法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

- 课程详情请咨询
  - 微信公众号：小象
  - 新浪微博：ChinaHadoop

分布式爬虫

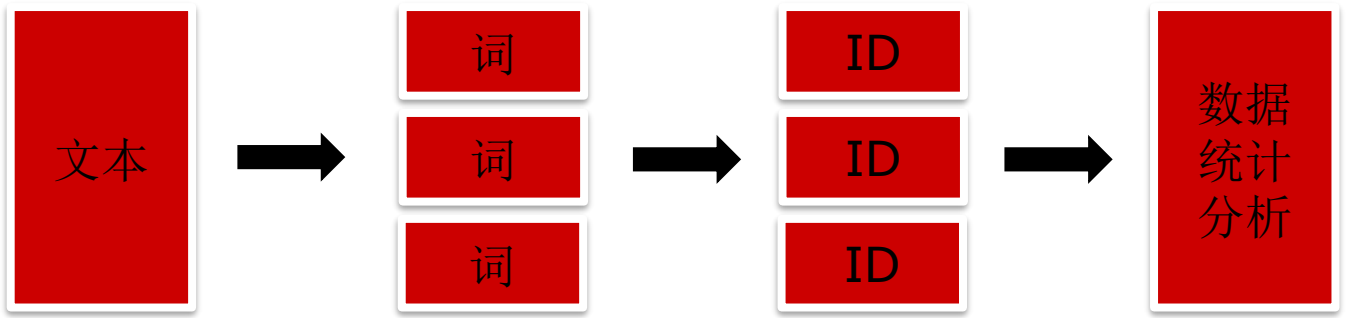
大纲

- 分词
- TF-IDF
- 线性回归
- Logistic 回归
- SVM
- 多分类器

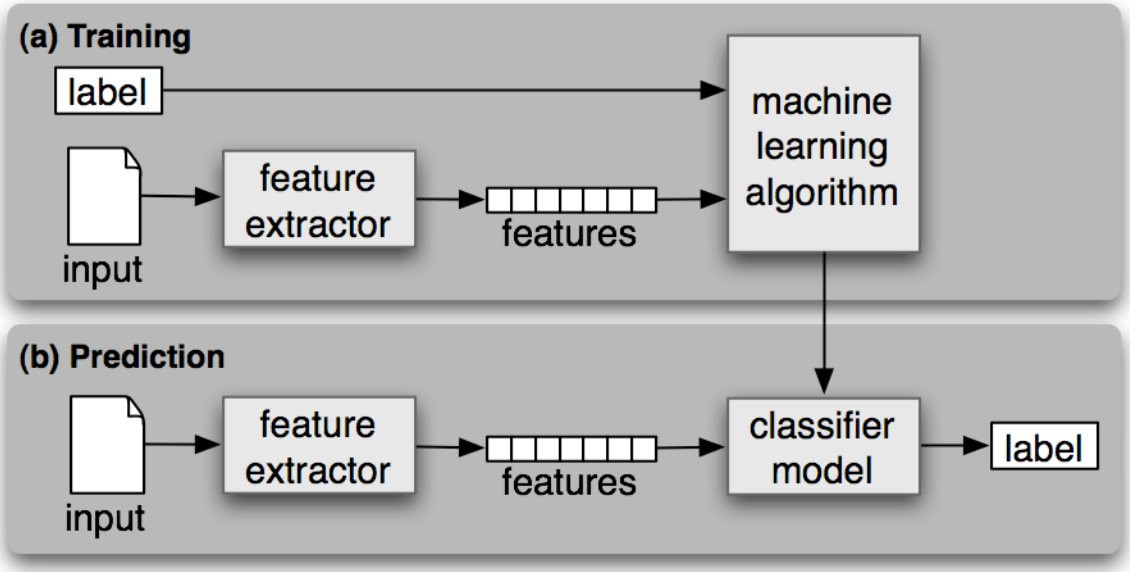
分词

计算机如何读懂文本？

计算机是无法直接处理、读懂文本的语义的，因此要对文本进行处理，必须把一篇非结构化的连续的文本，转换为一个数学问题。目前最常用的转换，就是找出文本的关键词，把关键词用一个数学特征来代替，进而利用 Logistic Regression、Support Vector Machine、Naïve Bayes 等办法来处理。



计算机如何读懂文本？


(a) Training
label
input
feature extractor
features
machine learning algorithm
(b) Prediction
input
feature extractor
features
classifier model
label

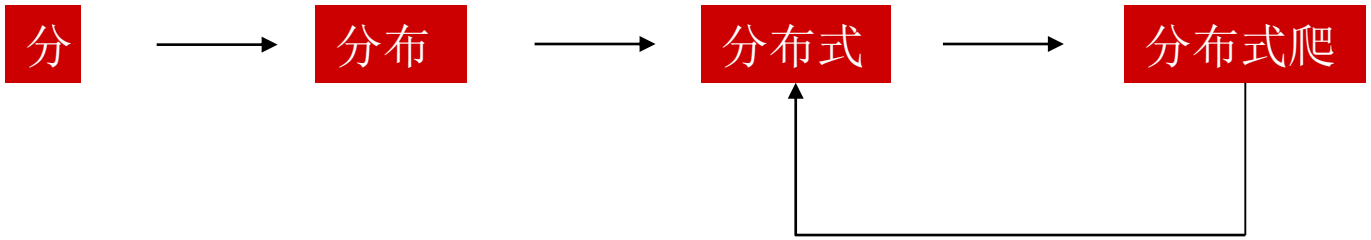
中文分词

分词最主要是针对中文的，因为以英文为代表的拉丁文语系，文本的单词之间是有天然分隔的，而中文则没有。

中文分词的手段主要依靠字典和统计学结合。分词是所有基于文本的应用的基础，分词效果直接决定了后续的使用。如果分词质量不高，后面的应用都会出错。比如 “清青花瓷”如果分为了 “清/青花/瓷” 这样3个词，那么在搜索引擎里搜索 青花瓷，就会检索不到任何结果。

