现代信息检索
Modern Information Retrieval

第11讲 文本分类 1

改编自斯坦福大学Information Retrieval & Web Search课件
https://web.stanford.edu/class/cs276/handouts/lecture12-textcat.pptx

常设查询（Standing Queries）

- 从检索到文本分类:
  - 假设某用户有一个经常关注的信息需求:
    - 北京新增确诊
  - 用户会经常输入这个查询来寻找关于这个主题的新内容
  - 关注于浏览新内容
    - 此时排序问题变成了一个分类问题（相关 vs. 不相关）
- 此类查询称为“**常设查询**”
  - 长久以来被“信息专业人员(information professionals)”所使用
  - Google Alerts 是一种常设查询的现代大规模实现
- 常设查询是一种（人工的）文本分类器

From: Google Alerts
Subject: **Google Alert - stanford -neuro-linguistic nlp OR "Natural Language Processing" OR parser OR tagger OR ner OR "named entity" OR segmenter OR classifier OR dependencies OR "core nlp" OR corenlp OR phrasal**
Date: May 7, 2012 8:54:53 PM PDT
To: Christopher Manning

Web

3 new results for **stanford -neuro-linguistic nlp OR "Natural Language Processing" OR parser OR tagger OR ner OR "named entity" OR segmenter OR classifier OR dependencies OR "core nlp" OR corenlp OR phrasal**

Twitter / **Stanford NLP** Group: @Robertoross If you only n ...
@Robertoross If you only need tokenization, java -mx2m edu.**stanford.nlp**. process.PTBTokenizer file.txt runs in 2MB on a whole file for me.... 9:41 PM Apr 28th ...
twitter.com/stanfordnlp/status/196459102770171905

[Java] LexicalizedParser lp = LexicalizedParser.loadModel("edu ...
loadModel("edu/**stanford**/**nlp**/models/lexparser/englishPCFG.ser.gz");. String[] sent = { "This", "is", "an", "easy", "sentence", "." };. Tree parse = lp.apply(Arrays.
pastebin.com/az14R9nd

More Problems with Statistical **NLP** || kuro5hin.org
Tags: nlp, ai, coursera, **stanford**, **nlp**-class, cky, nltk, reinventing the wheel, ... Programming Assignment 6 for **Stanford's nlp**-class is to implement a CKY parser .
www.kuro5hin.org/story/2012/5/5/11011/68221

Tip: Use quotes ("like this") around a set of words in your query to match them exactly. Learn more.

Delete this alert.
Create another alert.
Manage your alerts.

垃圾邮件过滤：另一个文本分类应用

From: "" <takworlld@hotmail.com>
Subject: real estate is the only way... gem oalvgkay

Anyone can buy real estate with no money down

Stop paying rent TODAY !

There is no need to spend hundreds or even thousands for similar courses

I am 22 years old and I have already purchased 6 properties using the methods outlined in this truly INCREDIBLE ebook.

Change your life NOW !

==========

Click Below to order:
http://www.wholesaledaily.com/sales/nmd.htm

==========

Categorization/Classification(分类)

- 给定:
  - 文档 d 的表示
    - 问题: 怎样表示文本文档?
    - 通常使用一种高维空间里的表示 - 例如词袋模型
  - 一个预定义的（通常是固定的）类别集合:

    $C = \{c_1, c_2, \cdots, c_J\}$
- 预测:
  - 文档所属类别 d: $\gamma(d) \in C$, 其中 $\gamma(d)$ 是一个 classification function (分类函数)
  - 我们需要构建分类函数(“classifiers/分类器”).

分类方法（1）

- 人工分类
  - 早期 Yahoo! Directory 基于人工分类
    - 网站分类目录
  - Looksmart, about.com, ODP, PubMed
  - 专家分类一般都是准确的
  - 当数据规模不大、标注者人数较少时，分类一致
  - 当数据规模变大，人工分类困难且代价昂贵
    - 这意味着我们需要自动分类方法来解决大规模数据分类问题

分类方法（2）

- 人工编写的基于规则的分类器
  - 新闻机构，情报机构等使用的一个技术
  - 广泛部署于政府和企业
  - 供应商提供“ IDE”来编写此类规则
  - 商业系统具有复杂的查询语言
  - 如果领域专家花时间精心完善规则，则准确性会很高
  - 建立和维护这些规则非常昂贵

一个 Verity 主题的例子
关于 art 类别的复杂分类规则

```
comment line          # Beginning of art topic definition
top-level topic       art ACCRUE
                          /author = "fsmith"
topic definition modifiers
                          /date   = "30-Dec-01"
                          /annotation = "Topic created
                                         by fsmith"
subtopic topic        * 0.70 performing-arts ACCRUE
  evidence topic      ** 0.50 WORD
  topic definition modifier
                          /wordtext = ballet
  evidence topic      ** 0.50 STEM
  topic definition modifier
                          /wordtext = dance
  evidence topic      ** 0.50 WORD
  topic definition modifier
                          /wordtext = opera
  evidence topic      ** 0.30 WORD
  topic definition modifier
                          /wordtext = symphony
subtopic              * 0.70 visual-arts ACCRUE
                      ** 0.50 WORD
                          /wordtext = painting
                      ** 0.50 WORD
                          /wordtext = sculpture
subtopic              * 0.70 film ACCRUE
                      ** 0.50 STEM
                          /wordtext = film
subtopic              ** 0.50 motion-picture PHRASE
                      *** 1.00 WORD
                          /wordtext = motion
                      *** 1.00 WORD
                          /wordtext = picture
                      ** 0.50 STEM
                          /wordtext = movie
subtopic              * 0.50 video ACCRUE
                      ** 0.50 STEM
                          /wordtext = video
                      ** 0.50 STEM
                          /wordtext = vcr
                      # End of art topic
```

- 注意：
  - 维护问题（作者等）
  - 关键字的手工加权

[Verity在2005年被Autonomy收购，在2011年被HP收购 - 使用起来很糟糕；现在已不再应用]

分类方法（3）:有监督学习

- 给定:
  - 文档 d
  - 一个固定的类别集合:

    $C = \{c_1, c_2, \cdots, c_J\}$
  - 一个 训练集 D，其中每个文档均有属于C的类别标签
- 决定:
  - 一种使我们能够学习分类器 γ 的学习方法或算法
  - 对于测试文档d，我们将其分配给类

    $\gamma(d) \in C$

分类方法（3）

- 有监督学习
  - Naive Bayes/朴素贝叶斯（simple, common）
  - k-Nearest Neighbors/K近邻（simple, powerful）
  - Support-vector machines/支持向量机（newer, generally more powerful）
  - Decision trees/决策树 → random forests/随机森林 → gradient-boosted decision trees/梯度增强决策树（例如, xgboost）
  - … 以及各种其它方法
  - 没有免费午餐：需要人工分类标记的训练数据
  - 但是数据标记的工作可以由非专家完成
- 许多商业系统结合使用多种方法

特征

- 监督学习分类器可以使用各种特征
  - URL，email 地址，标点符号，大写字母，词典，网络特征
- 在最简单的词袋模型文档表示中
  - 我们**仅**使用词项特征
  - 我们使用文本中的**所有**词项（而不是子集）

词袋表示

$$\gamma\left(\text{I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun… It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet.}\right)=c$$

词袋表示

为什么要做特征选择？

- 文本语料具有大量的词项/特征
  - 10,000 – 1,000,000 个甚至更多的词项
- 特征选择可以使得某些分类器可用
  - 有些分类器无法处理过多的特征
- 减少训练时间
  - 某些方法的训练时间是特征数量的二次方或更多
- 使运行时模型更小，更快
- 可以提高模型泛化能力
  - 消除噪声特征
  - 避免过拟合

特征选择：词频

- 最简单的特征选择方法：
  - 仅使用最常见词项
  - 没有特别的（理论）依据
  - 但是很好理解：
    - 这些词的概率可以被很好地估计（因为词频高），并且最常被用作相关性的证据
  - 在实际应用中，词频特征选择往往能达到一些更高的方法的90%的性能
- 更聪明的特征选择方法：
  - 卡方（chi-square）等

朴素贝叶斯(Naïve Bayes)分类器

- 朴素贝叶斯是一个概率分类器
- 文档 d 属于类别 c 的概率计算如下（多项式模型）：

$$P(c \mid d) = P(d \mid c)P(c) / P(d) \propto P(d \mid c)P(c) = P(c) \prod_{1 \le k \le n_d} P(t_k \mid c)$$

