

第三章 文本分类

- 1.1 分类问题和文本分类过程
- 1.2 常用分类算法
- 1.3 具体实现案例：新闻语料分类

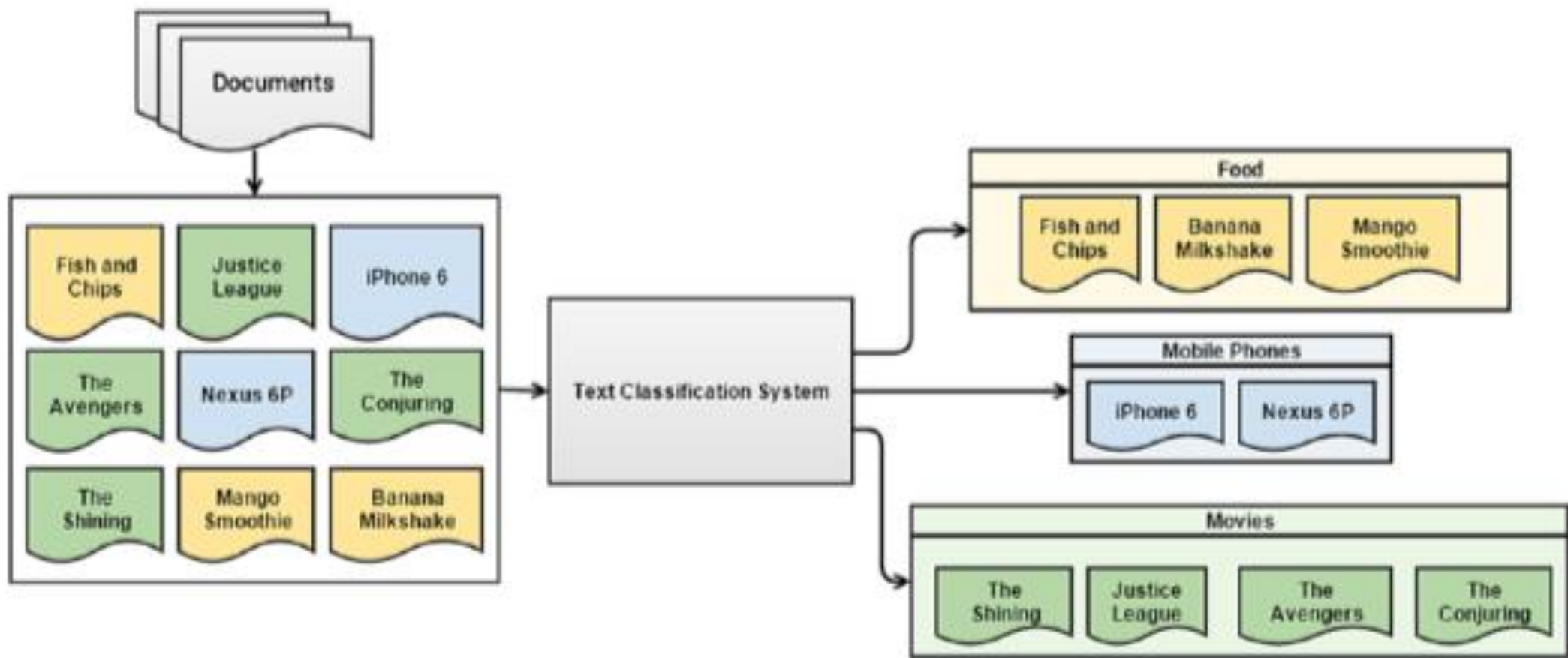
文本分类

- 文本分类有时又叫文档分类。
- 在这里，每个文档可以是语料库、博客、评论、论文等网上或数据库里面的标题（句子）、一条评论（段落）、或者完整的文档。
- 这个任务涉及把文本文档归到不同的类别，每个类别的文档具有某些共同特征，或者同一个主题。
- 在分类问题中，类别个数和每个类别是预先给定的。