词搜索

分词首先是基于词典的，也就是对于一句话，依次对字的组合与词典做比较，来发现一个词。比如“分布式爬虫是包含了分布式存储、任务管理、分布式数据库和爬虫进程的一套数据抓取系统”，分词的过程



分词的歧义

- 交集型歧义：从小/学/电脑，从/小学/毕业
- 组合型歧义：一位/美军/中将/曾经/说，新建/的/地铁/中/将/有3条线路
- 混合型歧义：人才能：人才/能，人/才能

对于歧义，需要依赖上下文来处理，有时候可以用正向最大匹配与逆向最大匹配（即从最后一个字往前来匹配词典）来同时提取，比如从小学电脑，正向会提取出 从小/学/电脑，逆向会提取出 从/小学/电脑，这样分出 从小、从、学、小学、电脑 5个词

结巴分词

结巴分词是一个python 的中文分词库

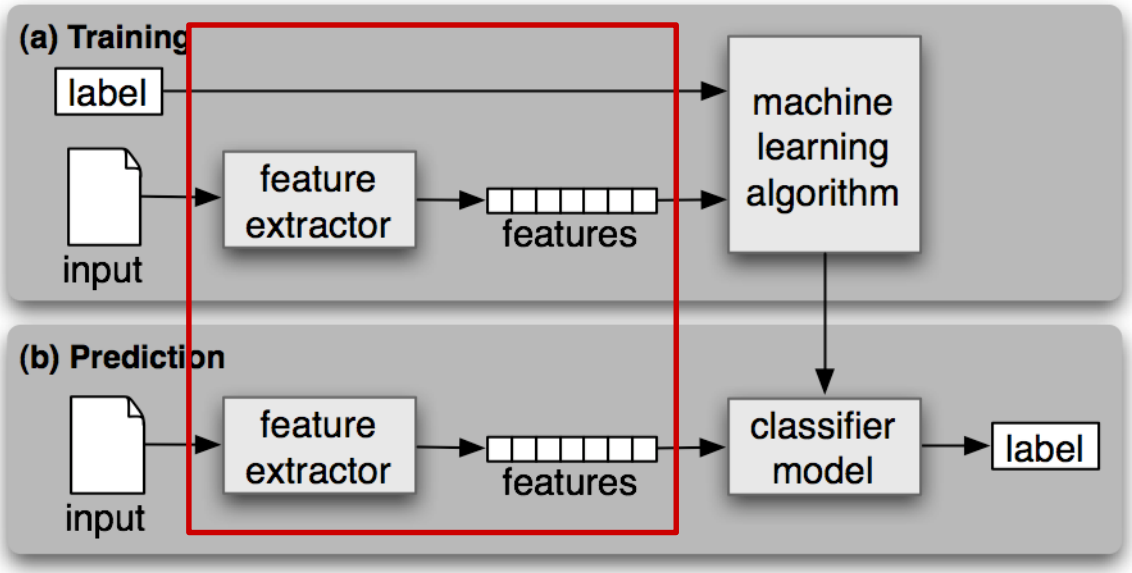
pip install jieba

```
import jieba

text = '分布式爬虫是包含了分布式存储、任务管理、分布式数据库和爬虫进程的一套数据抓取系统'

words = list(jieba.cut(text))

print ','.join(words)
```

分布式,爬虫,是,包含,了,分布式,存储,、,任务,管理,、,分布式,数据库,和,爬虫,进程,的,一套,数据,抓取,系统

TF-IDF

机器学习与分类算法


(a) Training
label
input
feature extractor
features
machine learning algorithm
(b) Prediction
input
feature extractor
features
classifier model
label

如何寻找特征词？

如同之前我们对文本排重一样，分类的前提也是要对文本进行降维，分类的降维过程，就是用一篇文章的特征词，构成一个集合，来代表这篇文章进行分类处理

接下来的问题就是，我们如何抽取出一篇文章的特征词？

寻找特征词

- 在英文中，把没有任何特征含义的词汇，比如介词、副词等统称为stopwords，也就是成为停用词。比如 的、是、在、因为、所以 等这样的词汇。这些停止词需要去除掉
- 统计每一篇文章里，各个词的词频，称为 Term Frequency

$$tf_i = \frac{N_i}{K}$$

- 对每一个词汇，统计它在各个文档里出现的频次，用总文件数/包含该词汇的文件的数量，再取对数，称为 Inverse Document Frequency

$$idf_i = log\frac{|D|}{|\{j: t_i \in d_j\}|}$$

TF-IDF

$$tf_i \times idf_i = \frac{N_i}{K} \times log \frac{|D|}{|\{j: t_i \in d_j\}|}$$

