\sum_{d\in D_c}\vec{v}(d)$$

  - 其中，D_c是文档集D中属于类别c的文档子集：$D_c=\{d:<d,c>\in D\}$。 这里将d归一化的文档向量记为$\vec{v}(d)$

Rocchio算法

- (1)计算每个类的中心向量
  - 中心向量是所有文档向量的算术平均
- (2)将每篇测试文档分到离它最近的那个中心向量

TRAINROCCHIO($\mathbb{C}$, $\mathbb{D}$)

1 **for each** $c_j \in \mathbb{C}$

2 **do** $D_j \leftarrow \{d : \langle d, c_j \rangle \in \mathbb{D}\}$

3 $\quad \vec{\mu}_j \leftarrow \frac{1}{|D_j|} \sum_{d \in D_j} \vec{v}(d)$

4 **return** $\{\vec{\mu}_1, \ldots, \vec{\mu}_J\}$

APPLYROCCHIO($\{\vec{\mu}_1, \ldots, \vec{\mu}_J\}, d$)

1 **return** $\arg\min_j |\vec{\mu}_j - \vec{v}(d)|$

图 14-4　Rocchio 分类方法中的训练和测试

Rocchio算法中的决策边界

- 利用质心(centroid)来定义分类边界。
- 两类的边界由哪些到两个类质心等距的点集组成(超平面)。
  - 如图有 $|a_1|=|a_2|$、$|b_1|=|b_2|$和$|c_1|=|c_2|$

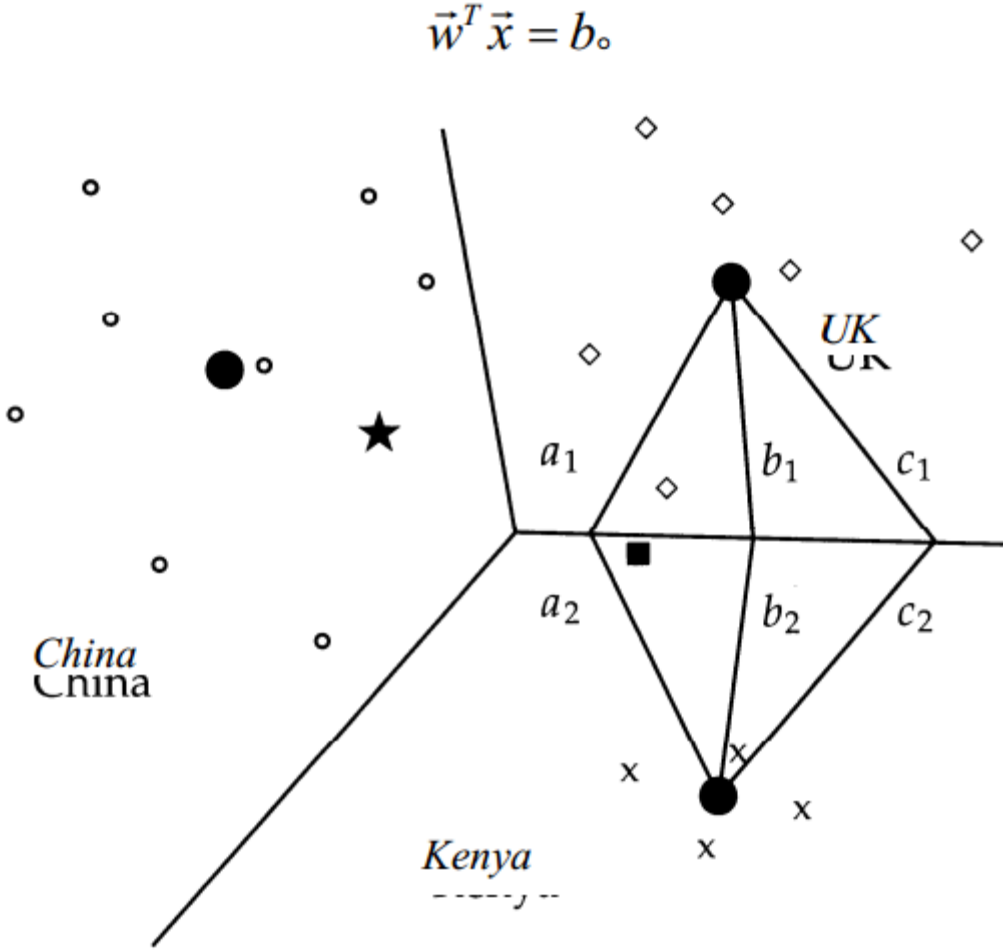


图 14-3　Rocchio 分类方法示意图

分类超平面

- 二维平面上的一条直线在M维空间中可以推广成一个超平面，上面的点满足：

$$\{\vec{x} \mid \vec{w}^T \vec{x} = b\}$$

  - $\vec{w}$称为超平面上的M维法向量(normal vector)
  - b是一个常数，determines the offset of the hyperplane from the origin
- 维度情况
  - 二维 → 直线 $w_1x_1 + w_2x_2 = b$
  - 三维 → 平面 $w_1x_1 + w_2x_2 + w_3x_3 = b$
  - 三维以上 → 超平面
- A hyperplane divides R^M into two half spaces:$\{\vec{x} \mid \vec{w}^T \vec{x} \le b\}$ and $\{\vec{x} \mid \vec{w}^T \vec{x} > b\}$

