现代信息检索
Modern Information Retrieval

第10讲 文本分类与排序学习

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

分类方法（3）：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）

- 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

15

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}：

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

16

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：

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

17

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

18

参数估计：极大似然估计

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

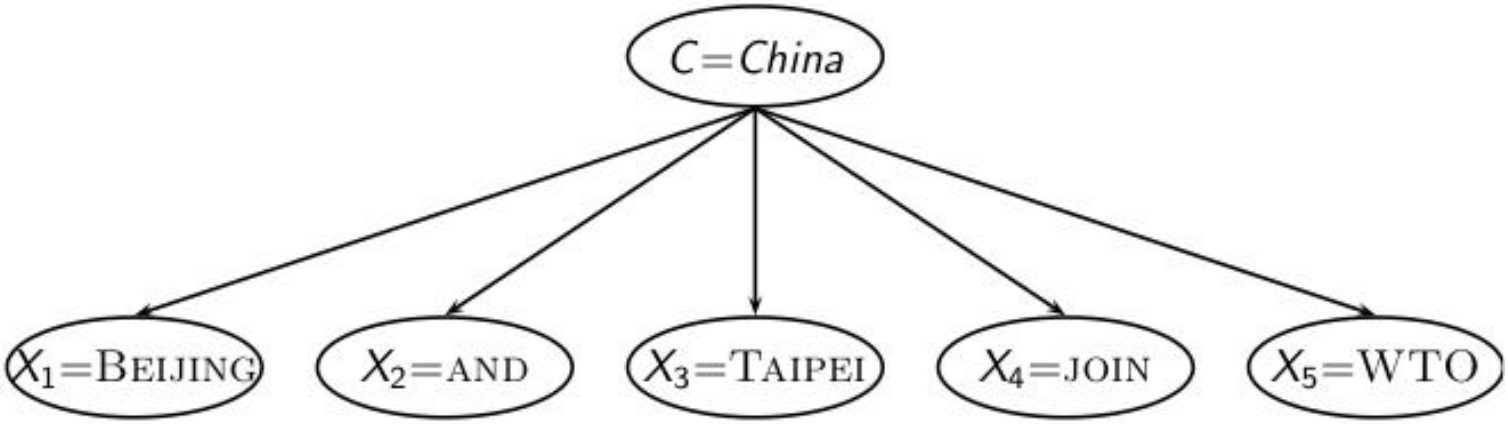
$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

- 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)：

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

19

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:

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

20

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

21

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 $|V| = B$)

22

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

23

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** $V, prior, condprob$

24

朴素贝叶斯: 测试/应用/分类

```
APPLYMULTINOMIALNB(ℂ, V, prior, condprob, d)
1  W ← EXTRACTTOKENSFROMDOC(V, d)
2  for each c ∈ ℂ
3  do score[c] ← log prior[c]
4     for each t ∈ W
5     do score[c] += log condprob[t][c]
6  return arg max_{c∈ℂ} score[c]
```


25

朴素贝叶斯的时间复杂度分析

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

- 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，$\mathbb{C}$: 类别集合
- $\Theta(|\mathbb{D}|L_{\text{ave}})$ 是计算所有统计数字的时间
- $\Theta(|\mathbb{C}||V|)$ 是从上述数字计算参数的时间
- 通常来说：$|\mathbb{C}||V| < |\mathbb{D}|L_{\text{ave}}$
- 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。**

26

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：
    - 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