- 一个词在一篇特定文章里，出现次数多，所得到的权重会增大
- 一个词在所有的文档里，出现的次数少，权重会增大
- 分类处理的时候，如果一个词在训练集里，C_i类别的 IDF 值很高，而在别的类别的 IDF 很低，那么这个词可以是这个类别的特征词，因此可以额外提高它的权重

Python TF-IDF

pip install sklearn scipy numpy

```
from sklearn.datasets import load_files
from sklearn.cross_validation import train_test_split
from sklearn.feature_extraction.text import  TfidfVectorizer
import jieba
```

构造Tfidf对象，指定分词器为 jieba，停止词是一个数组，用户自定义的

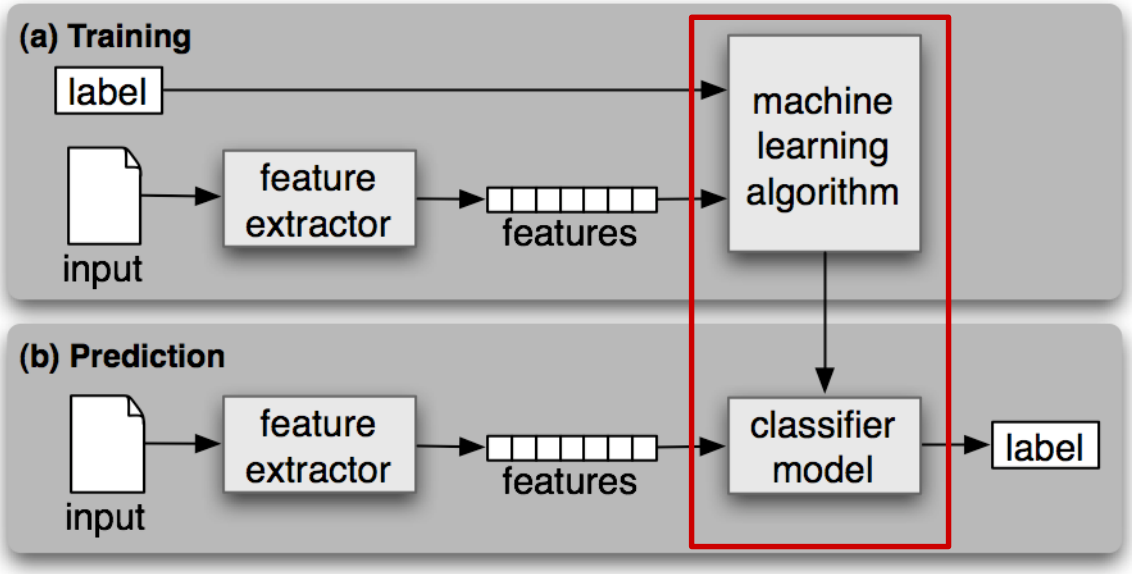
```
count_vec = TfidfVectorizer(binary = False, decode_error = 'ignore', tokenizer=jieba.cut, stop_words=stop_words)
```

获取特征词及权重

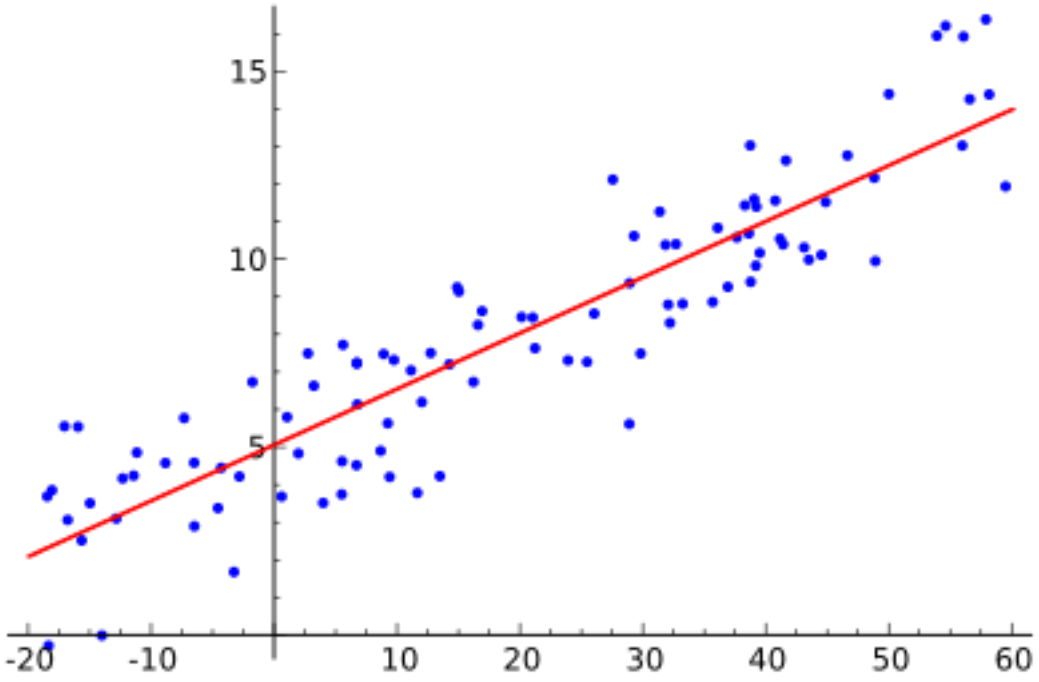
```
count_vec.get_feature_names()
```

线性回归

机器学习与分类算法


(a) Training
label
input
feature extractor
features
machine learning algorithm
(b) Prediction
input
feature extractor
features
classifier model
label