注意：

- 广义上讲，文档包括用声音、文字、图画等方式记录的信息。在这里，“文档”指文本文档。
- 分类，有时被叫成“归类”。

文本分类系统



每个文档被分到预先给定的类别中的一个。

文本分类应用

- 新闻归类：提前设定好类别，如政治、体育、娱乐、科技等。
- 垃圾邮件检测：垃圾邮件、正常邮件
- 情感分类：正面、负面

不同分类问题

根据数据、类别数目不同，有以下几种分类问题：

- 二分类

分到两个类中的一个，“是”和“否”问题。

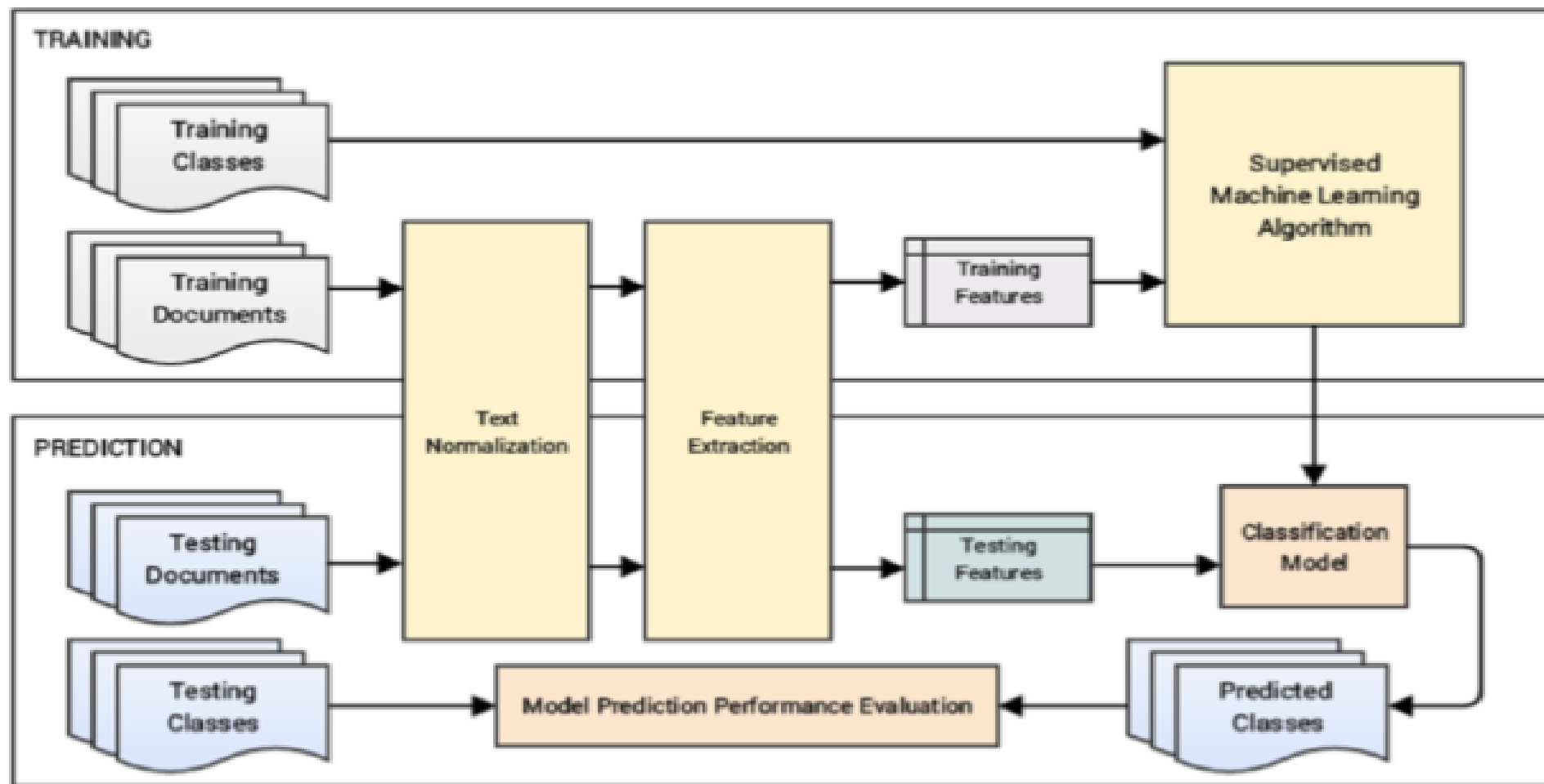
- 多分类

分到多个类别中的一个，是对二分类的一个扩展。

- 多标签分类

- 每个样本可以分到多个类别，如一个新闻既和政治有关，也和科技有关。

文本分类基本过程



当我们拿到训练集、测试集、验证集时，三个数据集用统一的预处理和文本表示模块，从而得到的特征表示具有一致性。

分类模型

- 朴素贝叶斯
- 随机森林
- 逻辑回归
- 支持向量机

朴素贝叶斯

回顾:

- 贝叶斯定理
- 朴素贝叶斯算法
- 拉普拉斯修正

朴素贝叶斯分类器 (naïve Bayes classifier)

$$P(c | \mathbf{x}) = \frac{P(c) P(\mathbf{x} | c)}{P(\mathbf{x})}$$

主要障碍：所有属性上的联合概率难以从有限训练样本估计获得，存在样本稀疏问题，类似“维度灾难”。

基本思路：假定对于给定的类属性之间相互独立？

$$P(c | \mathbf{x}) = \frac{P(c) P(\mathbf{x} | c)}{P(\mathbf{x})} = \frac{P(c)}{P(\mathbf{x})} \prod_{i=1}^d P(x_i | c)$$

d 为属性数， x_i 为 \mathbf{x} 在第 i 个属性上的取值

$P(\mathbf{x})$ 对所有类别相同，于是根据以下公式来预测类别

$$h_{nb}(\mathbf{x}) = \arg \max_{c \in \mathcal{Y}} P(c) \prod_{i=1}^d P(x_i | c)$$

朴素贝叶斯分类器

□ 估计 $P(c)$: $P(c) = \frac{|D_c|}{|D|}$

□ 估计 $P(\mathbf{x}|c)$:

- 对离散属性, 令 D_{c,x_i} 表示 D_c 中在第 i 个属性上取值为 x_i 的样本集合, 则

$$P(x_i|c) = \frac{|D_{c,x_i}|}{|D_c|}$$

- 对连续属性, 考虑概率密度函数, 假定 $p(x_i|c) \sim \mathcal{N}(\mu_{c,i}, \sigma_{c,i}^2)$

$$p(x_i | c) = \frac{1}{\sqrt{2\pi}\sigma_{c,i}} \exp\left(-\frac{(x_i - \mu_{c,i})^2}{2\sigma_{c,i}^2}\right)$$

□ 得到类别

$$h_{nb}(\mathbf{x}) = \arg \max_{c \in \mathcal{Y}} P(c) \prod_{i=1}^d P(x_i | c)$$

拉普拉斯修正 (Laplacian correction)

若某个属性值在训练集中没有与某个类同时出现过，则直接计算会出现问题，因为概率连乘将“抹去”其他属性提供的信息

例如，若训练集中未出现“敲声=清脆”的好瓜，则模型在遇到“敲声=清脆”的测试样本时

令 C 表示训练集 D 中可能的类别数， N_i 表示第 i 个属性可能的取值数

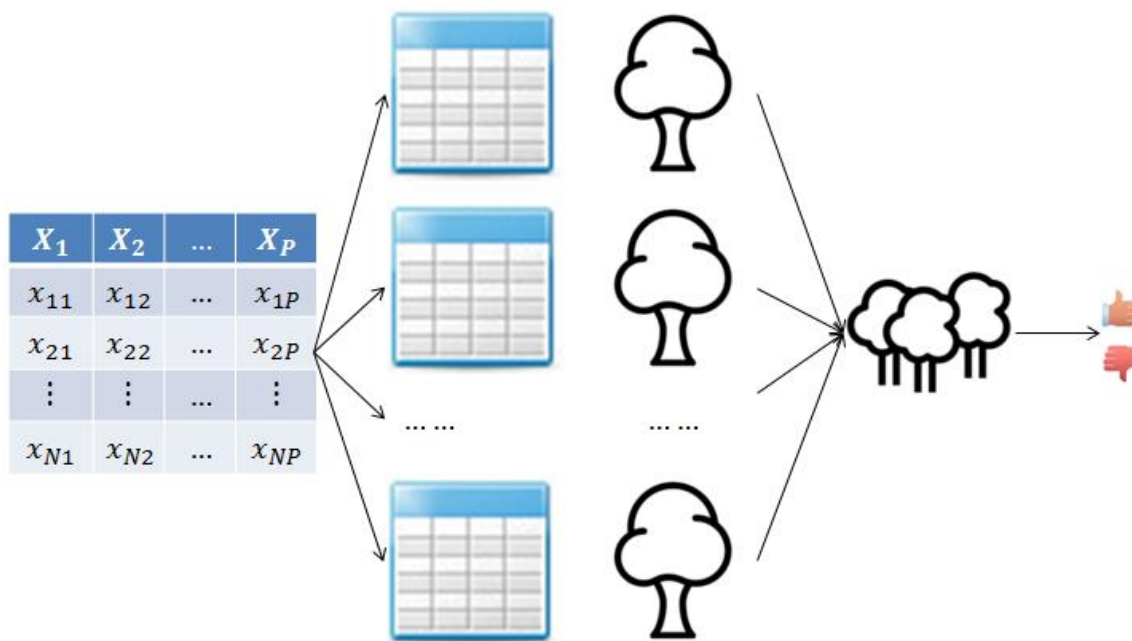
$$\hat{P}(c) = \frac{|D_c|+1}{|D|+C}, \quad \hat{P}(x_i | c) = \frac{|D_{c,x_i}|+1}{|D_c|+N_i}$$