  - n_d 是文档的长度(词条的个数)
  - $P(t_k \mid c)$ 是在类别c中文档抽样得到词项t_k的概率，即类别c文档的一元语言模型!
  - $P(t_k \mid c)$ 度量的是当c是正确类别时t_k 的贡献
  - $P(c)$ 是类别c的先验概率
  - 如果文档的词项无法提供属于哪个类别的信息，那么我们直接选择$P(c)$最高的那个类别

16

具有最大后验概率的类别

- 朴素贝叶斯分类的目标是寻找“最佳”的类别
- 最佳类别是指具有最大后验概率（**maximum a posteriori - MAP**）的类别 c_{map}：

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} \hat{P}(c|d) = \arg\max_{c \in \mathbb{C}} \hat{P}(c) \prod_{1 \le k \le n_d} \hat{P}(t_k|c)$$

17

对数计算

- 很多小概率的乘积会导致浮点数下溢出
- 由于 $\log(xy) = \log(x) + \log(y)$，可以通过取对数将原来的乘积计算变成求和计算
- 由于log是单调函数，因此得分最高的类别不会发生改变
- 因此，实际中常常使用的是：

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} [\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)]$$

18

朴素贝叶斯分类器

- 分类规则：

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} [\log \hat{P}(c) + \sum_{1 \le k \le n_d} \log \hat{P}(t_k|c)]$$

- 简单说明：
  - 每个条件参数 $\hat{P}(t_k|c)$ 是反映 t_k 对c的贡献高低的一个权重
  - 先验概率 $\hat{P}(c)$ 是反映类别c的相对频率的一个权重
  - 因此，所有权重的求和反映的是文档属于类别的可能性
  - 选择最具可能性的类别

19

参数估计：极大似然估计

- 如何从训练数据中估计 $\hat{P}(c)$ $\hat{P}(t_k|c)$ ？
- 先验：

$$\hat{P}(c) = \frac{N_c}{N}$$

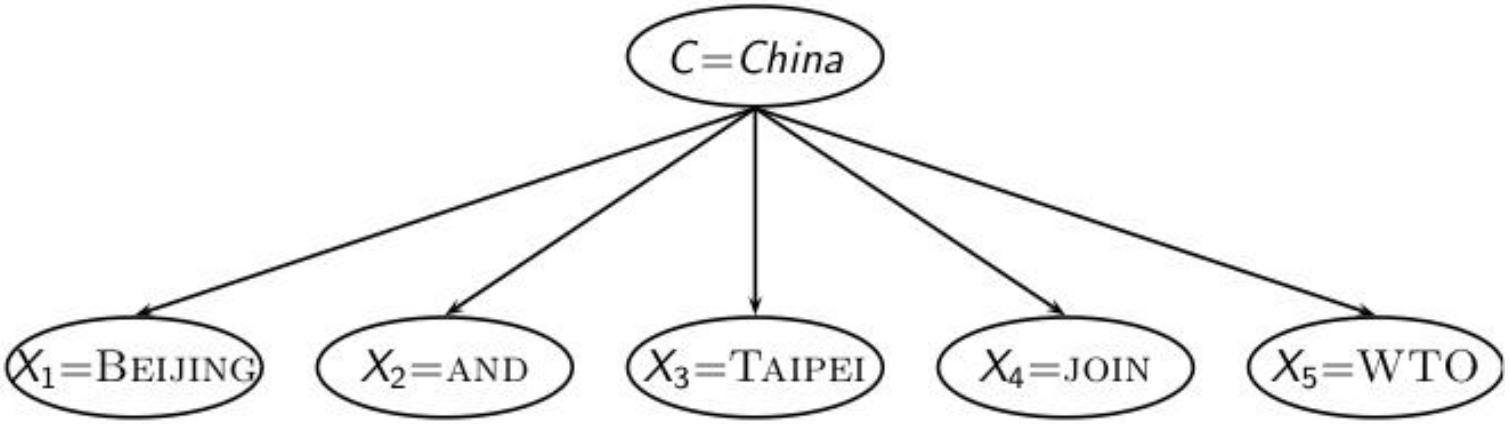
- N_c ： 类c中的文档数目； N: 所有文档的总数
- 条件概率：

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

- T_{ct} 是训练集中类别c中的词条t的个数（多次出现要计算多次）
- 给定如下的 **位置独立性假设**(positional independence assumption)：

$$\hat{P}(t_{k_1}|c) = \hat{P}(t_{k_2}|c)$$

MLE估计中的问题：零概率问题



$$P(China|d) \propto P(China) \cdot P(\text{BEIJING}|China) \cdot P(\text{AND}|China) \cdot P(\text{TAIPEI}|China) \cdot P(\text{JOIN}|China) \cdot P(\text{WTO}|China)$$

- 如果 WTO 在训练集中没有出现在类别 China中:

$$\hat{P}(\text{WTO}|China) = \frac{T_{China,\text{WTO}}}{\sum_{t' \in V} T_{China,t'}} = \frac{0}{\sum_{t' \in V} T_{China,t'}} = 0$$

21

MLE估计中的问题：零概率问题（续）

- 如果 WTO 在训练集中没有出现在类别 China中，那么就会有如下的零概率估计：

$$\hat{P}(\mathrm{WTO}|China) = \frac{T_{China,\mathrm{WTO}}}{\sum_{t' \in V} T_{China,t'}} = 0$$

- 那么，对于任意包含WTO的文档d，$P(\mathrm{China}|d) = 0$。
- 一旦发生零概率，将无法判断类别

22

避免零概率: 加一平滑

- 平滑前:

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

- 平滑后: 对每个量都加上1

$$\hat{P}(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V}(T_{ct'} + 1)} = \frac{T_{ct} + 1}{(\sum_{t' \in V} T_{ct'}) + B}$$

- B 是不同的词语个数 (这种情况下词汇表大小 $|V| = B$)

23

避免零概率: 加一平滑（续）

- 利用加1平滑从训练集中估计参数
- 对于新文档，对于每个类别，计算
  - （i）先验的对数值之和以及
  - （ii）词项条件概率的对数之和
- 将文档归于得分最高的那个类

24

朴素贝叶斯: 训练过程

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

1 $V \leftarrow$ EXTRACTVOCABULARY($\mathbb{D}$)
2 $N \leftarrow$ COUNTDOCS($\mathbb{D}$)
3 **for each** $c \in \mathbb{C}$
4 **do** $N_c \leftarrow$ COUNTDOCSINCLASS($\mathbb{D}, c$)
5 $prior[c] \leftarrow N_c/N$
6 $text_c \leftarrow$ CONCATENATETEXTOFALLDOCSINCLASS($\mathbb{D}, c$)
7 **for each** $t \in V$
8 **do** $T_{ct} \leftarrow$ COUNTTOKENSOFTERM($text_c, t$)
9 **for each** $t \in V$
10 **do** $condprob[t][c] \leftarrow \frac{T_{ct}+1}{\sum_{t'}(T_{ct'}+1)}$
11 **return** $V, prior, condprob$

25

朴素贝叶斯: 测试/应用/分类

$\textsc{ApplyMultinomialNB}(\mathbb{C}, V, prior, condprob, d)$

1 $W \leftarrow \textsc{ExtractTokensFromDoc}(V, d)$
2 **for each** $c \in \mathbb{C}$
3 **do** $score[c] \leftarrow \log prior[c]$
4 **for each** $t \in W$
5 **do** $score[c] += \log condprob[t][c]$
6 **return** $\arg\max_{c \in \mathbb{C}} score[c]$

26

课堂练习

	docID	words in document	in $c = China$?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

- 估计朴素贝叶斯分类器的参数
- 对测试文档进行分类

例子: 参数估计

先验 $\hat{P}(c) = 3/4$ 及 $\hat{P}(\overline{c}) = 1/4$ ，而条件概率如下：

$$\begin{aligned}
\hat{P}(\text{CHINESE}|c) &= (5+1)/(8+6) = 6/14 = 3/7 \\
\hat{P}(\text{TOKYO}|c) = \hat{P}(\text{JAPAN}|c) &= (0+1)/(8+6) = 1/14 \\
\hat{P}(\text{CHINESE}|\overline{c}) &= (1+1)/(3+6) = 2/9 \\
\hat{P}(\text{TOKYO}|\overline{c}) = \hat{P}(\text{JAPAN}|\overline{c}) &= (1+1)/(3+6) = 2/9
\end{aligned}$$

上述计算中的分母分别是（8 + 6）和（3 + 6），这是因为 $text_c$ 和 $text_{\overline{c}}$ （分别代表两类文档集的大小）的大小分别是8和3，而词汇表大小是6。