线性回归



$$y^{(i)} = \theta^T x^{(i)} + \varepsilon^{(i)}$$

误差$\varepsilon^{(i)}$ 是独立同分布的，均值为0，方差为固定值 σ^2

线性回归

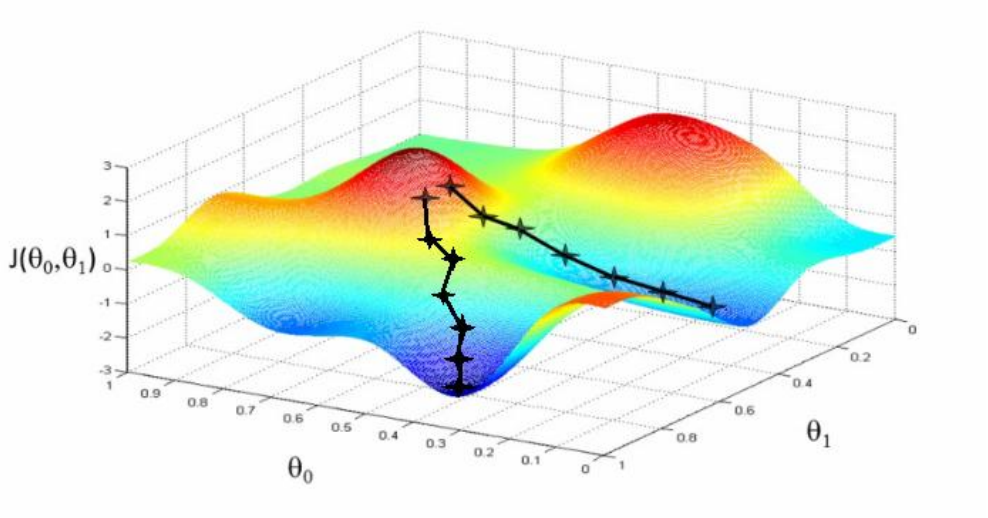
$$y^{(i)} = \theta^T x^{(i)} + \varepsilon^{(i)} \qquad h_\theta(x) = \theta^T x$$

利用最小二乘来估计，转化为求代价函数的最小值

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

梯度下降法来计算

$$\theta_j = \theta_j - \alpha \cdot \frac{\partial J(\theta)}{\partial \theta}$$



线性回归

$$\theta_j := \theta_j + \alpha\left(y^{(i)} - h_\theta\left(x^{(i)}\right)\right)x_j^{(i)}$$

$$\theta_j \in [\theta_0, \theta_1, \theta_2, \theta_3, \dots, \theta_n]$$

$$i = 1,2,3,\dots,n \quad j = 1,2,3,\dots,n$$

计算过程：

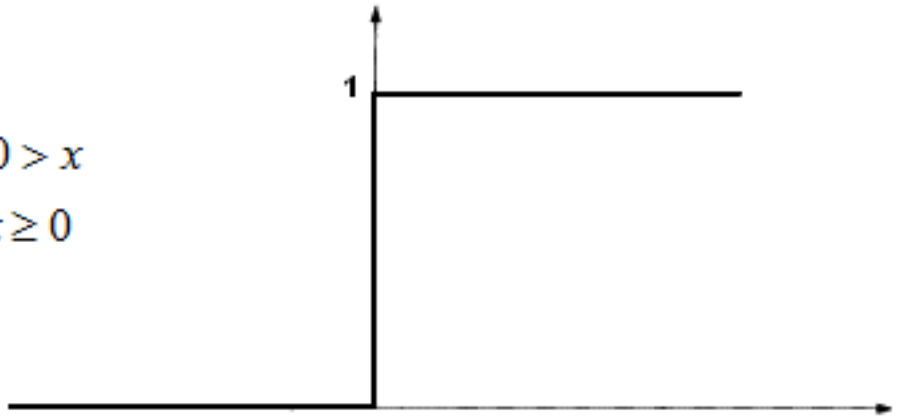
```
for j in range(0,n+1):
    for i in range(0, n+1):
```

$$\theta_j := \theta_j + \alpha\left(y^{(i)} - h_\theta\left(x^{(i)}\right)\right)x_j^{(i)}$$

分类问题

Unit step (threshold)

$$f(x) = \begin{cases} 0 \text{ if } 0 > x \\ 1 \text{ if } x \geq 0 \end{cases}$$