Rocchio分类示例

表13-1　用于参数估计的数据

	文档ID	文档中的词	属于c=China类?
训练集	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
测试集	5	Chinese Chinese Chinese Tokyo Japan	?

表14-1　表13-1中数据对应的文档向量及类别质心向量

向量	词项权重					
	Chinese	Japan	Tokyo	Macao	Bejing	Shanghai
$\vec{d}_1$	0	0	0	0	0.602	
$\vec{d}_2$	0	0	0	0	0	0.602
$\vec{d}_3$	0	0	0	0.602	0	0
$\vec{d}_4$	0	0.602	0.602	0	0	0

$$(1+\log tf)\log\frac{N}{df}$$

表14-1　表13-1中数据对应的文档向量及类别质心向量

向量	词项权重					
	Chinese	Japan	Tokyo	Macao	Bejing	Shanghai
$\vec{d}_1$	0	0	0	0	1.0	0
$\vec{d}_2$	0	0	0	0	0	1.0
$\vec{d}_3$	0	0	0	1.0	0	0
$\vec{d}_4$	0	0.71	0.71	0	0	0

向量归一化

$$\frac{0.602}{\sqrt{0.602^2+0.602^2}}$$

Rocchio分类示例

测试集	5	Chinese Chinese Chinese Tokyo Japan

表14-1　表13-1中数据对应的文档向量及类别质心向量

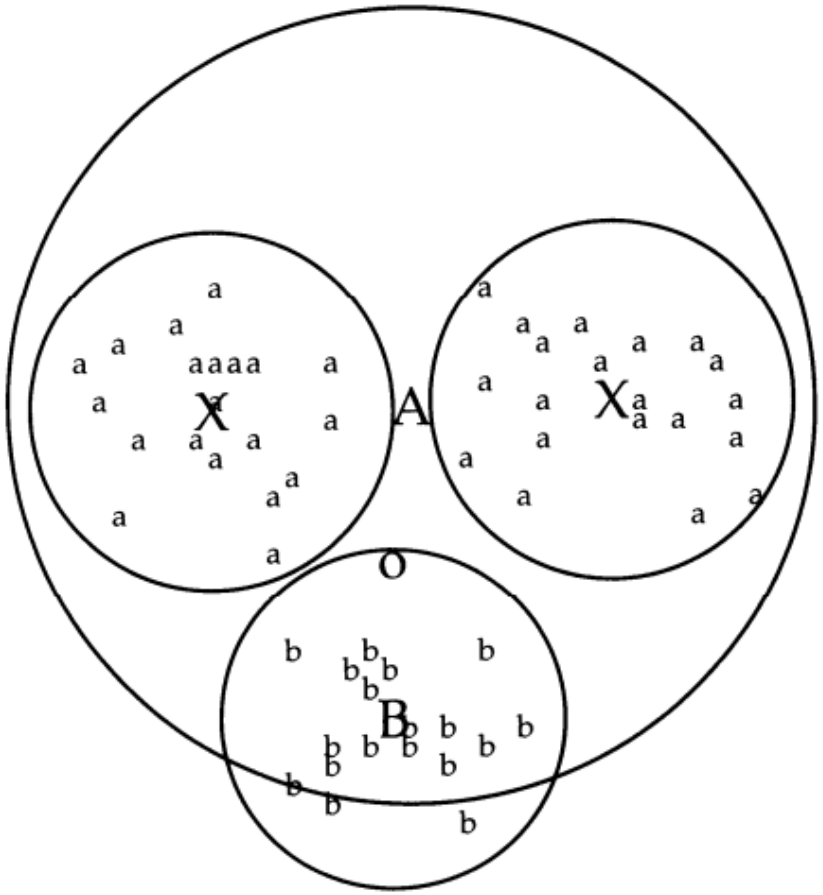
向量	词项权重					
	Chinese	Japan	Tokyo	Macao	Bejing	Shanghai
$\vec{d}_1$	0	0	0	0	1.0	0
$\vec{d}_2$	0	0	0	0	0	1.0
$\vec{d}_3$	0	0	0	1.0	0	0
$\vec{d}_4$	0	0.71	0.71	0	0	0
$\vec{d}_5$	0	0.71	0.71	0	0	0
$\vec{\mu}_c$	0	0	0	0.33	0.33	0.33
$\vec{\mu}_{\bar{c}}$	0	0.71	0.71	0	0	0

$$|\vec{\mu}(c)-\vec{d}_5| \approx 1.15$$

$$|\vec{\mu}(\bar{c})-\vec{d}_5| = 0.0$$

Rocchio分类方法的缺陷

- 为了遵循邻近性的要求，Rocchio分类中的每个类别一定要近似球形，并且它们之间具有相似的球半径。
- 多模态类别“a” 由两个不同簇(分别是以X为中心的两个小圆)组成。由于“O” 更接近“a”的中心A，因此，Rocchio分类会将其错分到“A”类