28

例子: 分类

$$\hat{P}(c|d_5) \propto 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003$$
$$\hat{P}(\overline{c}|d_5) \propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001$$

因此，分类器将测试文档分到$c = China$类，这是因为d_5中起正向作用的CHINESE出现3次的权重高于起反向作用的 JAPAN和TOKYO的权重之和。

朴素贝叶斯的时间复杂度分析

mode	time complexity
training	$\Theta(\lvert\mathbb{D}\rvert L_{\text{ave}} + \lvert\mathbb{C}\rvert\lvert V\rvert)$
testing	$\Theta(L_{\text{a}} + \lvert\mathbb{C}\rvert M_{\text{a}}) = \Theta(\lvert\mathbb{C}\rvert M_{\text{a}})$

- L_{ave}：训练文档的平均长度，L_{a}：测试文档的平均长度，M_{a}：测试文档中不同的词项个数 $\mathbb{D}$: 训练文档，V：词汇表，类别: $\mathbb{C}$: 类
- $\Theta(\lvert\mathbb{D}\rvert L_{\text{ave}})$ 是计算所有统计数字的时间
- $\Theta(\lvert\mathbb{C}\rvert\lvert V\rvert)$ 是从上述数字计算参数的时间
- 通常来说：$\lvert\mathbb{C}\rvert\lvert V\rvert < \lvert\mathbb{D}\rvert L_{\text{ave}}$
- 测试时间也是线性的（相对于测试文档的长度而言）
- 因此：**朴素贝叶斯 对于训练集的大小和测试文档的大小而言是线性的。这在某种意义上是最优的。**

30

SpamAssassin

- 朴素贝叶斯已成功应用于垃圾邮件过滤
  - 广泛应用：Apache SpamAssassin
  - 除词项特征外，还使用了很多其它特征：
  - But many features beyond words:
    - 黑名单，等
    - 人工定义的文本模式

SpamAssassin 使用的特征:

- 朴素贝叶斯模型计算的概率
- 关键字（mentions）: Generic Viagra
- 正则表达式: millions of (dollar) ((dollar) NN,NNN,NNN.NN)
- 短语: impress ... girl
- 短语: ‘Prestigious Non-Accredited Universities’
- 发信人: starts with many numbers
- 标题全大写
- HTML：文本 / 图片 比例（太低则很可能是垃圾邮件）
- Received行看上去是假的（RCVD line looks faked）
- http://spamassassin.apache.org/tests_3_3_x.html

朴素贝叶斯 并不朴素

- 朴素贝叶斯在多次竞赛中胜出（比如 KDD-CUP 97）
- 相对于其他很多更复杂的学习方法，朴素贝叶斯对不相关特征更具鲁棒性
- 相对于其他很多更复杂的学习方法，朴素贝叶斯对概念漂移(concept drift)更鲁棒(概念漂移是指类别的定义随时间变化)
- 当有很多同等重要的特征时，该方法优于决策树类方法
- 一个很好的文本分类基准方法（当然，不是最优的方法）
- 如果满足独立性假设，那么朴素贝叶斯是最优的（文本当中并不成立，但是对某些领域可能成立）
- (训练和测试)速度非常快
- 存储开销少
  - NB分类是一种“线上”模型，测试样本生成概率需实时计算得到

分类结果的评价

- 评估必须在独立于训练数据的测试数据上完成
  - 交叉验证是一种常见做法，见第七讲
- 如果测试数据和训练数据有重合，那么分类器会拟合测试数据，从而在测试数据上获得较好的效果
  - 但这并不是一种正确的做法

分类结果的评价

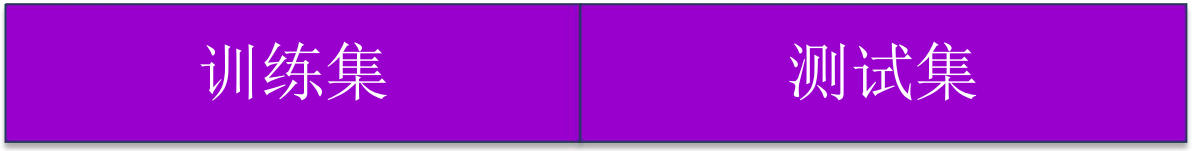
- 评价指标：正确率(Precision)，召回率(Recall)，F1，分类准确率
- 分类准确率(Classification accuracy)：r/n，其中 n 是所有测试文档的数量，r 是正确分类的测试文档数量

关于训练集和测试集

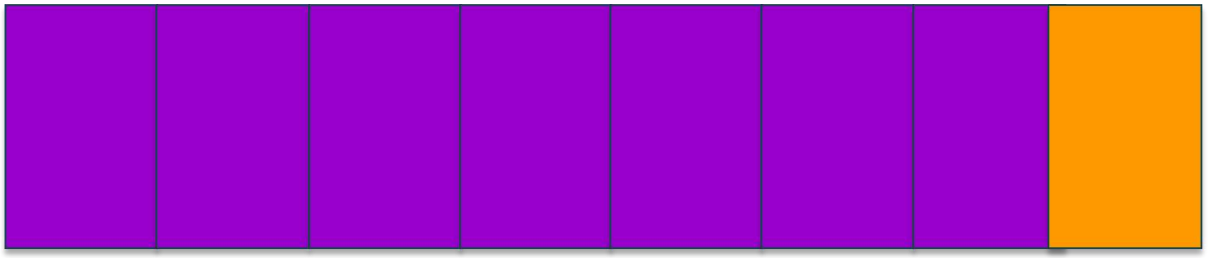
- 给定一个已标注好的数据集，将其中一部分划为训练集(training set)，另一部分划为测试集(test set)。在训练集上训练，训练得到的分类器用于测试集，计算测试集上的评价指标。
- 另一种做法：将上述整个数据集划分成k份，然后以其中k-1份为训练集训练出一个分类器，并用于另一份(测试集)，循环k次，将k次得到的评价指标进行平均。这种做法称为k交叉验证(k-cross validation)
- 有时，分类器的参数需要优化。此时，可以仍然将整个数据集划分成训练集和测试集，然后将训练集分成k份(其中k-1份作为训练，另一份称为验证集validation set，也叫开发集)，在训练集合上进行k交叉验证，得到最优参数。然后应用于测试集。

关于训练集和测试集

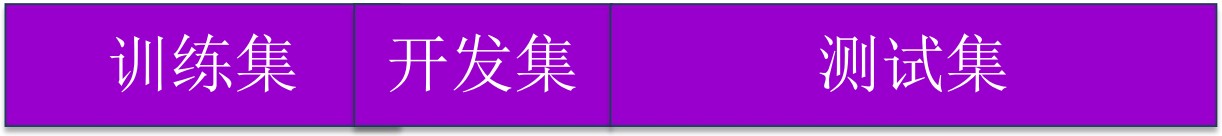
训练+测试

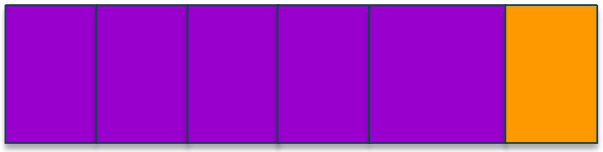


k交叉测试



参数确定





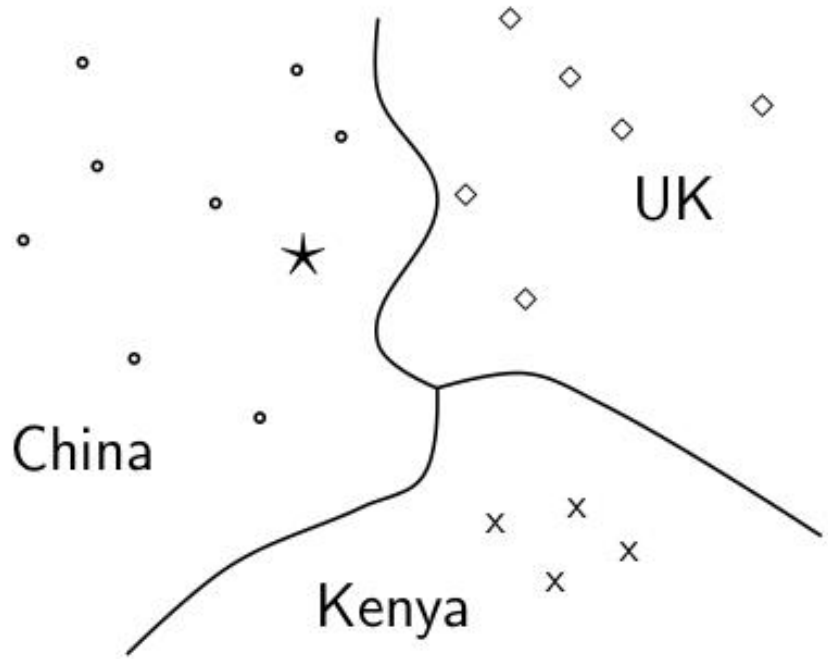
回顾：向量空间表示

- 每篇文档都表示一个向量，每一维对应一个词项
- 词项就是坐标轴
- 通常都高维：100,000多维
- 通常要将向量归一化到单位长度
- 如何在该空间下进行分类？