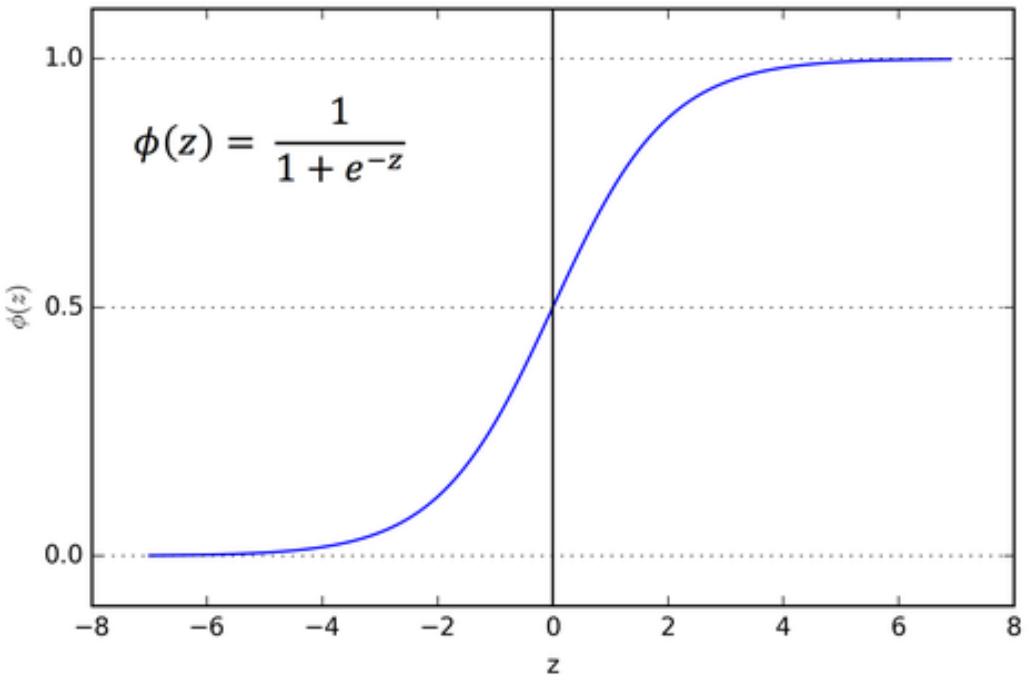
Sigmoid 函数

线性回归一般是用来做回归，而不用来做分类，如果要做分类，我们可以借助sigmoid函数，将线性回归的基础方程

$$y = \theta^T x + \varepsilon$$

代入到 Sigmoid 函数中

$$y = \frac{1}{1 + e^{-(\theta^T x + \varepsilon)}}$$



Logistic Regression

$$\theta_j := \theta_j + \alpha\left(y^{(i)} - h_\theta\left(x^{(i)}\right)\right)x_j^{(i)}$$

$$\theta_j \in [\theta_0, \theta_1, \theta_2, \theta_3, \dots, \theta_n]$$

$$i = 1,2,3,\dots,n \quad j = 1,2,3,\dots,n$$

参数计算过程与线性回归完全一样

Python的 Logistic 回归

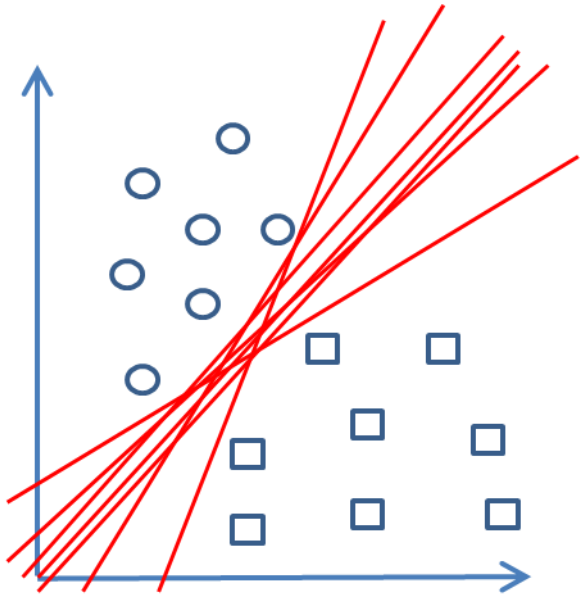
```
# 构造 LogisticRegression 对象
lr = LogisticRegression()

# 用基于 tfidf 的特征词表与训练数据的标签来训练模型
lr.fit(X_train_tfidf, movie_reviews.target)

# 构造测试样本
ratings_new = ['电影很好看','不好看','很不错的电影，太棒了','太赞了，很值得看的电影','烂','好让我失望']
X_new_counts = count_vec.transform(ratings_new)
X_new_tfidf = tfidf_transformer.transform(X_new_counts)

# 预测
lr.predict(X_new_tfidf)
```

输出结果： [1, 0, 1, 1, 0, 1]