假设了属性值与类别的均匀分布，这是额外引入的 **bias**

组合分类

- 并行化方法: 装袋 (**Bagging**)
 - 随机森林 (Random Forest)
 - 随机子空间 (Random Subspace)
 -
- 序列化方法: 提升 (**Boosting**)
 - **AdaBoost**
 - GradientBoost
 - LPBoost
 -

Bagging实例：随机森林



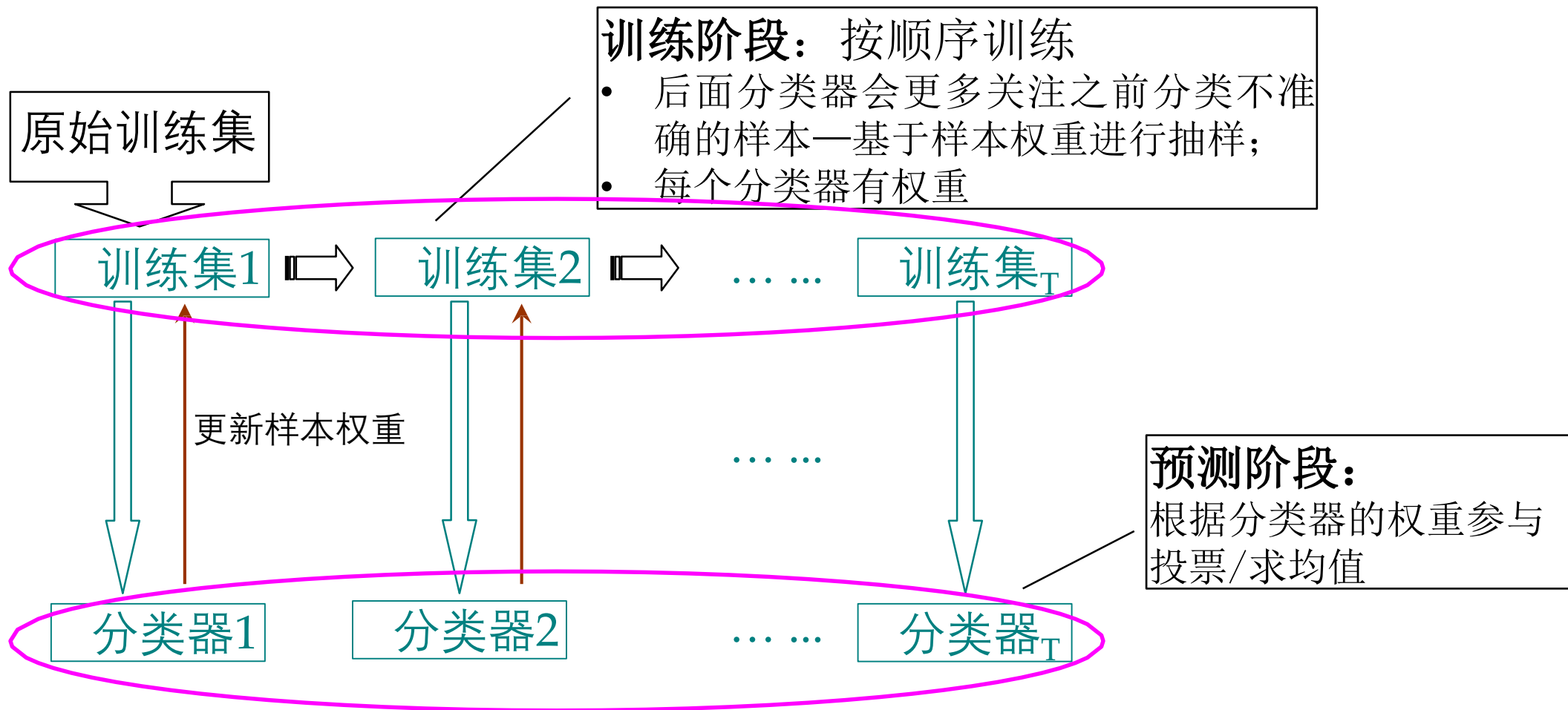
随机森林：决策树为基分类器的组合分类

1. 利用Bootstrap（有放回采样），从原始数据集中生成 k 个数据集，每个数据集都含有 N 个观测和 P 特征（即与给定数据集规模一样）。

2. 让每棵决策树尽可能地充分生长，使得树中的每个节点尽可能“纯净”，即随机森林中的每棵子树都不进行剪枝。

3. 针对 k 棵CART树的随机森林，对分类问题利用投票法，将最高得票的类别用于最终的判断结果；对回归问题利用均值法，将其用作预测样本的最终结果。

Boosting: 框架



AdaBoost (Adaptive Boost)

- 样本权重 w_i , 越大越可能出现在下一个分类器的训练集

样本权重怎么调整?

- ✓ 所有样本权重之和始终为1, 且初始样本权重相等 $w_i = \frac{1}{n}$;
- ✓ 若当前分类器对该样本分类正确, 则用以下公式减小权重, 并重新归一化 $\sum_{i=1}^n w_i = 1$ 。

$$w_{i-new} = w_{i-old} * \frac{err(C_t)}{1 - err(C_t)}$$

- 分类器错误率 $err(C_t)$ (若 $err(C_t) > 0.5$, 则丢弃)

$$err(C_t) = \sum_{y'_i \neq y_i} w_i, \quad (y'_i \neq y_i \text{ 表示预测的标签不正确})$$

- 分类器权重 g_t

权重与错误率成反比, 表示其在投票是的重要性

$$g_t = \log \frac{1 - err(C_t)}{err(C_t)}$$

逻辑回归(logistic regression)

- 也叫对数几率回归或对率回归
- 是一种广义线性模型：将线性回归输出通过对率函数映射到 $[0,1]$ 。
- 是一种“分类”方法，而不是“回归”。

线性模型(linear model)