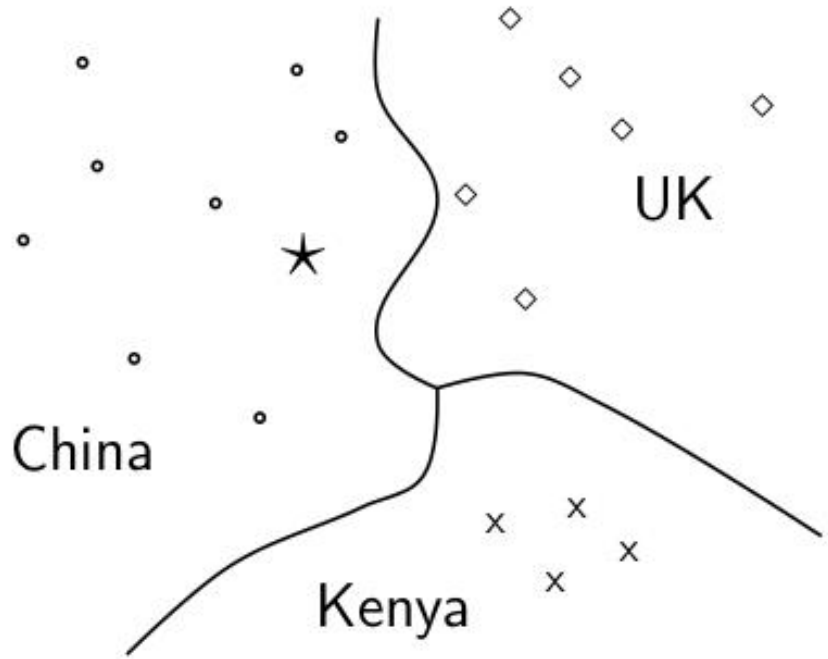
回顾: 向量空间表示

- 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。

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

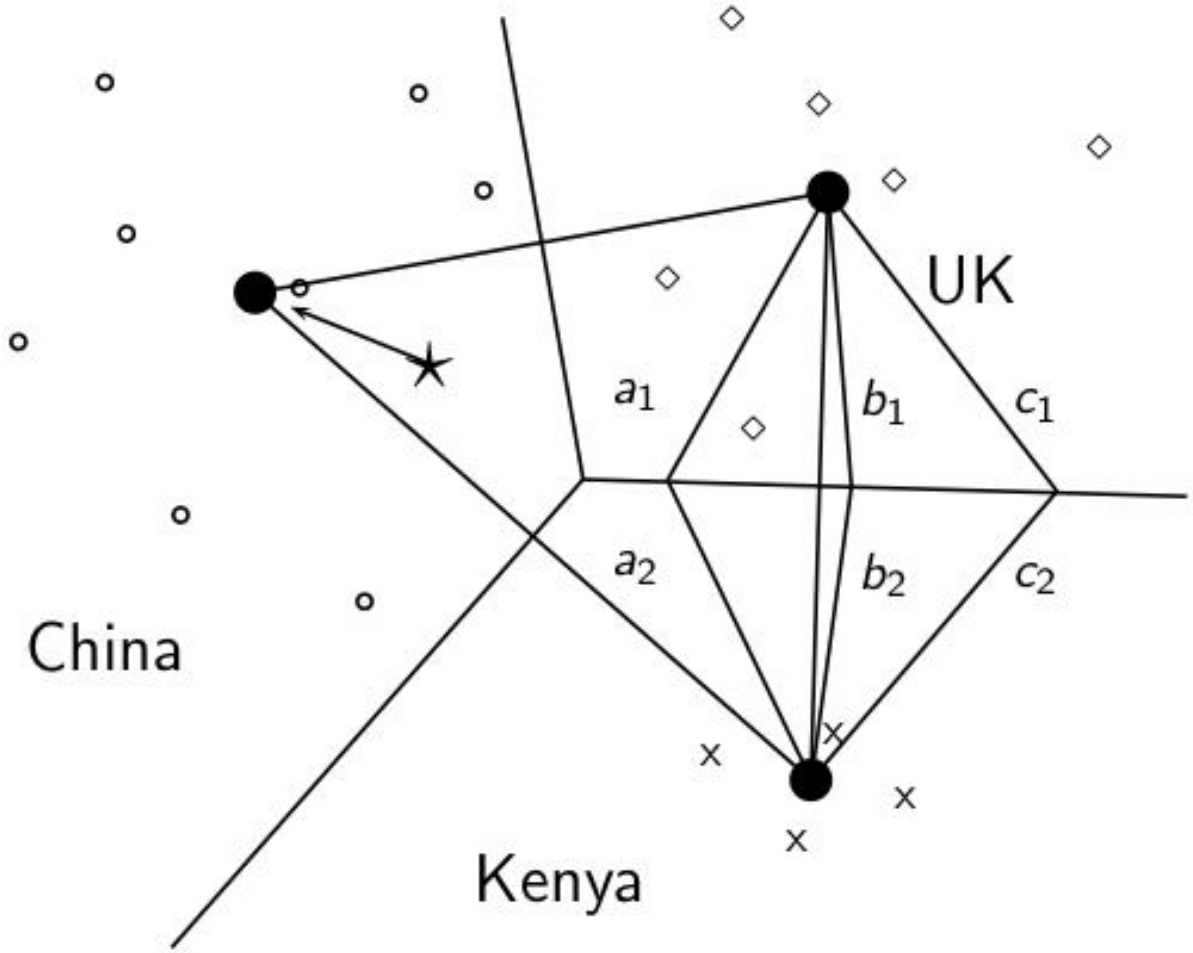
TRAINROCCHIO($\mathbb{C}, \mathbb{D}$)

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	$\Theta(\lvert\mathbb{D}\rvert L_{\text{ave}} + \lvert\mathbb{C}\rvert\lvert V\rvert) \approx \Theta(\lvert\mathbb{D}\rvert L_{\text{ave}})$
testing	$\Theta(L_{\text{a}} + \lvert\mathbb{C}\rvert M_{\text{a}}) \approx \Theta(\lvert\mathbb{C}\rvert M_{\text{a}})$