SVM

线性分割

考虑右边的情况，有很多条直线可以把两个类别分开，我们如何找出最优的一条直线，把两个类别进行切分？



线性分割

超平面的函数方程式：

$$w^T x + b = 0$$

超平面可以由法向量w与位移b 来确定，样本中任意点 x 到超平面的距离：

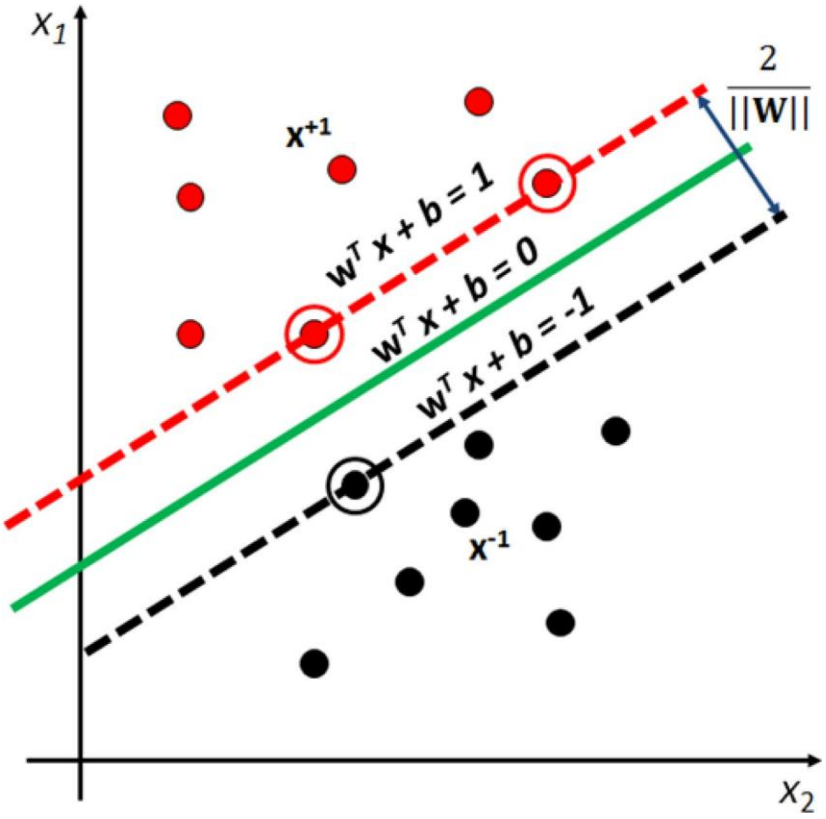
$$r = \frac{|w^T x + b|}{\|w\|}$$

对于$w^T(x_i, y_i) \in D$，令

$$\begin{cases} w^T x + b \geq +1, & y_i = +1 \\ w^T x + b \leq -1, & y_i = -1 \end{cases}$$

距离超平面最近的训练样本的点被成为支撑向量（Support Vector），两个异类支撑向量到超平面距离之和

$$\gamma = \frac{2}{\|w\|}$$



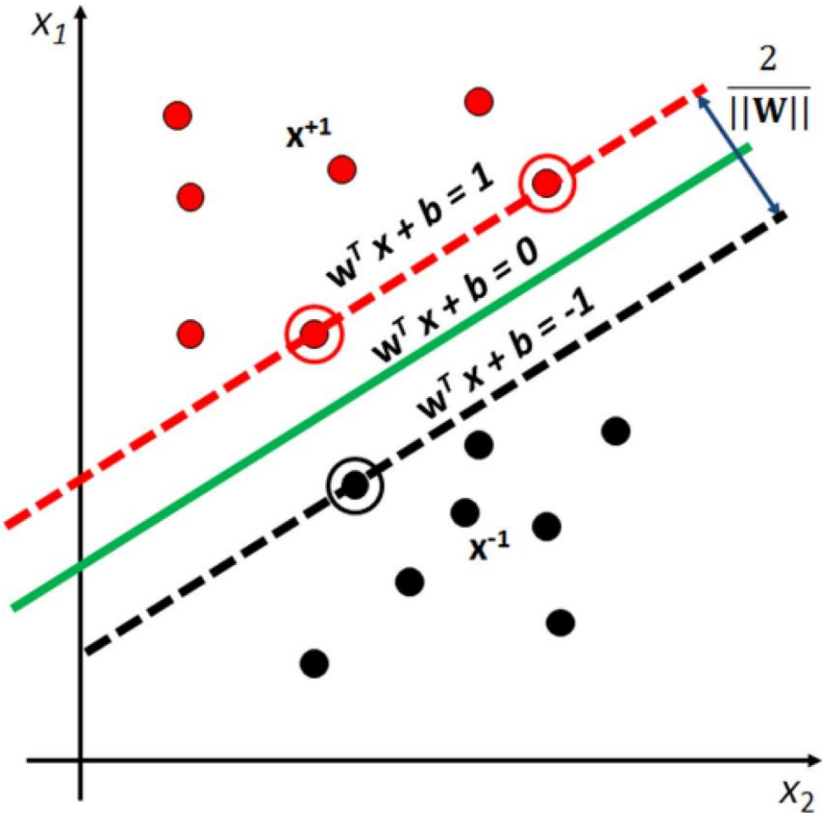
推导

要找到最大间隔，显然就是使得 γ 最大

$$max \frac{2}{\|w\|}$$

等价于

$$min \frac{1}{2} \|w\|^2$$



推导

求解方程式：

$$min\ \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j\left(\Phi(x_i)\cdot\Phi(x_j)\right)-\sum_{i=1}^{n}\alpha_i$$

$$s.t.\ \sum_{i=1}^{n}\alpha_i y_i=0\,,\quad \alpha_i\geq 0,\qquad i=1,2,3,\ldots,n$$