线性模型试图学得一个通过属性的线性组合来进行预测的函数

$$f_{\mathbf{w},b}(\mathbf{x}) = w_1x_1 + w_2x_2 + \cdots w_dx_d + b$$

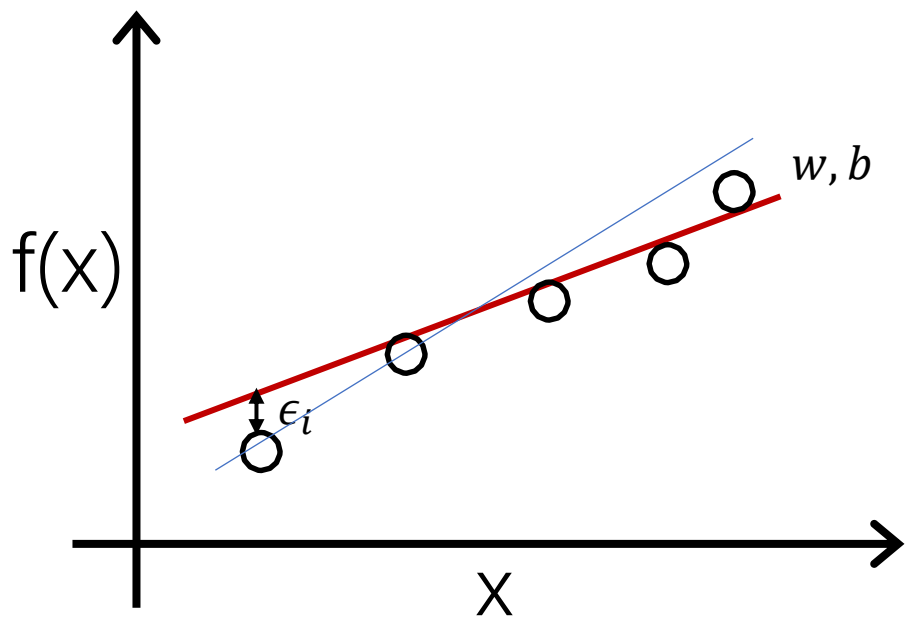
向量形式: $f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$

简单、基本、可理解性好

其中 $\mathbf{x} = (x_1, x_2, \dots x_d)^T$, $\mathbf{w} = (w_1, w_2, \dots w_d)^T$

如果只有一个变量/属性 x , 则为一元线性回归:

$$f_{w,b}(x) = wx + b$$



$$f_{w,b}(x) = wx + b$$

如何确定 w, b 使得对应模型的拟合能力最好?

$$f_{w,b}(x) \cong y$$

怎么衡量回归模型拟合能力?

$$E_{w,b} = \sum_{i=1}^n (f(x_i) - y_i)^2 = \sum_{i=1}^n \epsilon_i^2$$

或者 $E_{w,b} = \frac{1}{2n} \sum_{i=1}^n (f(x_i) - y_i)^2$

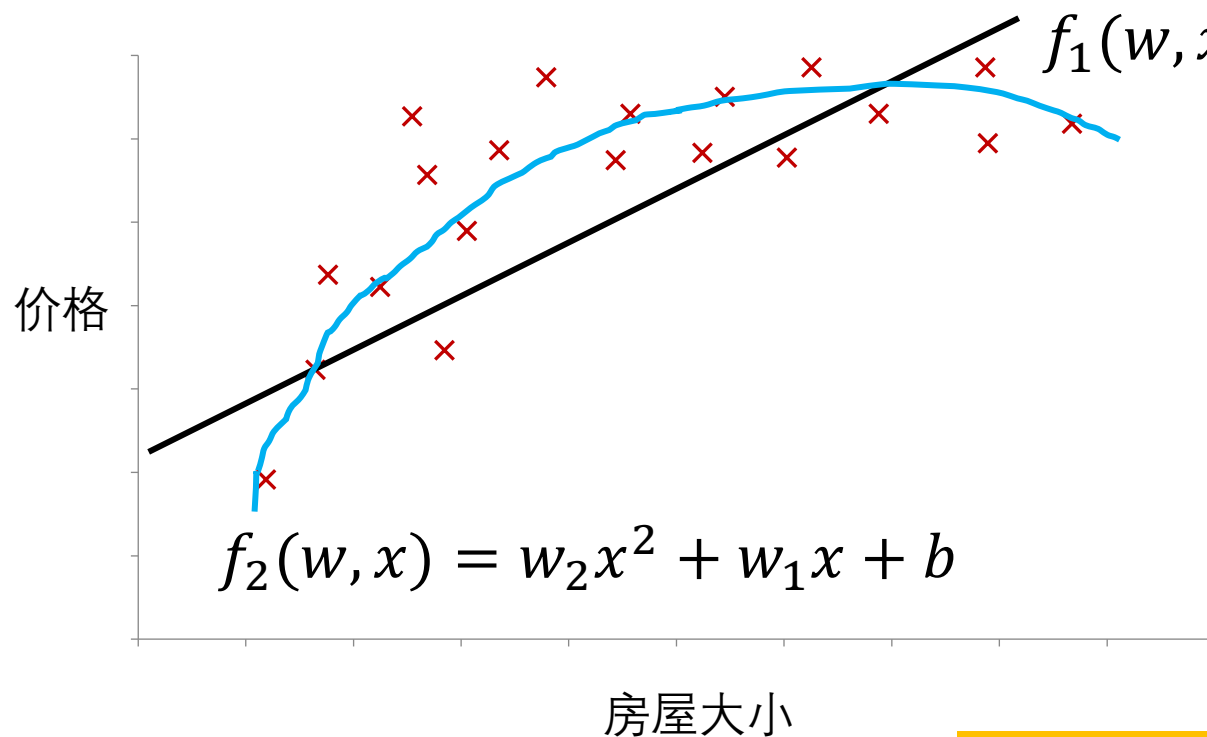
求解使得以上损失最小化时的 w, b



$$(w^*, b^*) = \operatorname{argmin} E_{w,b}$$

线性模型的拟合能力

只能产生线性拟合？通过特征变化可以得到非线性模型。



令 $x_2 = x^2$, 即大小的平方;
 $x_1 = x$, 即大小;



$$f_2(w, x) = w_2x_2 + w_1x_1 + b$$

(一元二次) 多项式回归转成 (二元) 线性回归

二分类任务

二分类问题的输出 $y \in \{0, 1\}$

反例

正例

找 z 和 y 的联系函数

线性回归模型产生的实值输出 $z = \mathbf{w}^T \mathbf{x} + b$

理想的“单位阶跃函数”
(unit-step function)