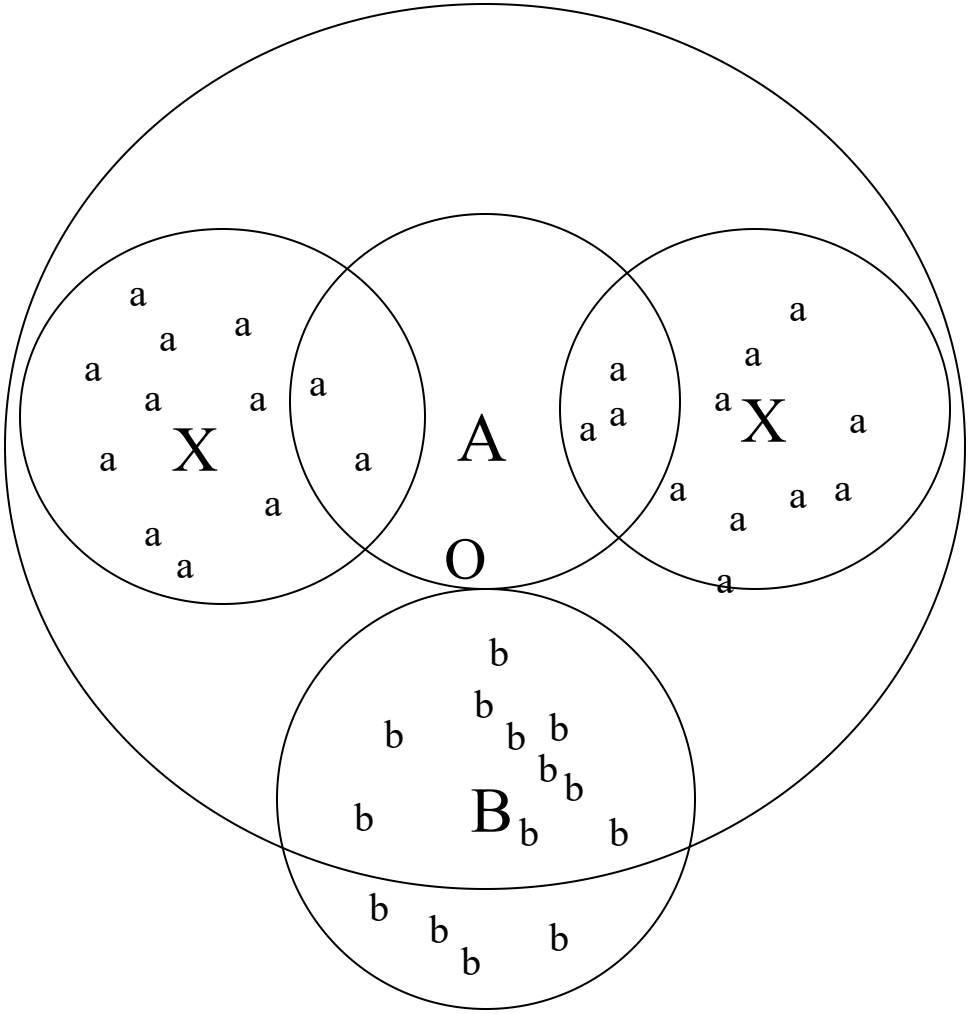
|D|：文档数量；|C|：类别数量；|V|：词典大小；
L_ave：平均文档长度；L_a：测试文档长度；
M_a：测试文档词汇数量

训练：与训练文档集体积线性相关
测试：与类别数量线性相关

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
  - 非凸规划：问题不能用凸函数描述。局部最优不一定全局最优

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(k nearest neighbour, k近邻)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