向量空间分类

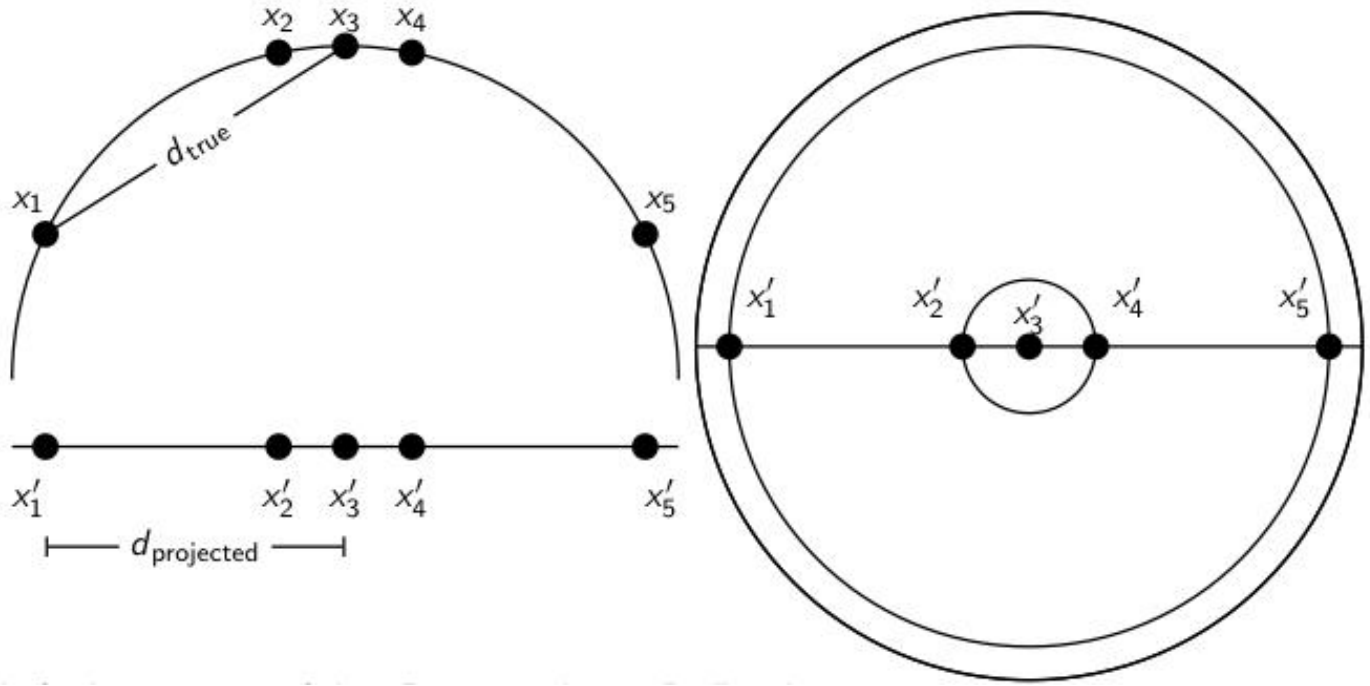
- 同前面一样，训练集包含一系列文档，每篇都标记着它的类别
- 在向量空间分类中，该集合对应着空间中一系列标记的点或向量。
- 假设 1: 同一类中的文档会构成一片连续区域（contiguous region）
- 假设2: 来自不同类别的文档没有交集
- 接下来我们定义直线、平面、超平面来将上述不同区域分开

向量空间中的类别



- 文档*到底是属于UK、China还是Kenya类？首先找到上述类别之间的分类面，然后确定文档所属类别，很显然按照图中分类面，文档应该属于China类
- 如何找到分类面并将文档判定给正确类别是本讲的重点。

题外话: 2D/3D 图形可能会起误导作用



左图：从二维空间的半圆映射到一维直线上。点x_1、x_2、x_3、x_4、x_5 的X轴坐标分别是−0.9、−0.2、0、0.2和0.9，距离$|x_2x_3| \approx 0.201$，和$|x_2{'}x_3{'}| = 0.2$只有0.5%的差异，但是当对较大的区域进行投影的话，比如$|x_1x_3| / |x_1{'}x_3{'}| = d_{\text{true}}/d_{\text{projected}} \approx 1.06/0.9 \approx 1.18$却会产生较大的差异（18%）。右图：相应的从三维的半球面到二维平面上的投影

利用Rocchio方法进行向量空间分类

- 相关反馈和文本分类的主要区别在于：
  - 在文本分类中，训练集作为输入的一部分事先给定
  - 在相关反馈中，训练集在交互中创建

Rocchio分类: 基本思想

- 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
- 将每篇测试文档分到离它最近的那个中心向量

中心向量的定义

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

其中 D_c 是所有属于类别 c 的文档，$\vec{v}(d)$ 是文档d的向量空间表示

Rocchio算法

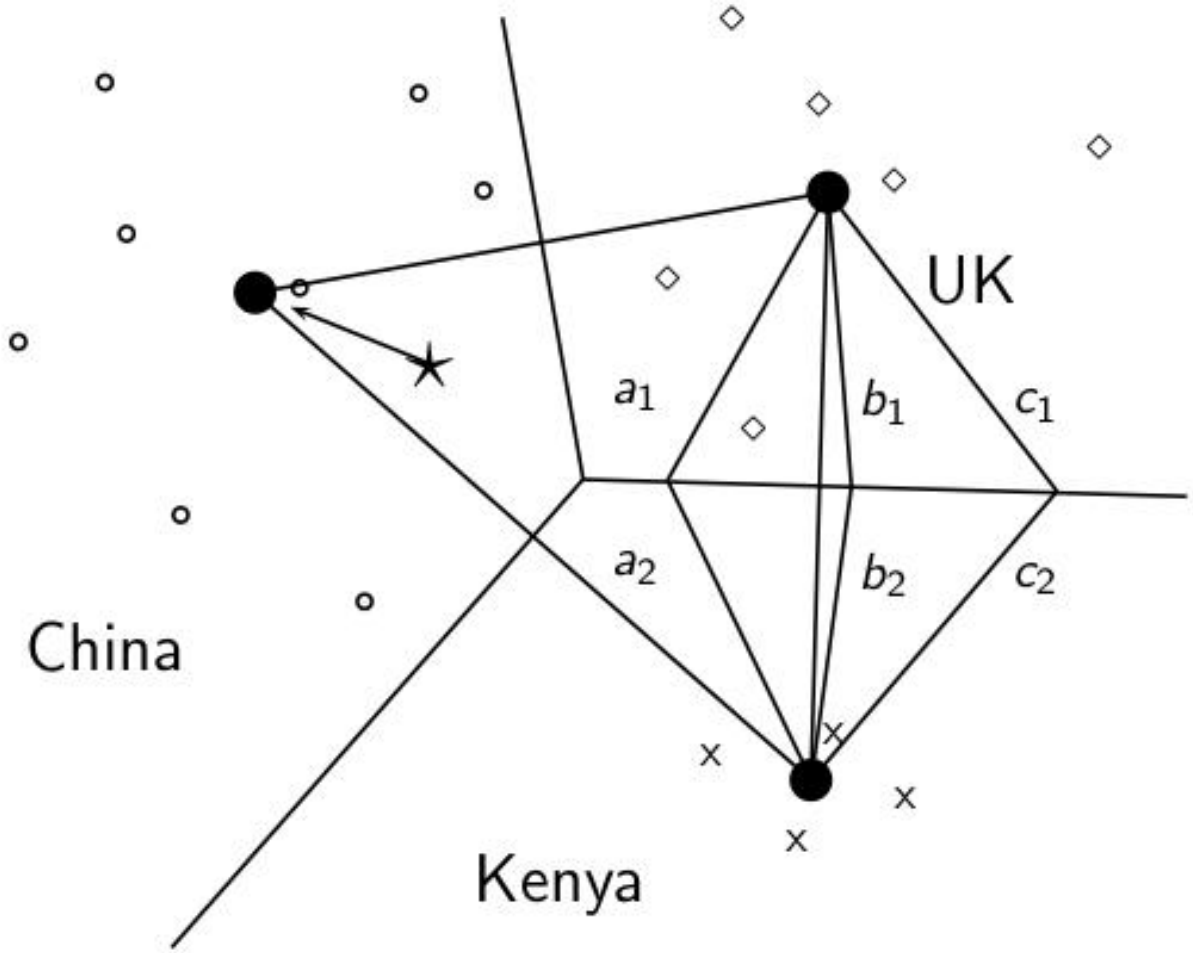
TRAINROCCHIO($\mathbb{C}, \mathbb{D}$)