$$y = \begin{cases} 0, & z < 0; \\ 0.5, & z = 0; \\ 1, & z > 0, \end{cases}$$

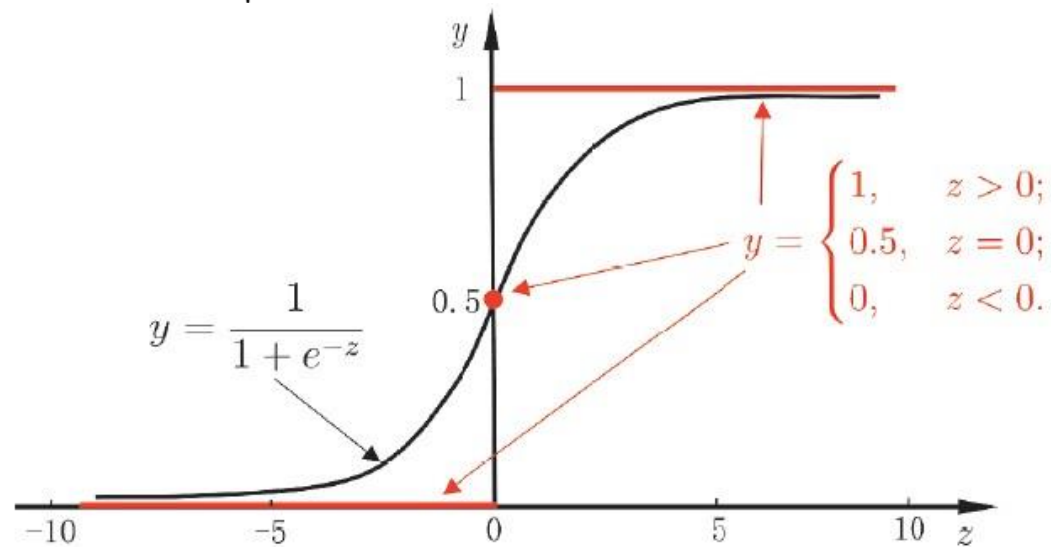
性质不好,
需找“替代函数”
(surrogate function)

常用 单调可微、任意阶可导

$$y = \frac{1}{1 + e^{-z}}$$

对数几率函数
(logistic function)
简称“对率函数”

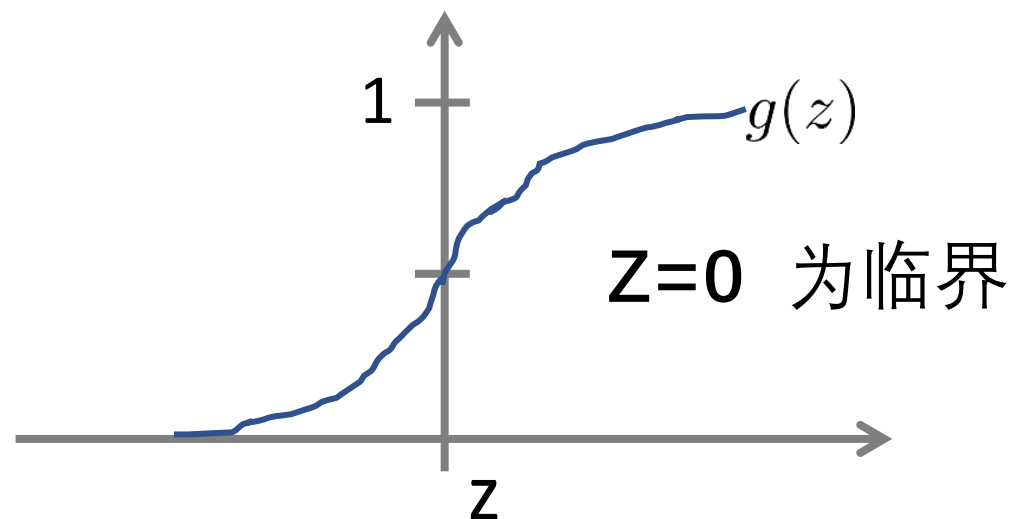
又称 Sigmoid function



怎么用对率回归进行预测

$$z = \mathbf{w}^T \mathbf{x} + b$$

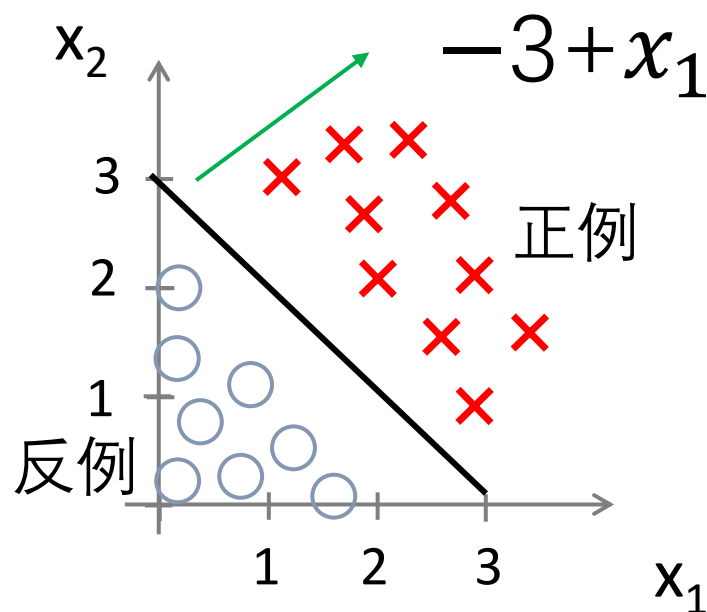
$$y = g(z) = \frac{1}{1 + e^{-z}}$$



假如 $y \geq 0.5$ 判断为正例, 对应要求 $\mathbf{w}^T \mathbf{x} + b \geq 0$

$y < 0.5$ 判断为反例, 对应要求 $\mathbf{w}^T \mathbf{x} + b < 0$

决策边界 (Decision Boundary)



$$-3 + x_1 + x_2 = 0$$

$$z = w_2 x_2 + w_1 x_1 + b$$

假设通过训练后得到:

$$w_2 = 1, \quad w_1 = 1, \quad b = -3$$

代入后得到预测为正例的条件 $-3 + x_1 + x_2 \geq 0$

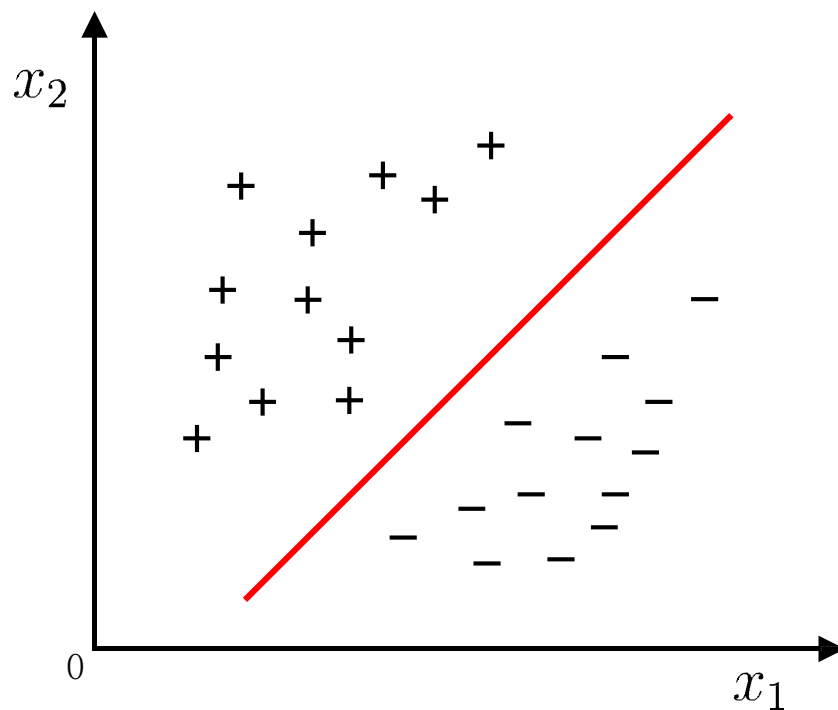
预测为反例的条件 $-3 + x_1 + x_2 < 0$

支持向量机 (Support Vector Machine)

- 最大化间隔的分类器
- 训练好的模型至与支持向量有关

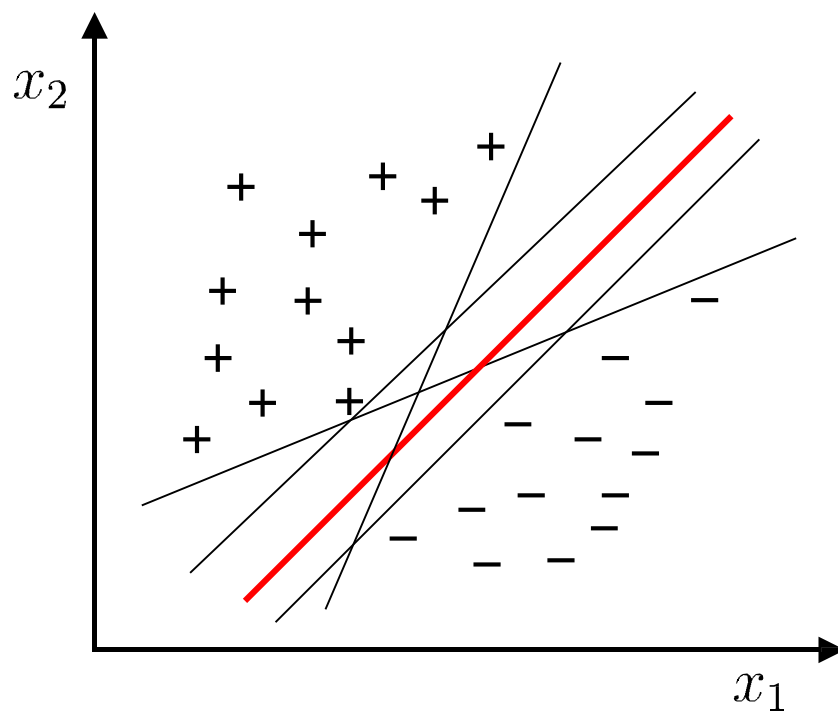
线性分类器回顾

在样本空间中寻找一个超平面, 将不同类别的样本分开



线性分类器回顾

将训练样本分开的超平面可能有很多, 哪一个更好呢?



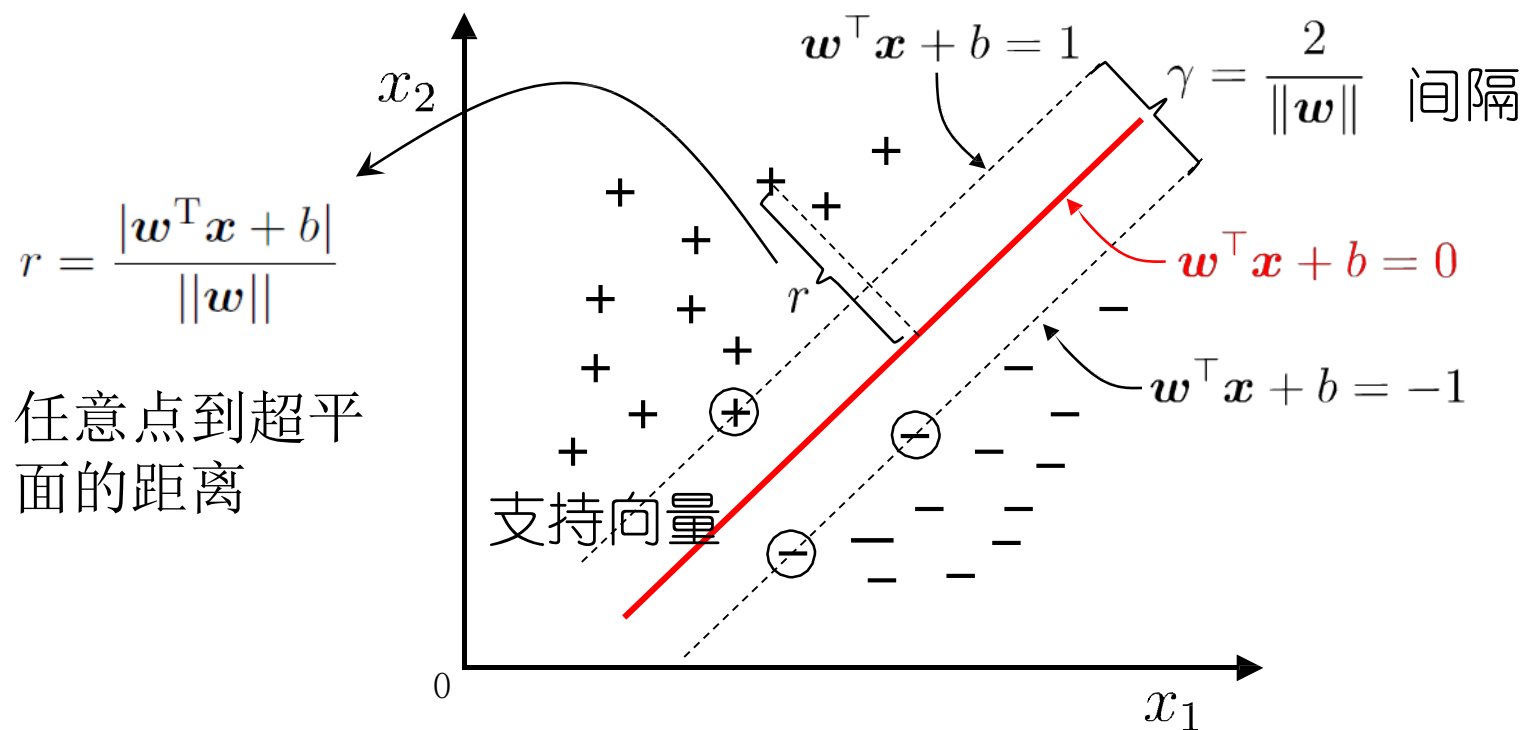
“正中间”的:

鲁棒性最好, 泛化能力最强

间隔(margin)与支持向量(support vector)

超平面方程: $\mathbf{w}^\top \mathbf{x} + b = 0$

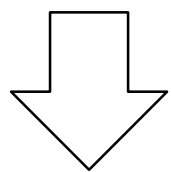
SVM分类:
正例 $y = 1$: $\mathbf{w}^\top \mathbf{x} + b \geq 1$
反例 $y = -1$: $\mathbf{w}^\top \mathbf{x} + b \leq -1$



支持向量机基本型

最大间隔：寻找参数 \mathbf{w} 和 b ，使得 γ 最大

$$\begin{aligned} \arg \max_{\mathbf{w}, b} \quad & \frac{2}{\|\mathbf{w}\|} \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \dots, m. \end{aligned}$$



其中 $y_i \in \{-1, 1\}$

$$\begin{aligned} \arg \min_{\mathbf{w}, b} \quad & \frac{1}{2} \|\mathbf{w}\|^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \dots, m. \end{aligned}$$

凸二次规划问题，能用优化计算包求解，但可以有更高效的办法

对偶问题

拉格朗日乘子法

□ 第一步：引入拉格朗日乘子 $\alpha_i \geq 0$ 得到拉格朗日函数

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{i=1}^m \alpha_i (1 - y_i(\mathbf{w}^T \mathbf{x}_i + b))$$

□ 第二步：令 $L(\mathbf{w}, b, \boldsymbol{\alpha})$ 对 \mathbf{w} 和 b 的偏导为零可得

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i, \quad 0 = \sum_{i=1}^m \alpha_i y_i$$

□ 第三步：把以上两式代入拉格朗日函数得

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{s.t.} \quad & \sum_{i=1}^m \alpha_i y_i = 0, \quad \alpha_i \geq 0, \quad i = 1, 2, \dots, m \end{aligned}$$

对偶问题具有闭式解，
比用数值优化求解基本型
中的 \mathbf{w} 和 b 更加高效。

解的稀疏性

KKT条件:

$$\begin{cases} \alpha_i \geq 0, \\ y_i f(\mathbf{x}_i) \geq 1, \\ \alpha_i (y_i f(\mathbf{x}_i) - 1) = 0. \end{cases} \quad \Longrightarrow \quad \begin{aligned} &\text{对所有 } i \text{ 满足 } \alpha_i = 0 \text{ 或 } y_i f(\mathbf{x}_i) = 1 \\ &\text{即只有支持向量 } i \in S \text{ 的 } \alpha_i > 0 \end{aligned}$$

解的**稀疏性**: 训练完成后, 最终模型仅与支持向量有关

即 $\mathbf{w} = \sum_{i \in S} \alpha_i y_i \mathbf{x}_i^T$

根据训练集中任意支持向量满足 $y_s f(\mathbf{x}_s) = 1$ 即

$$y_s (\sum_{i \in S} \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_s + b) = 1 \quad \Longrightarrow \quad b = \frac{1}{y_s} - \sum_{i \in S} \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_s$$

训练结束后只需保留支持向量用于预测: $f(\mathbf{x}) = \sum_{i \in S} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$

评估度量

- 混淆矩阵 (confusion matrix)
- 准确率/精度 Accuracy
- 查准率 Precision
- 查全率/召回率 Recall
- F1-score

预处理

	Article
0	my brother is in a market for
1	back in high school i worked as a lab assistant
2	
3	when I came from the place

```
corpus=['my brother is in a market for',  
        'back in high school i worked as a lab assistant',  
        '',  
        'when I came from the place']  
data_df = pd.DataFrame({'Article': corpus})
```



去掉原始空文档，然后调用normalize_corpus(...)函数得到经过一系列预处理之后的文本数据集

```
data_df = data_df[~(data_df.Article.str.strip() == '')]  
norm_corpus=normalize_corpus(data_df['Article'])  
data_df['Clean Article'] = norm_corpus
```



	Article	Clean Article
0	my brother is in a market for	brother market
1	back in high school i worked as a lab assistant	back high school work lab assist
3	when I came from the place	I came place

文本数据集：[20 Newsgroups](#)

- 该数据集包含大约 18,000 个来自20个不同主题类别的新闻。
- 一般来讲，类别数越多，任务难度越大。

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

文本数据集：[20 Newsgroups](#)

从scikit learn里面用fetch_20newsgroups加载：

设置Scikit-Learn 加载参数“remove”去掉email的header/footer/quote信息

```
from sklearn.datasets import fetch_20newsgroups
data = fetch_20newsgroups(subset='all', shuffle=True, remove=('headers', 'footers', 'quotes'))
data_labels_map = dict(enumerate(data.target_names))
corpus, target_labels, target_names = (data.data, data.target, [data_labels_map[label] for label in data.target])
data_df = pd.DataFrame({'Article': corpus, 'Target Label': target_labels, 'Target Name': target_names})
```

	Article	Target Label	Target Name
0	\n\nI am sure some bashers of Pens fans are pr...	10	rec.sport.hockey
1	My brother is in the market for a high-perform...	3	comp.sys.ibm.pc.hardware
2	\n\n\n\n\nFinally you said what you dream abou...	17	talk.politics.mideast
3	\nThink!\n\nIt's the SCSI card doing the DMA t...	3	comp.sys.ibm.pc.hardware
4	1) I have an old Jasmine drive which I cann...	4	comp.sys.mac.hardware