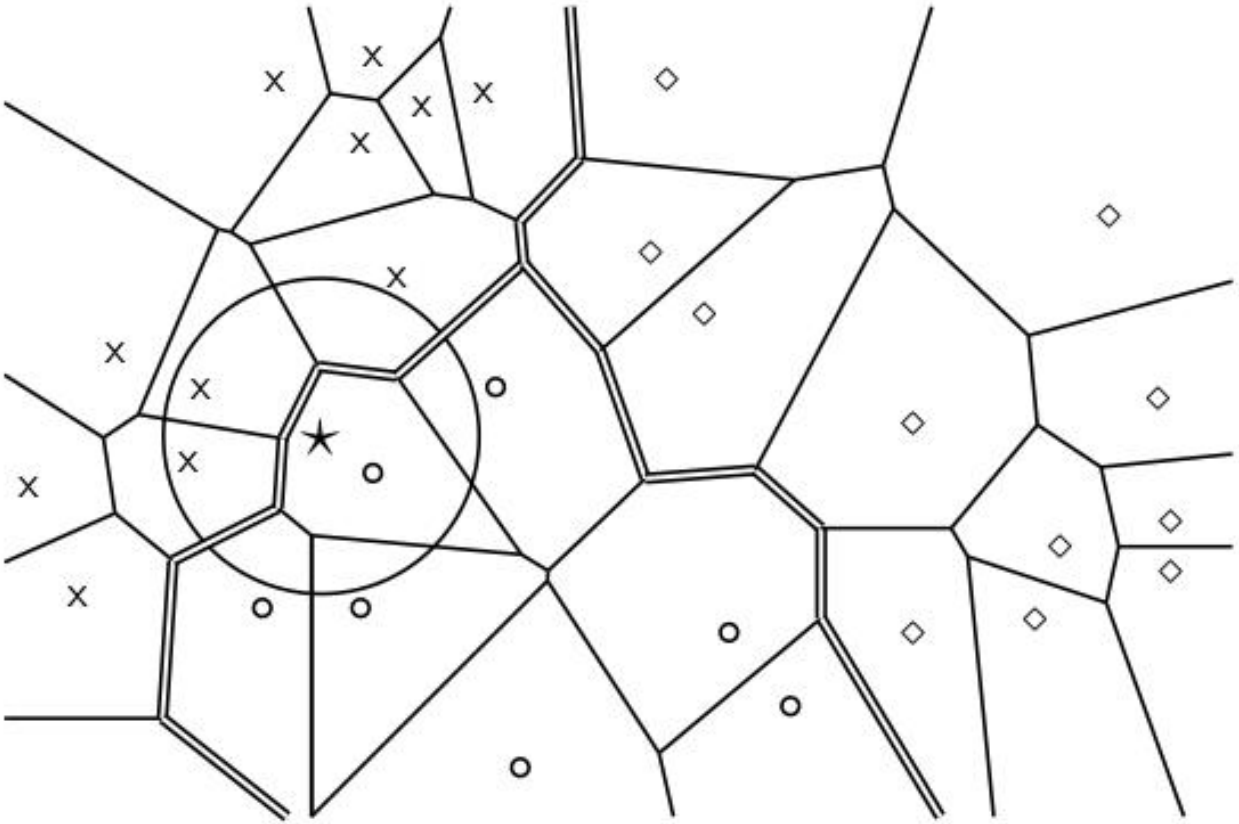
kNN分类

- 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** $\arg\max_j p_j$

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

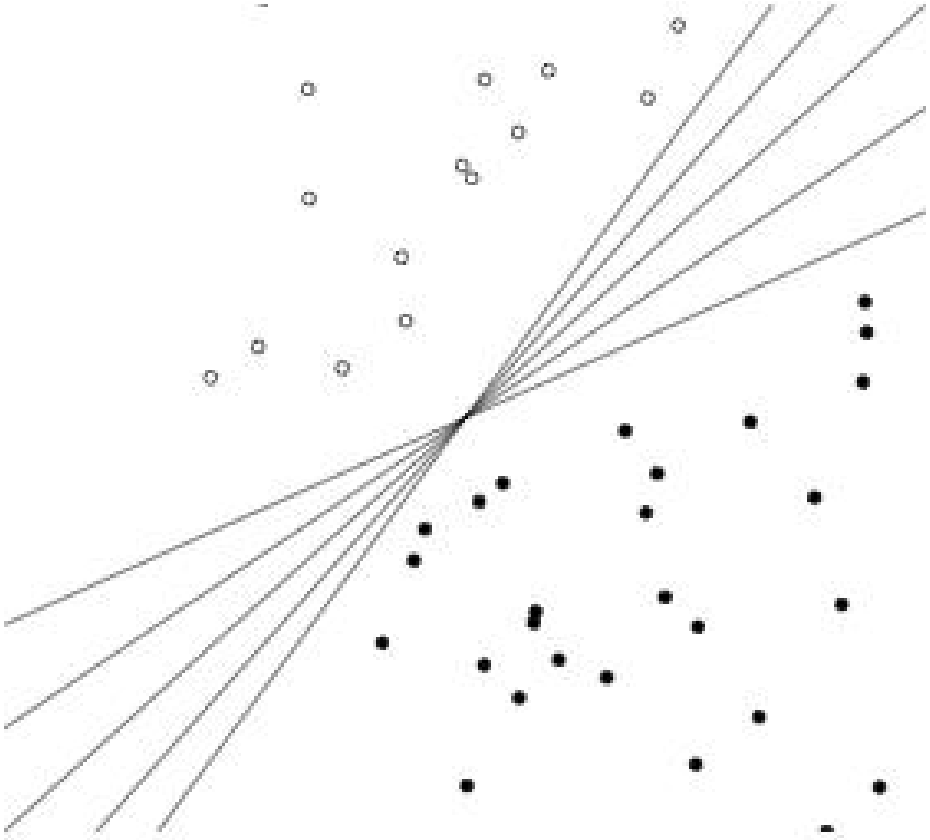
kNN: 讨论

- 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

应该选哪个超平面?