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

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



Rocchio性质

- Rocchio简单地将每个类别表示成其中心向量
  - 中心向量可以看成类别的原型(prototype)
- 分类基于文档向量到原型的相似度或聚类来进行
- 并不保证分类结果与训练集一致，即得到分类器后，不能保证训练集中的文档能否正确分类

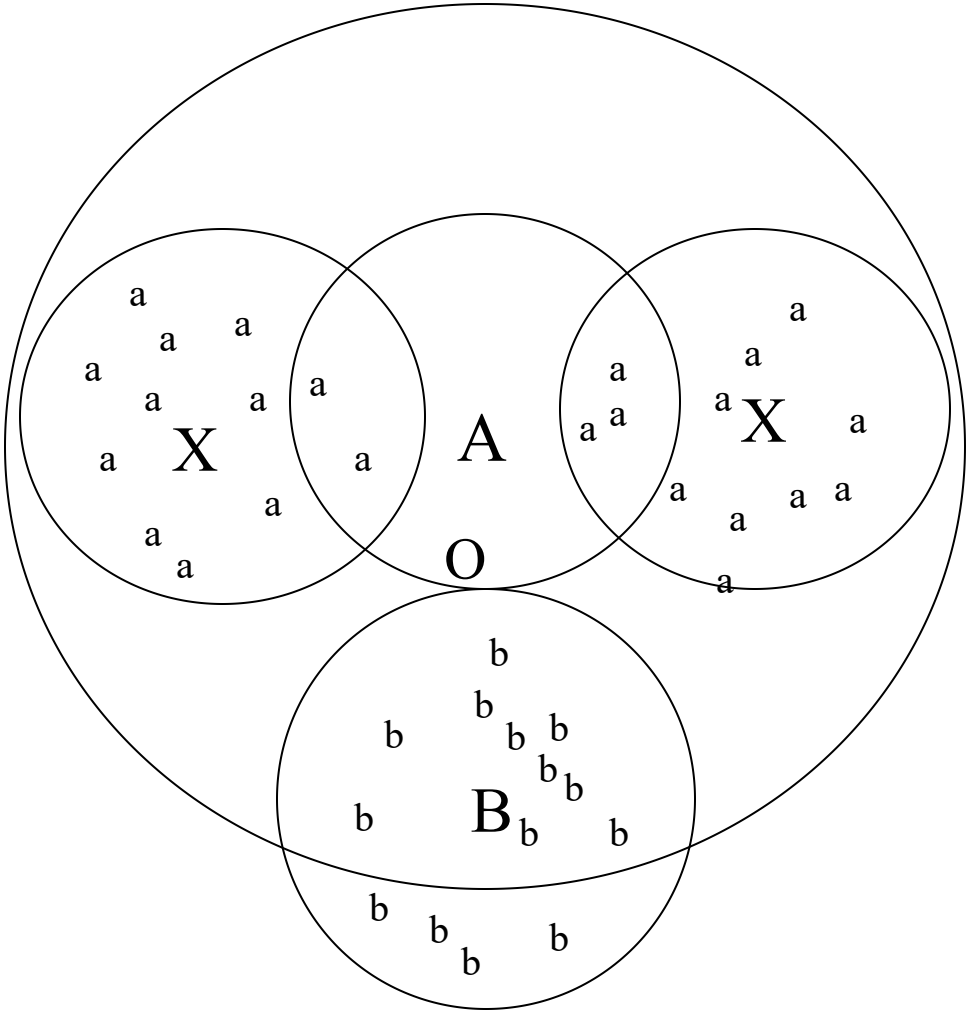
Rocchio算法的时间复杂度

mode	time complexity
training	$\Theta(\lvert\mathbb{D}\rvert L_{ave} + \lvert\mathbb{C}\rvert\lvert V\rvert) \approx \Theta(\lvert\mathbb{D}\rvert L_{ave})$
testing	$\Theta(L_a + \lvert\mathbb{C}\rvert M_a) \approx \Theta(\lvert\mathbb{C}\rvert M_a)$

Rocchio vs. 朴素贝叶斯

- 很多情况下，Rocchio的效果不如朴素贝叶斯
- 一个原因是，Rocchio算法不能正确处理非凸、多模式类别问题

Rocchio不能正确处理非凸、多模式类别问题



课堂练习: 对于左图的A/B分类问题，为什么Rocchio方法难以有效处理?

- A 是所有a的中心向量，B是所有b的中心向量
- 点o 离A更近
- 但是o更适合于b类
- A 是一个有两个原型多模式类别
- 但是，在Rocchio算法中，每个类别只有一个原型

kNN分类器

- kNN 是另外一种基于向量空间的分类方法
- 该方法非常简单，也容易实现
- 在大多数情况下，kNN的效果比朴素贝叶斯和Rocchio要好
- 如果你急切需要一种精度很高分类器并很快投入运行 . .
- . . . 如果你不是特别关注效率 . . .
- . . . 那么就使用kNN

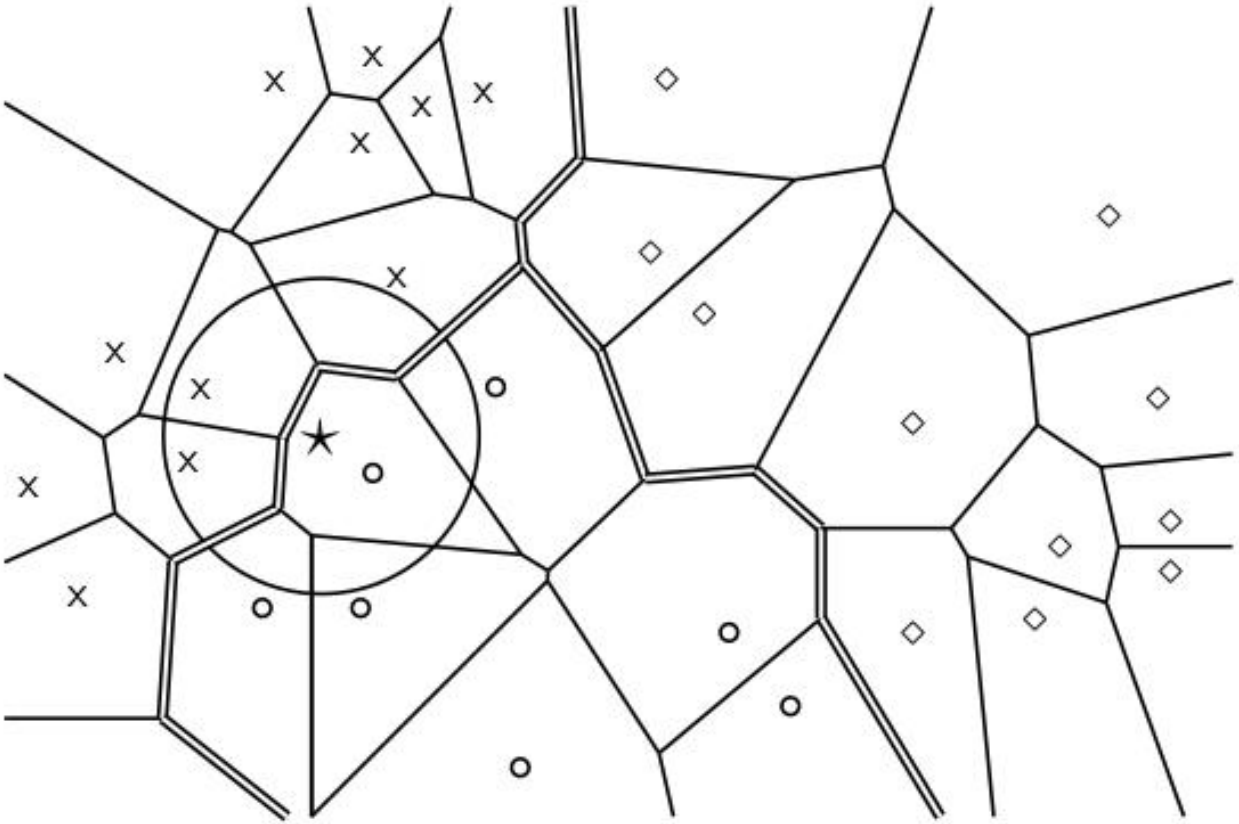
kNN分类

- kNN = k nearest neighbors，k近邻
- $k = 1$ 情况下的kNN (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kNN: 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
- kNN的基本原理: 邻近性假设
  - 我们期望一篇测试文档d与训练集中d周围邻域文档的类别标签一样。