小结

- 算法步骤
  - (1)计算每个类的中心向量
  - (2)将每篇测试文档分到离它最近的那个中心向量

$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{d \in D_c} \vec{v}(d)$$

- 特性
  - Rocchio 分类方法类的边界由那些到两个类质心等距的点集组成(超平面)。
  - Rocchio 分类中的每个类别一定要近似球形，并且它们之间具有相似的球半径。
  - Rocchio算法的时间复杂度与NB方法在训练上具有相同的时间复杂度

内容

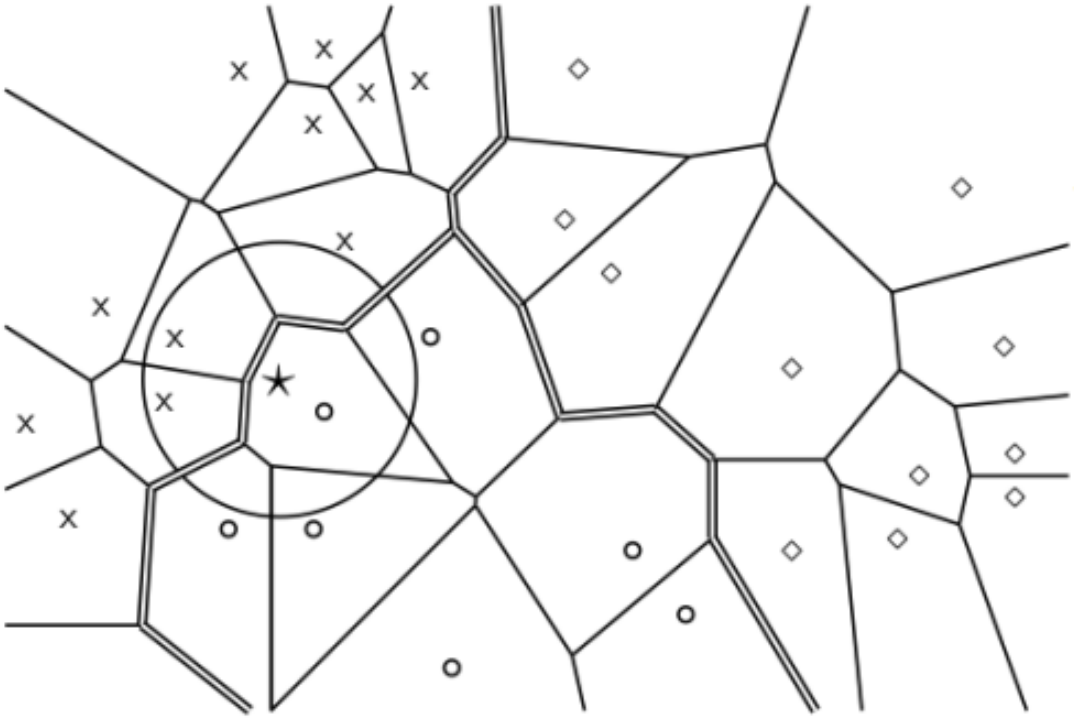
- 基于向量空间的分类方法
- Rocchio方法
- *k*NN(*k* 近邻)方法
- 线性分类器

*k*NN(k 近邻)方法

- *k*NN = *k* nearest neighbors， *k*近邻
- *k* = 1 情况下的*k*NN (最近邻): 将每篇测试文档分给训练集中离它最近的那篇文档所属的类别。
- 1NN 不很鲁棒—一篇文档可能会分错类或者这篇文档本身就很反常
- *k* > 1 情况下的*k*NN: 将每篇测试文档分到训练集中离它最近的*k*篇文档所属类别中最多的那个类别
- *k*NN 的基本依据
  - 根据邻近假设，一篇测试文档*d* 将和其邻域中的训练文档应该具有相同的类别。

1NN分类器

- 1NN分类器的判别边界是Voronoi剖分(Voronoi tessellation)形成的多个线段的连接。Voronoi剖分会将整个平面分成|D|个凸多边形，每个多边形都仅包含其对应的文档，而每个凸多边形是在二维空间中通过直线围成的凸区域。