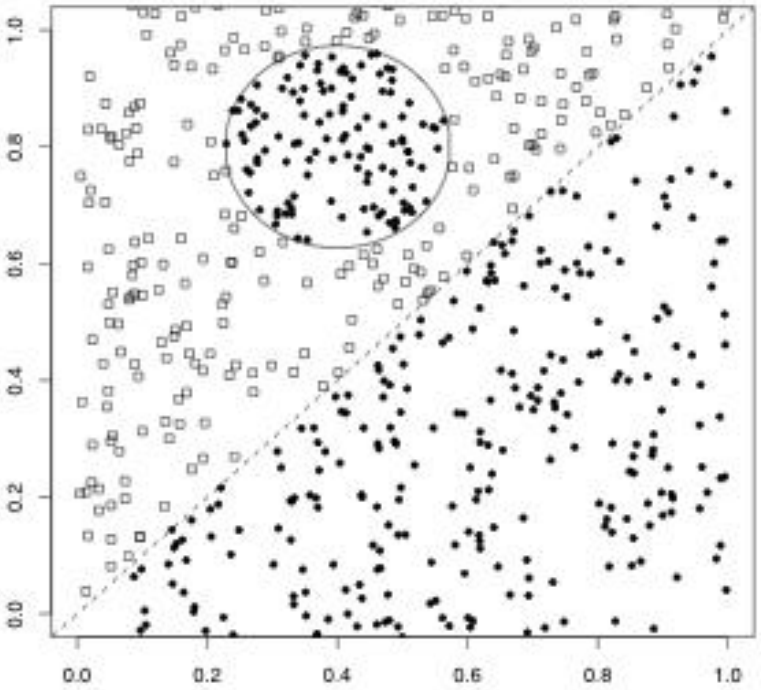
超平面的选择

- 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

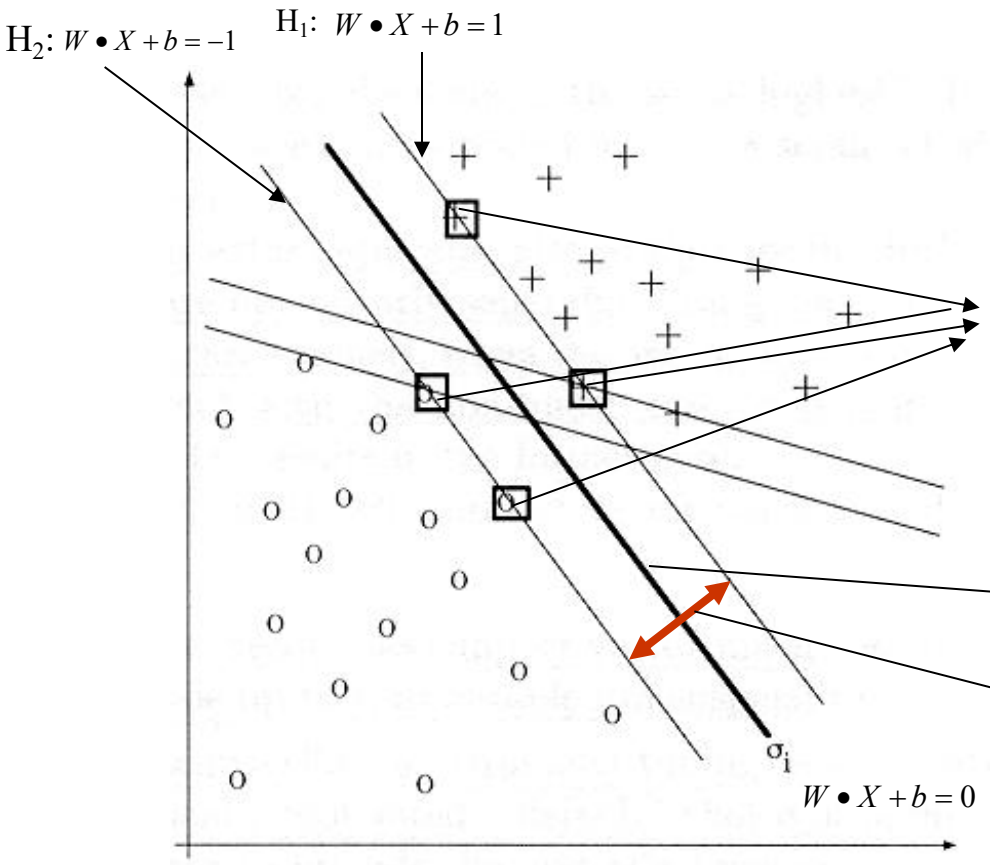
- 是否存在某个学习方法对于任何分类问题都最优？
- 答案显然是否，这是因为存在着偏差(bias)和方差(variance)之间的折中
- 需要考虑的因素:
  - 有多少训练数据？
  - 问题的复杂性如何？ (分类面是线性还是非线性？)
  - 问题的噪音有多大？
  - 随时间推移问题的稳定性如何？
    - 对于一个不稳定的问题，最好使用一个简单鲁棒的分类器

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)