概率型kNN

- kNN的概率型版本: $P(c|d)$ = d的最近的k个邻居中属于c类的比例
- 概率型kNN: 将d分到具有最高概率$P(c|d)$的类别c中

概率kNN



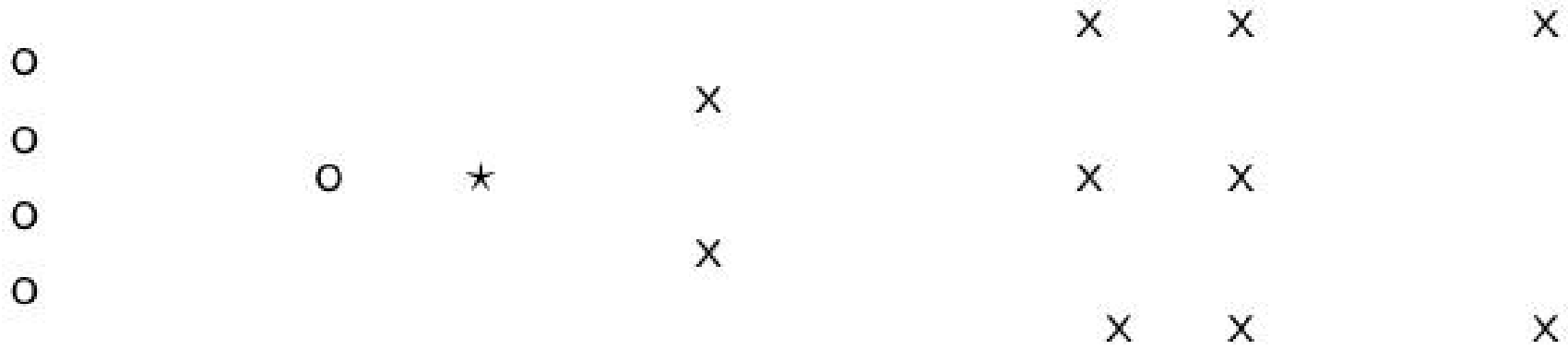
对于★ 对应的文档，在1NN和 3NN下，分别应该属于哪个类？

kNN 算法

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

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

课堂练习



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) 9-NN (iv) 15-NN (v) Rocchio?

kNN的时间复杂度

kNN (包括训练集的预处理)

训练 $\Theta(|\mathbb{D}|L_{ave})$

测试 $\Theta(L_a + |\mathbb{D}|M_{ave}M_a) = \Theta(|\mathbb{D}|M_{ave}M_a)$

- kNN 测试时间复杂度与训练集的大小成正比!
- 训练集越大，对测试文档分类的时间越长
- 在大训练集情况下，kNN效率较低

kNN: 讨论

- 不需要训练过程
  - 但是，文档的线性预处理过程和朴素贝叶斯的训练开销相当
  - 对于训练集来说我们一般都要进行预处理，因此现实当中kNN的训练时间是线性的。
- 当训练集非常大的时候，kNN分类的精度很高
- 如果训练集很小， kNN可能效果很差。
- kNN倾向于大类，可以将相似度考虑在内来缓解这个问题。

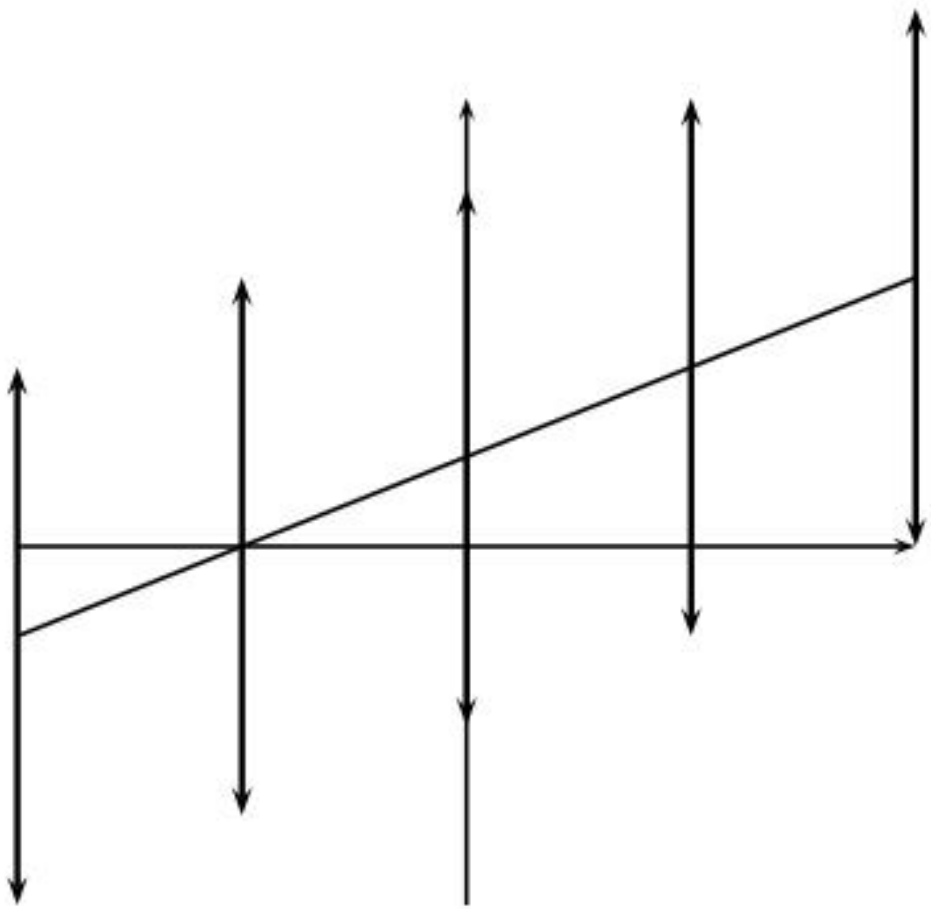
线性分类器

- 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， θ 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

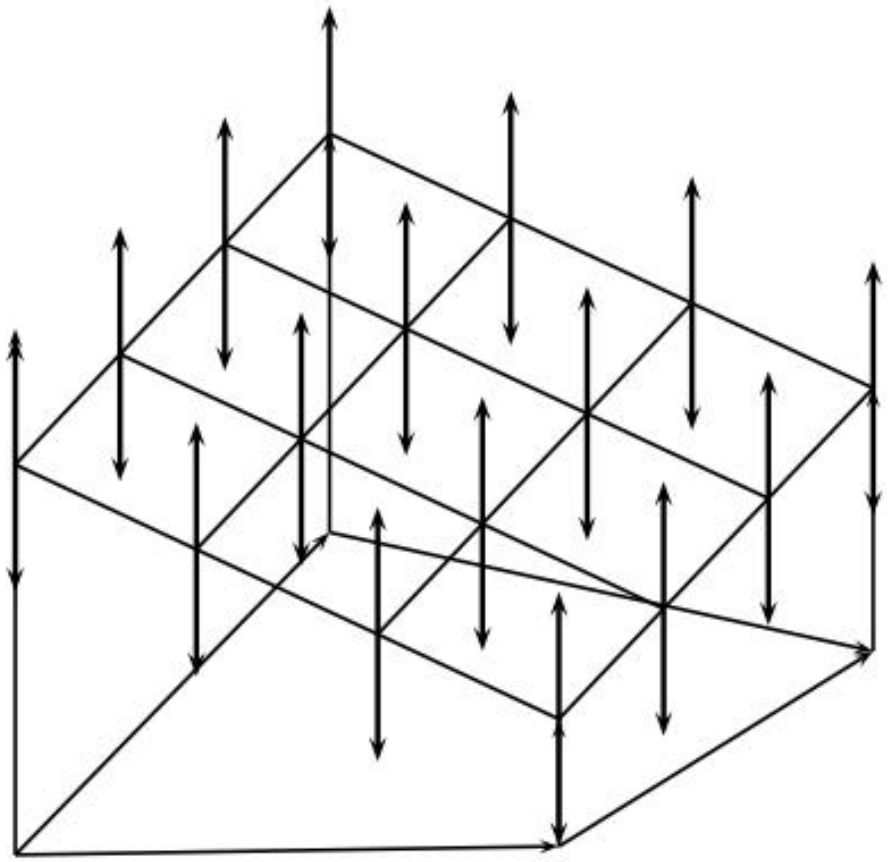
- 一维下的分类器是方程$w_1 d_1 = \theta$对应的点
- 点的位置是θ / w_1
- 那些满足$w_1 d_1 \geq \theta$ 的点d_1属于类别c
- 而那些$w_1 d_1 < \theta$的点d_1属于类别 $\bar{c}$.