利用SMO（Sequential minimal Optimization) 来求解 α，带入下面的等式，就可以得到 SVM 的分割超平面

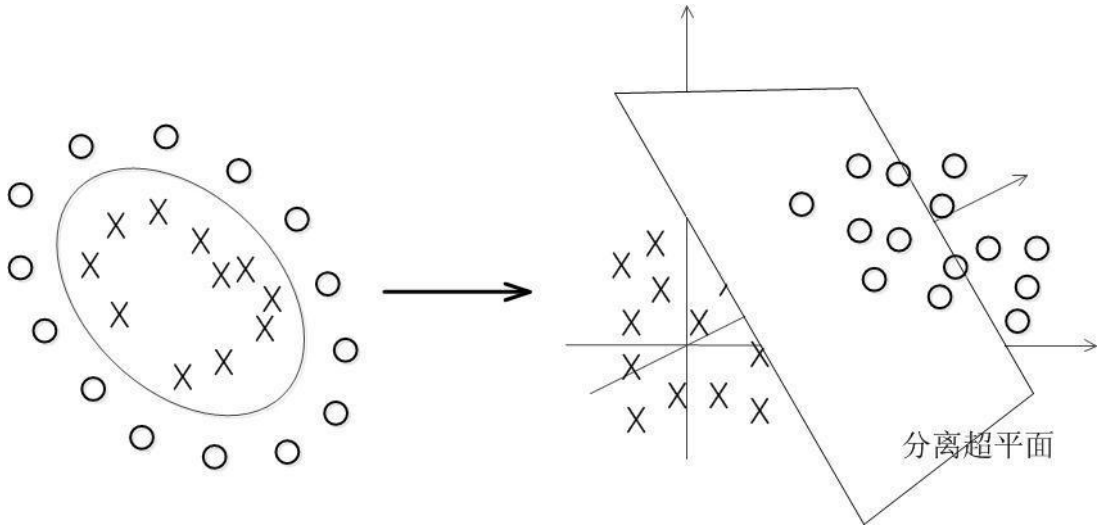
$$w=\sum_{i=1}^{n}\alpha_i y_i x_i$$

分割超平面 ⟹ $wx+b=0$

$$b=\frac{\max\limits_{i:y_i=-1} w\cdot x_i+\min\limits_{i:y_i=1} w\cdot x_i}{2}$$

核函数

对于下图所示的数据集合，是线性不可分的，所以需要把它们映射到一个高维空间，再来进行分割。训练样例一般是不会独立出现的，它们总是以成对样例的内积形式出现，而用对偶形式表示学习器的优势在为在该表示中可调参数的个数不依赖输入属性的个数，通过使用恰当的核函数来替代内积，可以隐式得将非线性的训练数据映射到高维空间。



核函数

$$min\ \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(x_i,x_j)-\sum_{i=1}^{n}\alpha_i$$

核函数本质，就是把内积，转换为通过线性函数可以展开为高维的形式，使得计算本身是在低维上进行，而分类的效果表现在了高维上

$x \to a+bx+cx^2$ 直接转换后内积，计算量显著增大

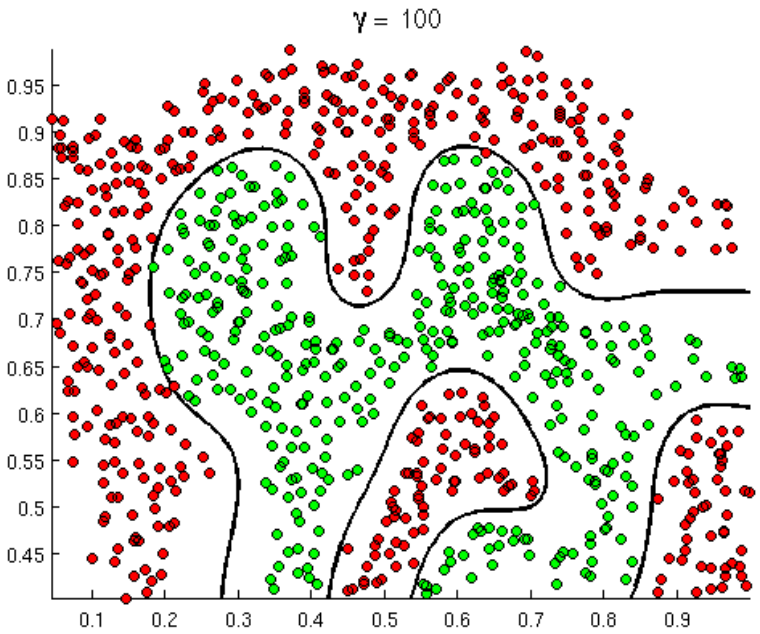
$x \to (ax+b)^2 = a^2x^2+2abx+b^2$，但是计算就与一维是一样的

高斯函数本身是无穷维的： $e^x = 1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\cdots+\frac{x^n}{n!}+R_n$

高斯核函数： $K(x_i,x_j) = e^{-\frac{\|x_1-x_2\|^2}{2\sigma^2}}$

核函数

除了高斯核函数，常用还有多项式核函数、Sigmoid 核函数等，核函数的使用依赖经验和交叉验证方案，如果没有太多的信息，可以简单使用高斯核函数。核函数的选择直接影响非线性分类的效果



Python SVM 文本分类

```
# 构造 Linear Kernel 的 SVC
linear_svc = SVC(kernel='linear')
# 构造 rbf 的 SVC，rbf 是默认kernel，因此等价于 SVN()
rbf_svc = SVC(kernel='rbf')
# 构造 sigmoid 的 SVC
sigmoid_svc = SVC(kernel='sigmoid')
# 构造 poly 的 SVC
poly_svc = SVC(kernel='poly')

# 用基于 tfidf 的特征词表与训练数据的标签来训练模型
linear_svc.fit(X_train_tfidf, movie_reviews.target)
```