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

其中*W*•*X*是内积(dot product)，*b*是标量。

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

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

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：

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

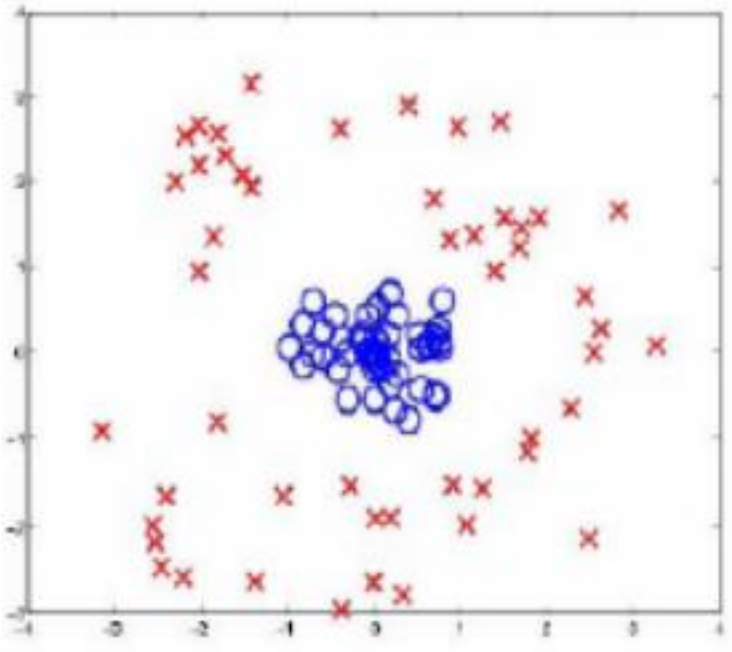
此时的目标函数是求下式的最小值：

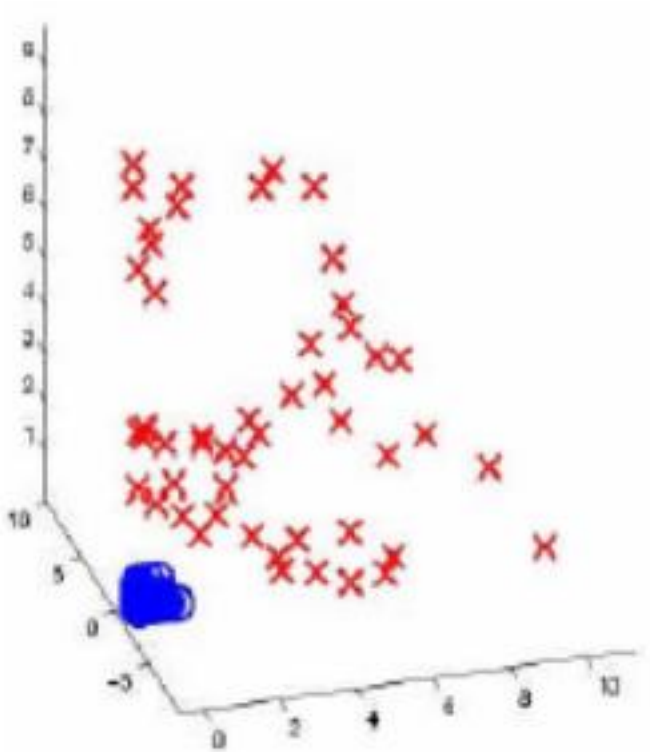
期望风险　　经验风险

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

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

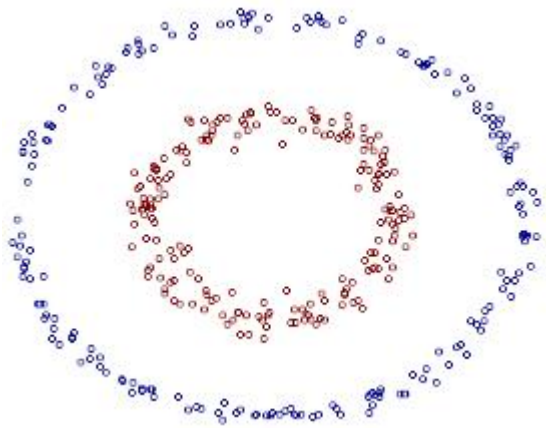
- 采用如下的内积函数(核函数):

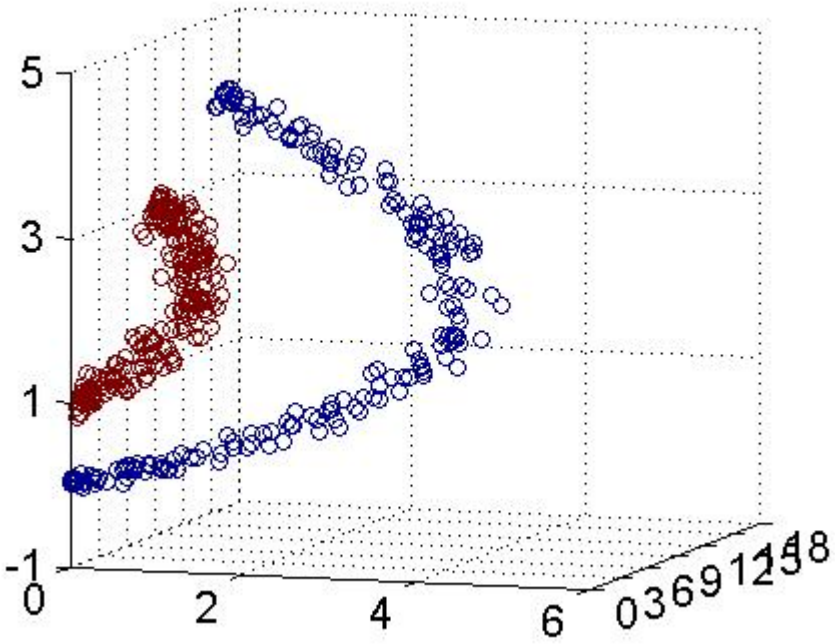
多项式 $K(X, X_i) = [(X \bullet X_i) + 1]^q$