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

$\overline{c}$.

Rocchio是一个线性分类器

- Rocchio的线性分类面定义为：

$$\sum_{i=1}^{M} w_i d_i = \vec{w}\vec{d} = \theta$$

- 其中 $\vec{w}$ 是向量 $\vec{\mu}(c_1) - \vec{\mu}(c_2)$ 的法向量，

$$\theta = 0.5 * (|\vec{\mu}(c_1)|^2 - |\vec{\mu}(c_2)|^2).$$

朴素贝叶斯也是线性分类器

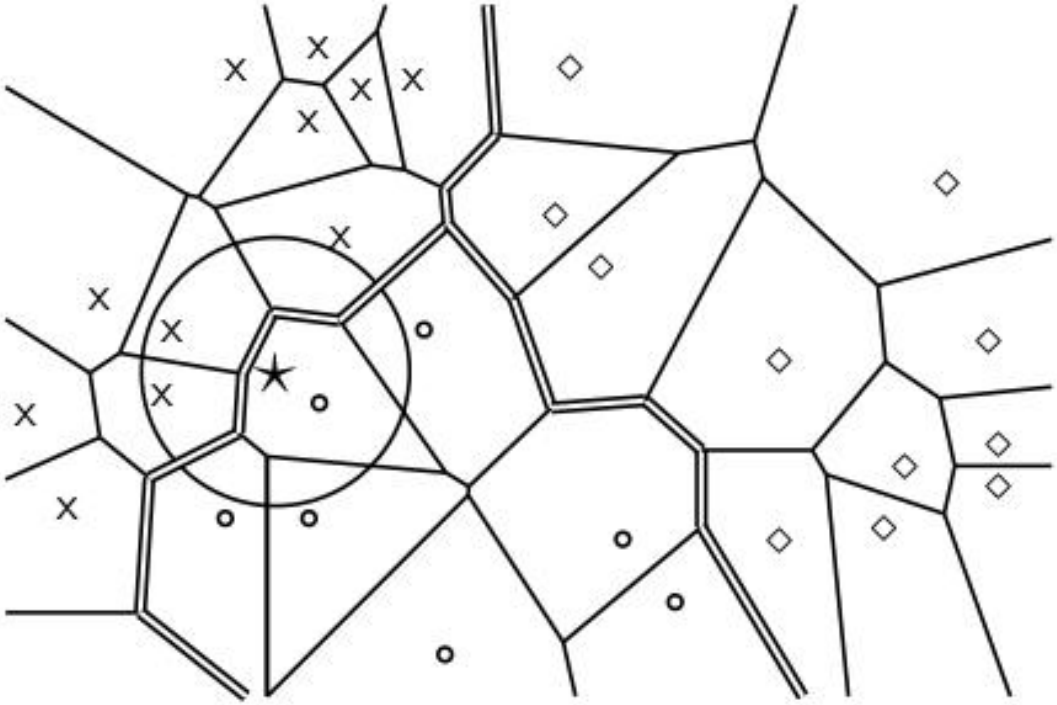
多项式模型的朴素贝叶斯也是线性分类器，其分类面定义为：

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

其中 $w_i = \log[\hat{P}(t_i|c)/\hat{P}(t_i|\bar{c})]$, $d_i = t_i$ 在d中的出现次数, $1 \le i \le M$, $\theta = -\log[\hat{P}(c)/\hat{P}(\bar{c})]$

(注意：这里的t_i指的是所有词汇表中的词项，而不是上一讲中出现在文档d中的词项)，

kNN不是线性分类器



- kNN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
- . . . 但是一般来说，很难表示成如下的 线性分类器

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

一个线性分类器的例子

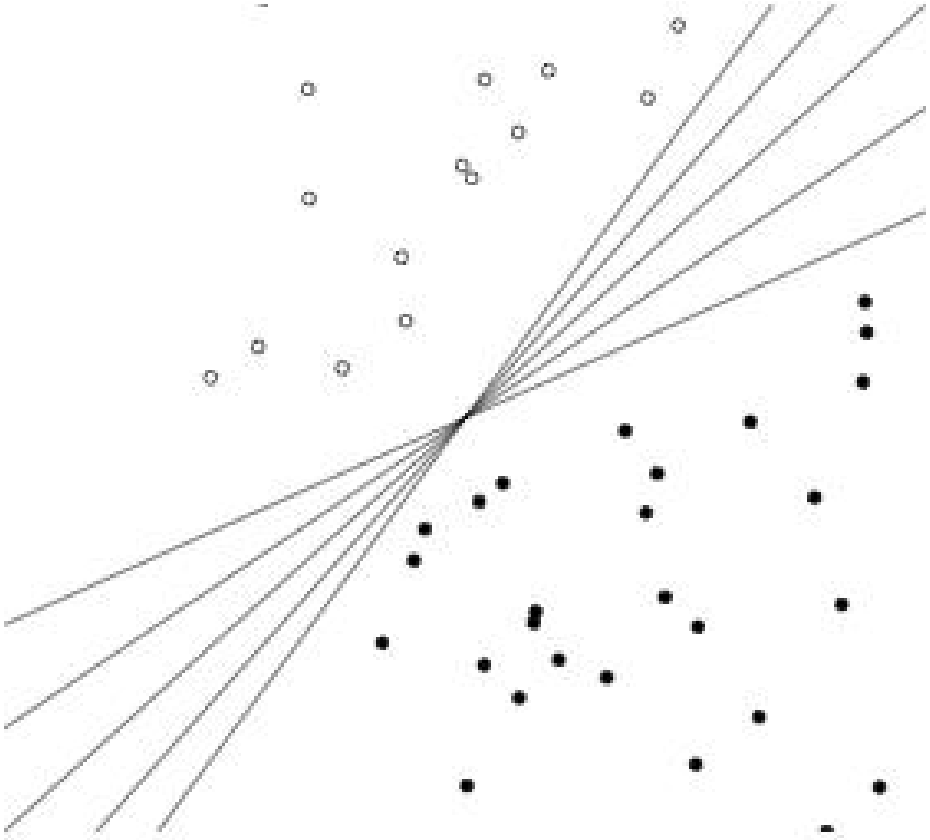
t_i	w_i	d_{1i}	d_{2i}	t_i	w_i	d_{1i}	d_{2i}
prime	0.70	0	1	dlrs	-0.71	1	1
rate	0.67	1	0	world	-0.35	1	0
interest	0.63	0	0	sees	-0.33	0	0
rates	0.60	0	0	year	-0.25	0	0
discount	0.46	1	0	group	-0.24	0	0
bundesbank	0.43	0	0	dlr	-0.24	0	0

- 对应Reuters-21578 语料中的*interest*类
- 简化起见：文档向量均采用布尔向量来表示
- d_1: “rate discount dlrs world”
- d_2: “prime dlrs”
- $\theta = 0$
- 课堂练习： 文档d_1、d_2分别属于哪一类？
- $\vec{d}_1$ rate discount dlrs world” 属于*interest*类:
- $\vec{w}^T \vec{d}_1 = 0.67 \cdot 1 + 0.46 \cdot 1 + (-0.71) \cdot 1 + (-0.35) \cdot 1 = 0.07 > 0 = \theta.$
- $\vec{d}_2$ prime dlrs” 不属于 *interest*类：
- $\vec{w}^T \vec{d}_2 = -0.01 \leq \theta.$

向量空间分类的学习算法

- 按照实际计算方法，主要有两类学习算法：
- (i) 简单学习算法：通过训练集直接估计分类器的参数，通常只需要单遍线性扫描
  - 如朴素贝叶斯、Rocchio、kNN等
- (ii) 迭代式学习算法：
  - 支持向量机(Support vector machine)
  - 感知机(Perceptron)
- 性能最好的学习算法通常需要迭代学习

应该选哪个超平面?