1NN分类中的Voronoi剖分及分类边界(双线表示)。3个类别分别采用x、圆圈和菱形表示

*k*NN：多边形扩展到高维空间就是多面体

*k*NN思路的改进

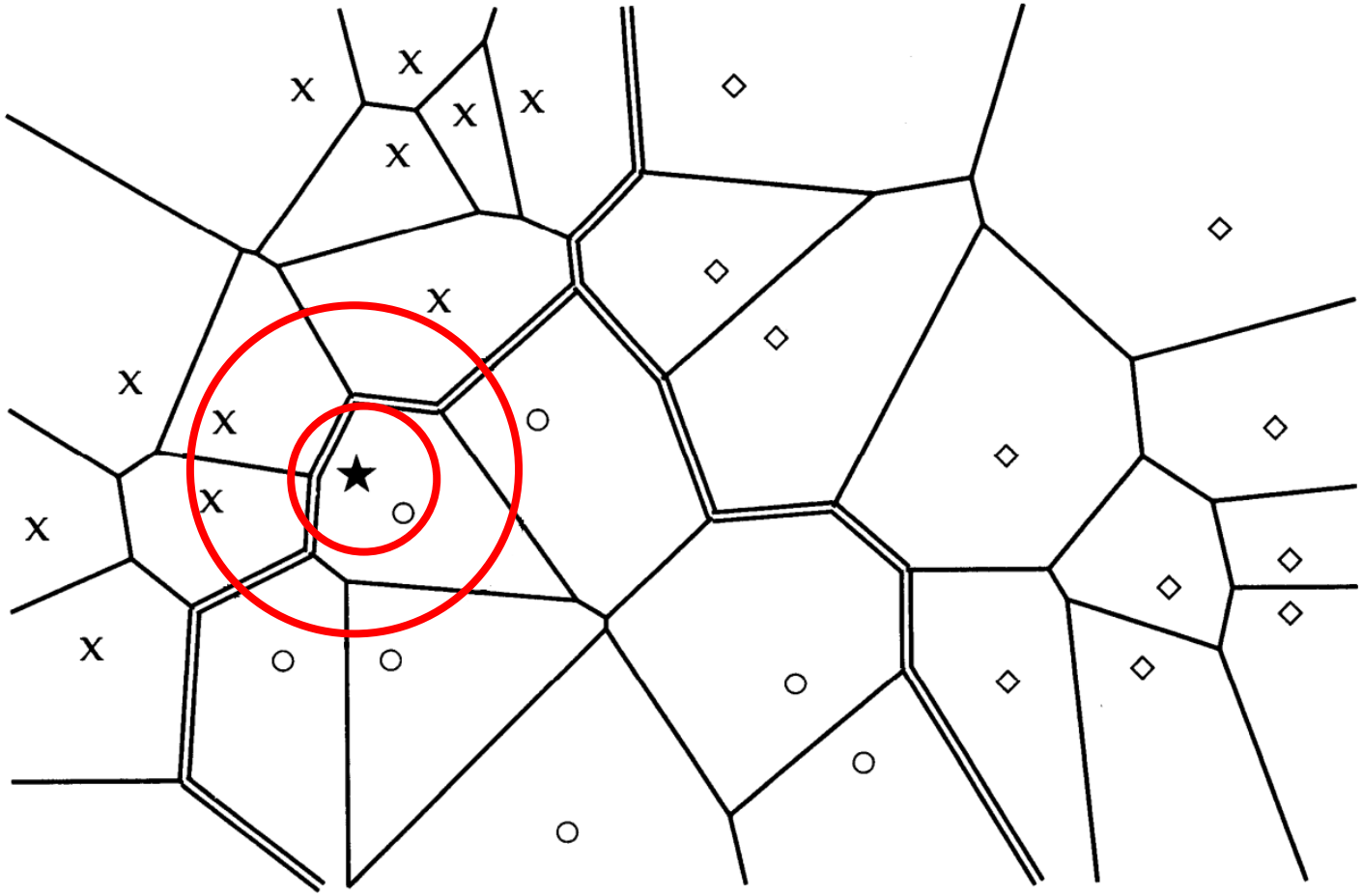
- **改进1**：*k*NN的概率型版本：将属于类别*c*的概率估计为*k*个近邻中属于类别*c*的文档比例
  - $P(c|d)$ = *d*的最近的*k*个邻居中属于*c*类的比例
  - 将*d*分到具有最高概率$P(c|d)$的类别*c*中
- **改进2**：也可以将*k*个近邻基于其余弦相似度进行加权。文档*d*属于某个类别*c*的得分计算如下

$$\text{score}(c,d) = \sum_{d' \in S_k} I_c(d') \cos(\vec{v}(d'), \vec{v}(d))$$

  - 其中，S_k表示的是文档*d*'的*k*个近邻文档组成的集合
  - 如果*d*'属于类别*c*则$I_c(d')=1$，否则$I_c(d')=0$ 。
  - 最后将得分最高的类别赋予文档*d*。

kNN示例1

- 对于★ 对应的文档，在1NN和 3NN下，分别应该属于哪个类？



*k*NN 算法

- 对于$k \in N$的一般*k*NN分类来说，考虑*k*个最近邻的区域的方法同前面一样。这里会再次得到一个凸多边形，整个空间也会划分为多个凸多边形，每个凸多边形中的*k*个近邻组成的集合是不变的