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

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

将数据映射到高维空间，从而线性可分





判别函数成为：

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

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。

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，NB，SVM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再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

  → $g_1 = 0.2, g_2 = 0.3, g_3 = 0.5$ (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
Φ_1	37	linux	1	1	Relevant
Φ_2	37	penguin	0	1	Nonrelevant
Φ_3	238	system	0	1	Relevant
Φ_4	238	penguin	0	0	Nonrelevant
Φ_5	1741	kernel	1	1	Relevant
Φ_6	2094	driver	0	1	Relevant
Φ_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
Φ_1	37	linux	1	1	Relevant
Φ_2	37	penguin	0	1	Nonrelevant
Φ_3	238	system	0	1	Relevant
Φ_4	238	penguin	0	0	Nonrelevant
Φ_5	1741	kernel	1	1	Relevant
Φ_6	2094	driver	0	1	Relevant
Φ_7	3194	driver	1	0	Nonrelevant

- 对每个训练样例 Φ_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：

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

- 于是，整个训练集上的总错误为：

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

- 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ϵ 最小的g值

训练样例

Example	DocID	Query	s_T	s_B	Judgment
Φ_1	37	linux	1	1	Relevant
Φ_2	37	penguin	0	1	Nonrelevant
Φ_3	238	system	0	1	Relevant
Φ_4	238	penguin	0	0	Nonrelevant
Φ_5	1741	kernel	1	1	Relevant
Φ_6	2094	driver	0	1	Relevant
Φ_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 α)
  2. 查询词项在文档中存在的最小窗口宽度 (记为 ω)
     - 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

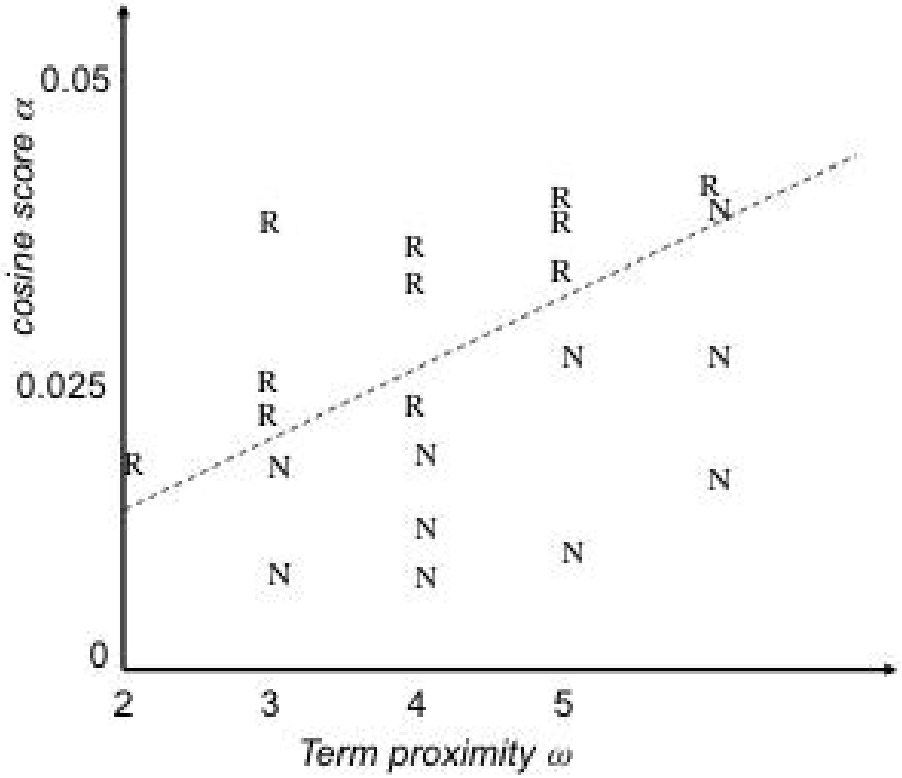
- 向量空间余弦相似度 α
- 窗口宽度 ω

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 (α, ω)

例子

Example	DocID	Query	Cosine score	ω	Judgment
Φ_1	37	linux operating system	0.032	3	*relevant*
Φ_2	37	penguin logo	0.02	4	*nonrelevant*
Φ_3	238	operating system	0.043	2	*relevant*
Φ_4	238	runtime environment	0.004	2	*nonrelevant*
Φ_5	1741	kernel layer	0.022	3	*relevant*
Φ_6	2094	device driver	0.03	2	*relevant*
Φ_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