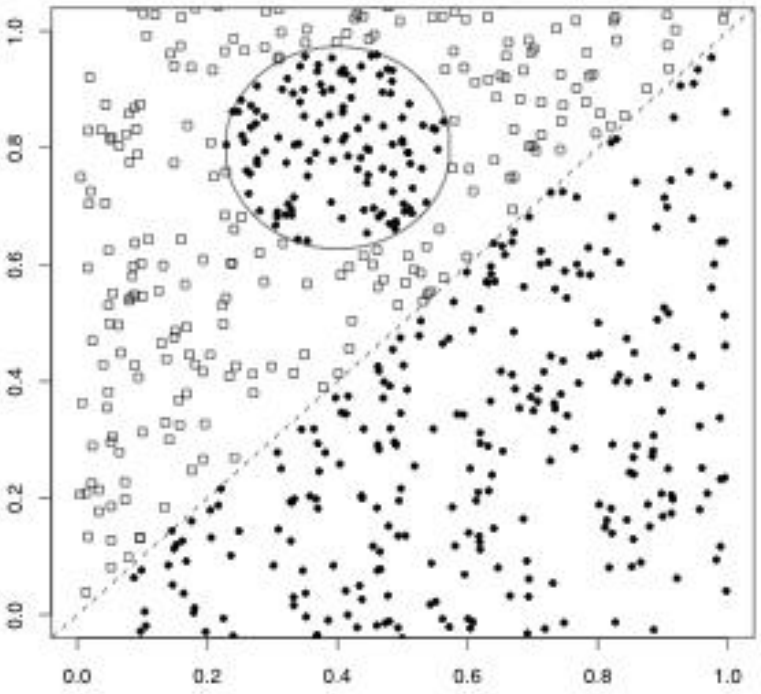
超平面的选择

- 对于线性可分的训练集而言，肯定存在无穷多个分类面可以将两类完全正确地分开
- 但是不同的分类面在测试集的表现完全迥异. . .
- 对于新数据，有些分类器的错误率很高，有一些却很低
- 感知机：通常很差；朴素贝叶斯、Rocchio：一般；线性SVM：好

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
- 不同的方法选择超平面的策略不同
  - 这造成在测试文档分类性能的巨大差异
- 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

一个非线性问题



- 诸如Rocchio的线性分类器在处理上述问题时效果很差
- 在训练集规模充分时，kNN 可以获得好的效果

给定文本分类问题下的分类器选择

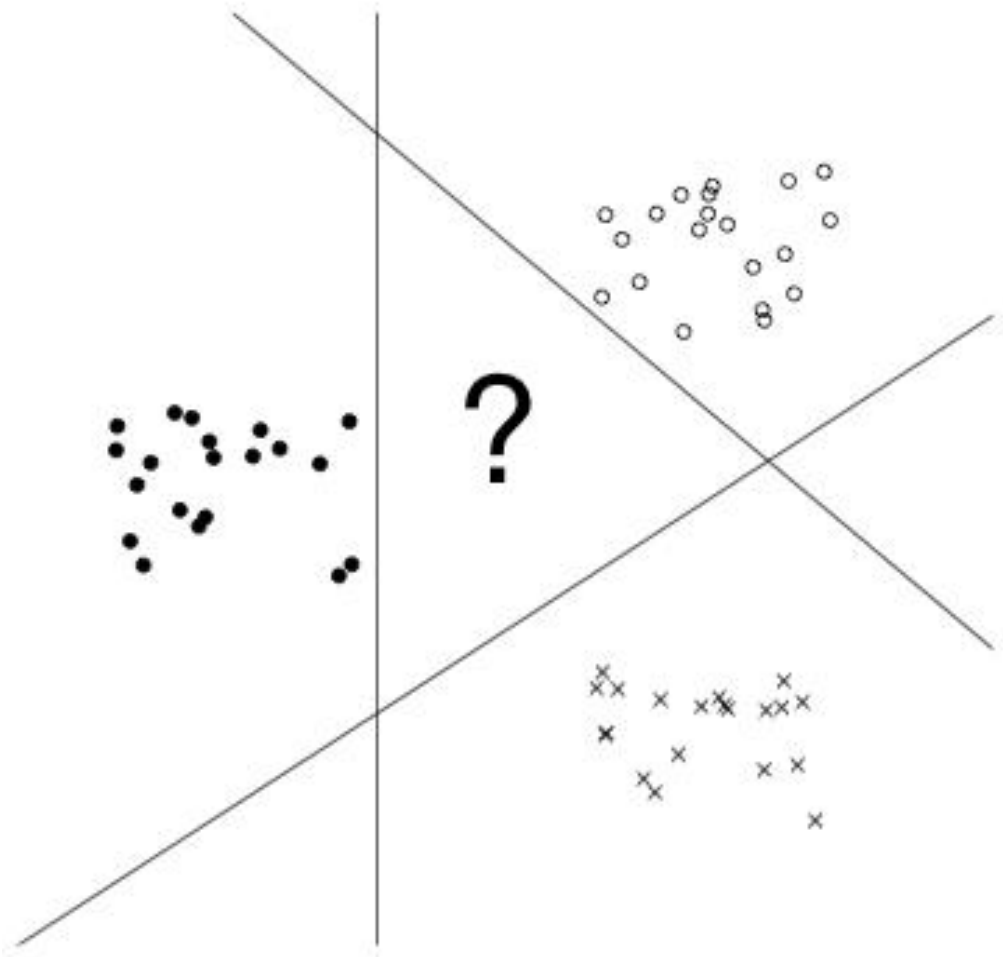
- 是否存在某个学习方法对于任何分类问题都最优？
- 答案显然是否，这是因为存在着偏差(bias)和方差(variance)之间的折中
- 需要考虑的因素:
  - 有多少训练数据？
  - 问题的复杂性如何？ (分类面是线性还是非线性？)
  - 问题的噪音有多大？
  - 随时间推移问题的稳定性如何？
    - 对于一个不稳定的问题，最好使用一个简单鲁棒的分类器

kNN vs. 朴素贝叶斯

- 偏差/方差 (Bias/Variance)这种准则
  - 方差 ≈ 记忆量(Capacity)
- kNN高方差低偏差
  - 无穷记忆(Infinite memory)
- NB低方差高偏差
  - 决策分类面必须是线性的
- 考虑问一个植物学家问题： Is an object a tree?
  - 高方差，低偏差(Too much capacity/variance, low bias
    - 记忆一切的植物学家 Botanist who memorizes
    - 对新对象总是回答no (e.g., 即使叶子数不同，也认为是不同品种)
  - 低方差，高偏差
    - 懒惰的植物学家
    - 如果对象是绿色的就回答 “yes”
  - 一般要在两者之间折中

(Example due to C. Burges)

多类(>2)的情况下的超平面组合



单标签问题（One-of problem）

- 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
  - 例子：文档的语言类型 (假定：任何一篇文档都只包含一种语言)

基于线性分类器的单标签分类

- 对于单标签分类问题（比如A、B、C三类），可以将多个二类线性分类器(A vsBC、B vs. AC、C vs. AB)进行如下组合：
  - 对于输入文档，独立运行每个分类器
  - 将分类器排序 (比如按照每个分类器输出的在A、B、C上的得分)
  - 选择具有最高得分的类别

多标签问题(Any-of problem)

- 多标签分类问题，也称multilabel classification
  - 一篇文档可以属于0、1或更多个类
  - 针对某个类的决策并不影响其他类别上的决策
  - 一种 “独立”类型，但是不是统计意义上的“独立”
  - 例子：主题分类
  - 通常：地区、主题领域、工业等类别之上的决策是互相独立的

基于线性分类器的多标签分类

- 对于多标签分类问题（比如A、B、C三类），可以将多个二类线性分类器(A vsBC、B vs. AC、C vs. AB)进行如下组合：
  - 对测试文档独立地运行每个分类器
  - 按照每个分类器自己的输出结果进行独立决策

对于给定的分类问题，应当使用哪种分类算法？

- 是否存在一个对所有分类问题均最优的算法？
- 答案是否定的。这是因为存在偏差与方差之间的权衡
- 需要考虑的因素：
  - 有多少训练数据?
  - 问题的简单/复杂程度？（线性 vs. 非线性 决策边界）
  - 数据噪声有多大?
  - 问题随时间推移是否保持稳定?
    - 对于不稳定的问题，最好使用简单鲁棒的分类算法