TRAIN-KNN($\mathbb{C}$, $\mathbb{D}$)
1 $\mathbb{D}' \leftarrow$ PREPROCESS($\mathbb{D}$)
2 $k \leftarrow$ SELECT-K($\mathbb{C}$, $\mathbb{D}'$)
3 **return** $\mathbb{D}'$, k

训练过程只是确定*k*值和对文档的预处理。如果预先选择*k*值并且不进行预处理，那么不需要任何训练过程。

*k*NN 算法的流程：
*k*NN 的训练(包括预处理)和分类过程。

APPLY-KNN($\mathbb{C}$, $\mathbb{D}'$, k, d)
1 $S_k \leftarrow$ COMPUTENEARESTNEIGHBORS($\mathbb{D}'$, k, d)
2 **for each** $c_j \in \mathbb{C}$
3 **do** $p_j \leftarrow |S_k \cap c_j|/k$
4 **return** $\arg\max_j p_j$

S_k表示的是文档d 的k个近邻文档组成的集合
p_j 是概率$P(c_j|S_k) = P(c_j|d)$的估计值，
c_j 表示的是类别c_j中的所有文档

kNN示例2

- 对于★对应的文档，在下列分类器下，分别应该属于哪个类？

  - (i) 1-NN
  - (ii) 3-NN
  - (iii) 7-NN
  - (iv) 15-NN
  - (v) Rocchio(绿色质心)

小结： *k*NN (*k*近邻)方法

- **思路**：将每篇测试文档分到训练集中离它最近的*k*篇文档所属类别中最多的那个类别
- 基本依据：根据邻近假设，一篇测试文档*d*应该和其邻域中的训练文档具有相同的类别。
  - 当训练集非常大的时候，*k*NN分类的精度很高
  - 如果训练集很小，*k*NN可能效果很差。

内容

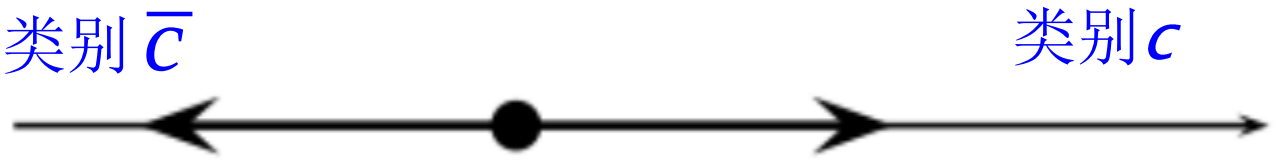
- 基于向量空间的分类方法
- Rocchio方法
- *k*NN(*k* 近邻)方法
- 线性分类器

线性分类器

- 定义
  - 计算特征值的一个线性加权和 $\sum_i w_i x_i$
  - 决策规则：$\sum_i w_i x_i > b$，b是一个参数
- 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
- 分类面
  - 基于训练集来寻找该分类面
  - 寻找分类面的方法：Rocchio, Naive Bayes – 将解释为什么两种方法是二元分类器