测试结果：高维映射后的非线性分类效果并不理想

多分类器

多个分类的情况

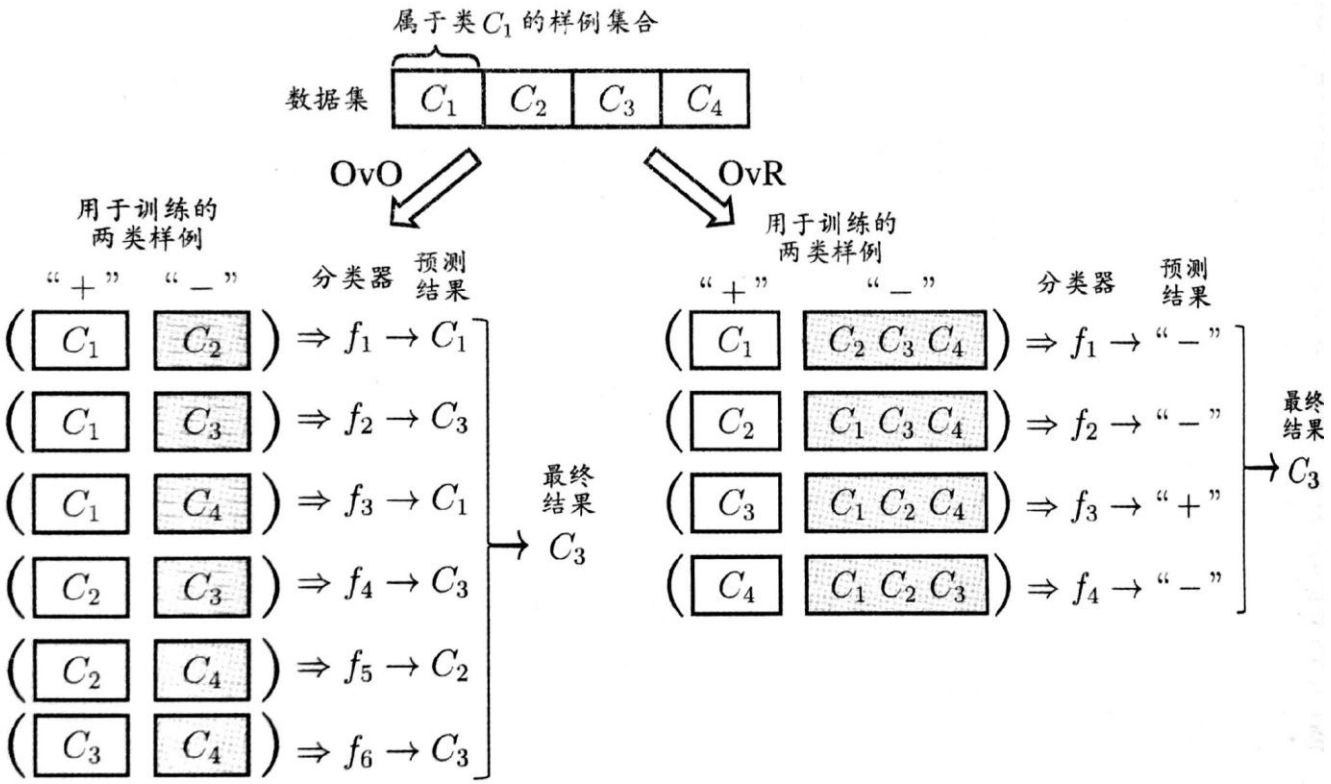
SVM、Logistic 回归都是2分类的分类器，也就是0、1分类。对于多个类别，比如 K-Means 所处理的 K 个类别的情况，如何处理呢？

两种思路：

One v.s. One: 对所有类别，两两训练分类器，n 个类别的情况下，会需要 $\frac{n(n-1)}{2}$ 个分类器，最后对所有预测结果排序，得分最高的类别胜出

One v.s. Others: 每个类别训练一个分类器，也就是属于类别 C_i 或者其它

图例


属于类C_1的样例集合
数据集
C_1 C_2 C_3 C_4
OvO
OvR
用于训练的两类样例
“+” “−”
分类器
预测结果
$(C_1\ C_2) \Rightarrow f_1 \rightarrow C_1$
$(C_1\ C_3) \Rightarrow f_2 \rightarrow C_3$
$(C_1\ C_4) \Rightarrow f_3 \rightarrow C_1$
$(C_2\ C_3) \Rightarrow f_4 \rightarrow C_3$
$(C_2\ C_4) \Rightarrow f_5 \rightarrow C_2$
$(C_3\ C_4) \Rightarrow f_6 \rightarrow C_3$
最终结果
C_3
用于训练的两类样例
“+” “−”
分类器
预测结果
$(C_1\ C_2\ C_3\ C_4) \Rightarrow f_1 \rightarrow$ “−”
$(C_2\ C_1\ C_3\ C_4) \Rightarrow f_2 \rightarrow$ “−”
$(C_3\ C_1\ C_2\ C_4) \Rightarrow f_3 \rightarrow$ “+”
$(C_4\ C_1\ C_2\ C_3) \Rightarrow f_4 \rightarrow$ “−”
最终结果
C_3

OvO vs OvR

- OvO 训练集更多 $\frac{n(n-1)}{2}$ 个分类器，而OvR 只需要 N 个分类器，因此OvO的存储和测试的开销比较大
- OvR 每个训练集都包含了全部训练数据，因此训练的开销比较大
- 多数情况下，两者的总体开销差不多

疑问

- 问题答疑： http://www.xxwenda.com/
  - 可邀请老师或者其他人回答问题

联系我们

小象学院：互联网新技术在线教育领航者

- 微信公众号：大数据分析挖掘
- 新浪微博：ChinaHadoop