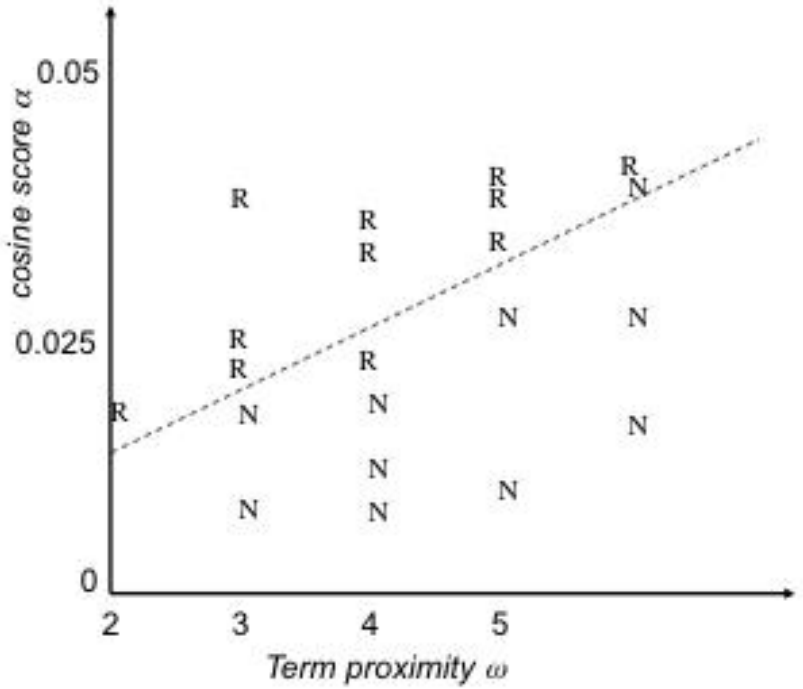
不失一般性，线性分类器可以采用下列通用形式：

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

公式中的系数 a、b、 c 可以从训练语料中学到

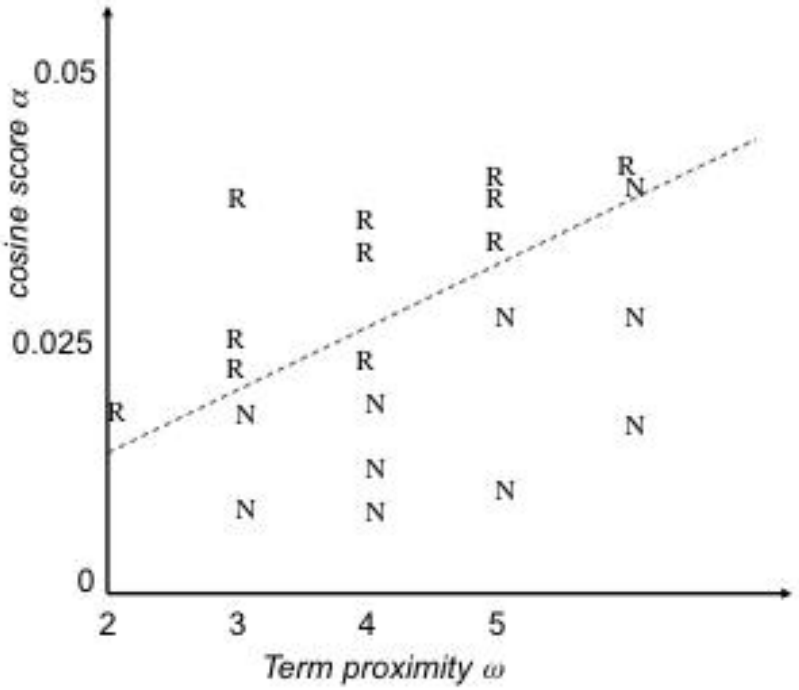
一个简单的机器学习评分的例子

- 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α-ω 平面上，该直线可以写成基于α 和ω (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：

$$\Phi(d_i , d_j , q) = \psi(d_i , q) - \psi(d_j , q)$$

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：

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

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

LTR参考文献

- Hang Li: Learning to Rank for Information Retrieval and Natural Language Processing, Second Edition. Synthesis Lectures on Human Language Technologies, Morgan & Claypool Publishers 2014

- Tie-Yan Liu: Learning to Rank for Information Retrieval. Springer 2011, ISBN 978-3-642-14266-6, pp. I-XVII, 1-285

- Tao Qin, Tie-Yan Liu, Jun Xu, Hang Li: LETOR: A benchmark collection for research on learning to rank for information retrieval. Inf. Retr. 13(4): 346-374 (2010)

- Tie-Yan Liu: Learning to Rank for Information Retrieval. Found. Trends Inf. Retr. 3(3): 225-331 (2009)