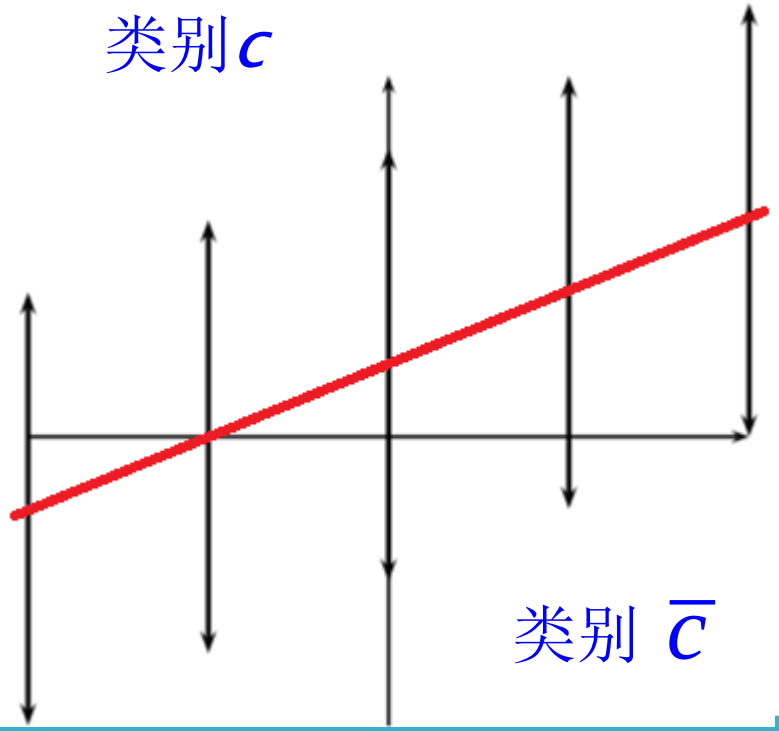
一维下的线性分类器

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



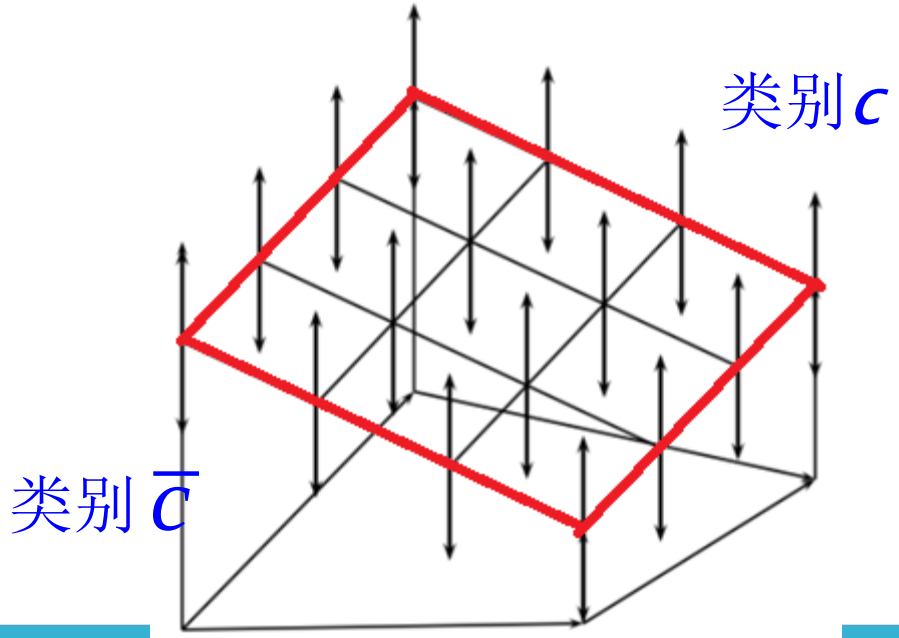
二维平面下的线性分类器

- 二维下的分类器是方程 $w_1 d_1 + w_2 d_2 = b$对应的直线
- 那些满足$w_1 d_1 + w_2 d_2 \geq b$ 的点(d_1, d_2)属于类别c
- 那些满足$w_1 d_1 + w_2 d_2 < b$ 的点(d_1, d_2)属于类别 $\bar{c}$



三维空间下的线性分类器

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



Two-class Rocchio as a linear classifier

- Line or hyperplane defined by:

$$\sum_{i=1}^{M} w_i d_i = b$$

- For Rocchio, set:

$$\vec{w} = \vec{\mu}(c_1) - \vec{\mu}(c_2)$$

$$b = 0.5 \times (|\vec{\mu}(c_1)|^2 - |\vec{\mu}(c_2)|^2)$$

Naive Bayes is a linear classifier

- Two-class Naive Bayes. We compute:

$$\log \frac{P(c \mid d)}{P(\overline{c} \mid d)} = \log \frac{P(c)}{P(\overline{c})} + \sum_{t \in d} \frac{P(w \mid c)}{P(w \mid \overline{c})}$$

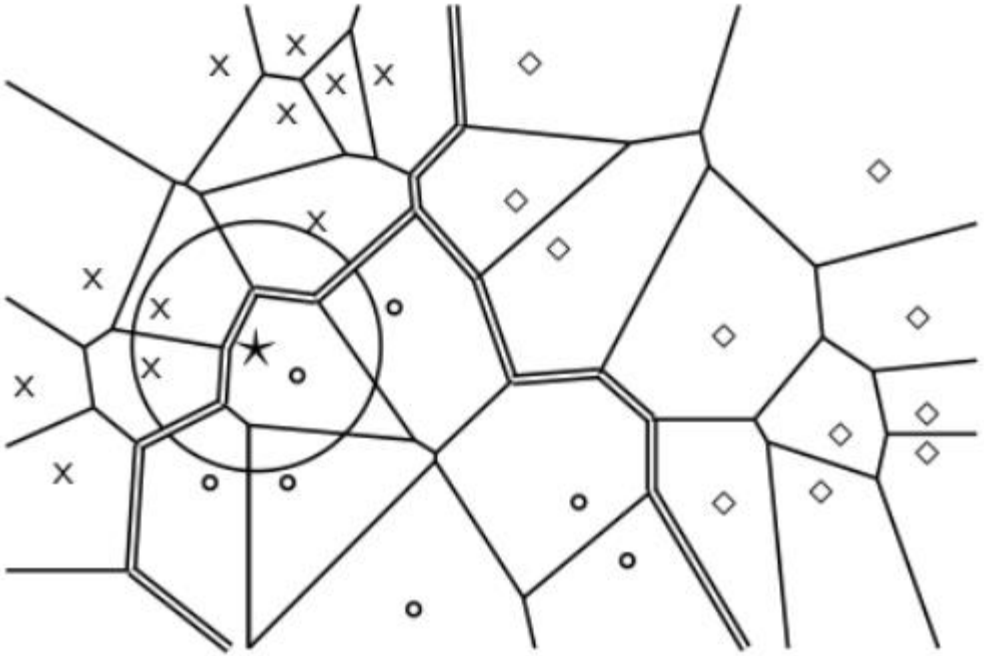
- Decide class c if the odds is greater than 1, i.e., if the log odds is greater than 0.
- So decision boundary is hyperplane:

$$\alpha + \sum_{t \in V} \beta_w \times n_w = 0 \quad \text{where } \alpha = \log \frac{P(c)}{P(\overline{c})}$$

$$\beta_w = \log \frac{P(w \mid c)}{P(w \mid \overline{c})} \qquad n_w = \#\text{of occurrences of } w \text{ in } d$$

*k*NN不是线性分类器

- *k*NN分类决策取决于*k*个邻居类中的多数类
- 类别之间的分类面是分段线性的(单个线段是线性的)
- 但是一般来说，很难表示成如下的 线性分类器



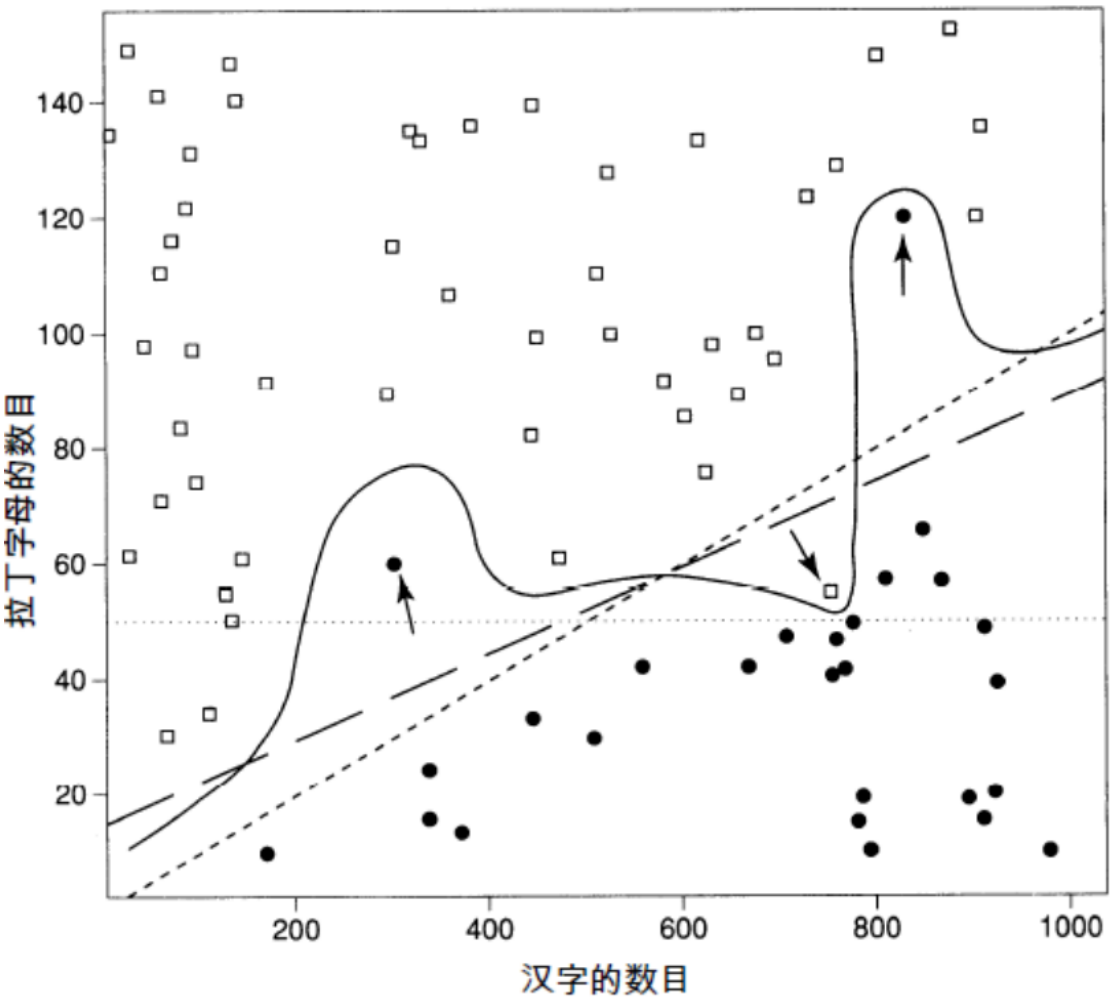
$$\sum_{i=1}^{M} w_i d_i = b$$

线性分类器:讨论

- 很多常用的文本分类器都是线性分类器：朴素贝叶斯、Rocchio、logistic回归、线性SVM等等
- 不同的方法在测试文档分类性能时存在巨大差异(分类面的选择不同)
- 能否通过更强大的非线性分类器来获得更好的分类性能?
  - 一般情况下不能，给定数量的训练集可能足以估计一个线性分类面，但是不足以估计一个更复杂的非线性分类面

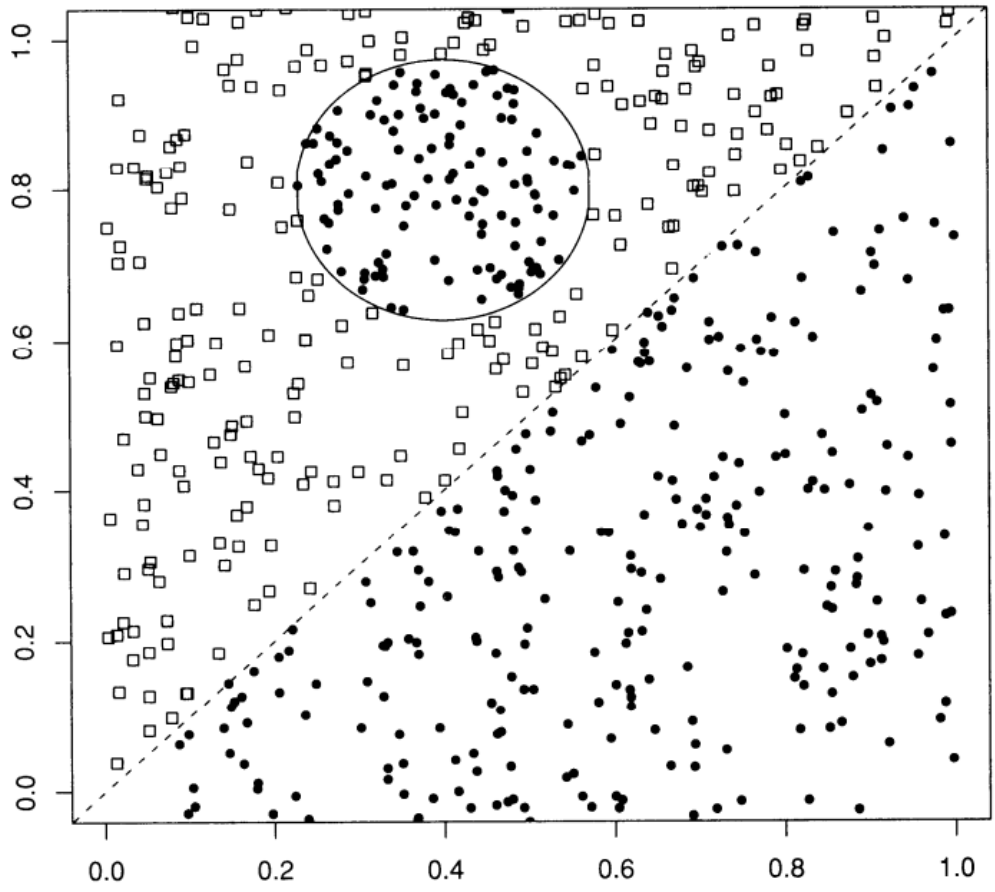
线性分类器训练困难的原因之一：噪音文档

一个带噪音的线性问题。在这个假想的网页分类下，仅包含中文的网页用实心圆表示，而中英文混合网页用小方块表示。除了3篇噪音文档(用箭头标记)外，这两个类可以被一个线性类别边界(用短破折号虚线表示)分开



非线性的分类问题 nonlinear classifiers

- 如果一个问题是非线性的，即它的类别边界不能通过线性超平面来近似，那么此时使用非线性分类器的分类结果往往会好于使用线性分类器；但是如果一个问题是线性的，那么最好用简单的线性分类器。



多标签分类问题

- 单标签分类问题，也称single label problem
  - 类别之间互斥。每篇文档属于且仅属于某一个类
- 多标签分类问题，也称multilabel classification
  - 一篇文档可以属于0个、1个或更多个类
  - 对于多标签分类问题(比如A、 B、 C三类)，可以组合为多个二类线性分类器(A vs. BC、B vs. AC、C vs. AB)