注意：
在线下载有问题时，建议手动下载数据集。具体参考该章最后一页ppt。

预处理

- 用normalize_corpus(...)函数来得到经过一系列预处理之后的文本数据集。

```
#过滤空文档
data_df = data_df[~(data_df.Article.str.strip() == '')]
norm_corpus=normalize_corpus(data_df['Article'])
data_df['Clean Article'] = norm_corpus
data_df = data_df[['Article', 'Clean Article', 'Target Label', 'Target Name']]
data_df.head()
```

	Article	Clean Article	Target Label	Target Name
0	\n\nI am sure some bashers of Pens fans are pr...	sure basher pen fan pretti confus lack ani kin...	10	rec.sport.hockey
1	My brother is in the market for a high-perform...	brother market high-perform video card support...	3	comp.sys.ibm.pc.hardware
2	\n\n\n\n\nFinally you said what you dream abou...		 ; final said dream about. mediterranean?? ...	17	talk.politics.mideast
3	\nThink!\n\nIt's the SCSI card doing the DMA t...	think ! ' scsi card dma transfer disk ... scsi...	3	comp.sys.ibm.pc.hardware
4	1) I have an old Jasmine drive which I cann...	1) old jasmin drive cannot use new system. un...	4	comp.sys.mac.hardware

```

import nltk
import spacy
from nltk.tokenize.toktok import ToktokTokenizer
tokenizer = ToktokTokenizer()
#nltk默认英语停用词表
stopword_list =nltk.corpus.stopwords.words('english')
nlp = spacy.load('en_core_web_sm', parse=True, tag=True, entity=True)

```

定义Spacy模型nlp;

```

def normalize_corpus(corpus,
    text_stemming=True,
    text_lemmatization=False,
    stopwords_removal=True, text_lower_case=False):
    normalized_corpus = []
    # 对数据集中的每个文档进行预处理
    for doc in corpus:
        #词干提取
        if text_stemming:
            doc=stem_text(doc)
        # 词性还原
        if text_lemmatization:
            doc = lemmatize_text(doc)
        # 去停用词
        if stopwords_removal:
            doc = remove_stopwords(doc, is_lower_case=text_lower_case)
        normalized_corpus.append(doc)
    return normalized_corpus

```

默认进行词干提取、
去停用词。

```
def stem_text(text):
    ps = nltk.porter.PorterStemmer()
    tokenizer = nltk.tokenize.toktok.ToktokTokenizer()
    tokens = tokenizer.tokenize(text)
    stem_token = [ps.stem(token.strip()) for token in tokens]
    text = ' '.join(stem_token)
    return text
```


用刚才定义的pacy模型nlp()对输入进行分析。对text的每个token调用.lemma_方法得到结果。

```
def lemmatize_text(text):
    text = nlp(text)
    lemma_word = [word.lemma_ if word.lemma_ != '-PRON-' else word.text for word in text]
    text = ' '.join(lemma_word)
    return text
```

```
def remove_stopwords(text, is_lower_case=False):
    tokens = tokenizer.tokenize(text)
    # .strip()方法去掉字符串头尾多余的空格、\t, \n等字符。
    tokens = [token.strip() for token in tokens]
    if is_lower_case:
        filtered_tokens = [token for token in tokens if token not in stopwords_list]
    else:
        filtered_tokens = [token for token in tokens if token.lower() not in stopwords_list]
    filtered_text = ' '.join(filtered_tokens)
    return filtered_text
```

预处理

```
▶ import numpy as np
data_df = data_df[['Article', 'Clean Article', 'Target Label', 'Target Name']]
data_df = data_df.replace(r' ^(\s?)+$', np.nan, regex=True)
data_df = data_df.dropna().reset_index(drop=True)
data_df.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18330 entries, 0 to 18329
Data columns (total 4 columns):
Article          18330 non-null object
Clean Article     18330 non-null object
Target Label     18330 non-null int64
Target Name      18330 non-null object
dtypes: int64(1), object(3)
memory usage: 572.9+ KB
```

经过预处理，又会出现一些文档为空，把这些空文档删除。

#可以用下面代码把预处理后的数据保存，下次直接拿预处理好的数据。

```
data_df.to_csv('D:\Anaconda3\Lib\site-packages\sklearn\datasets\data\clean_newsgroups.csv', index=False)
```

分割训练集和测试集

```
import numpy as np
from sklearn.model_selection import train_test_split
train_corpus, test_corpus, train_label_nums, test_label_nums, train_label_names, \
test_label_names = train_test_split(np.array(data_df['Clean Article']),
                                    np.array(data_df['Target Label']),
                                    np.array(data_df['Target Name']),
                                    test_size=0.33, random_state=42)
```

因为真实应用中，测试集是模型部署的时候才有，所以先分割再进行向量表示。预处理和分割的先后关系应该不影响结果，否则先分割再预处理。

文档表示

- 词袋：基于词的 N -gram的词频、TF-IDF
- Word2Vec

实现： 文本分类-模型训练和评估

文档表示成词频向量

```
from sklearn.feature_extraction.text import CountVectorizer
# 对训练集建立词袋模型
cv = CountVectorizer(min_df=0.0, max_df=1.0)
cv_train_features = cv.fit_transform(train_corpus)
# 把测试集转成向量表示
cv_test_features = cv.transform(test_corpus)
```

如果在测试集中出现训练集里面没有的词会怎么样?

5折交叉验证朴素贝叶斯分类结果

```
▶ from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import cross_val_score
mnb = MultinomialNB(alpha=1)
mnb.fit(cv_train_features, train_label_names)
mnbBow_cv_scores = cross_val_score(mnb, cv_train_features, train_label_names, cv=5)
mnbBow_cv_mean_score = np.mean(mnbBow_cv_scores)
print('CV Accuracy (5-fold):', mnbBow_cv_scores)
print('Mean CV Accuracy:', mnbBow_cv_mean_score)
mnbBow_test_score = mnb.score(cv_test_features, test_label_names)
print('Test Accuracy:', mnbBow_test_score)
```

参数alpha: 拉普拉斯修正的权重系数,
alpha=0则没有修正。

```
CV Accuracy (5-fold): [0.65827922 0.63998375 0.64454397 0.65089723 0.65849673]
Mean CV Accuracy: 0.6504401799889876
Test Accuracy: 0.6688708877500413
```

Scikit-Learn 库中的[MultinomialNB](#)实现多分类朴素贝叶斯

5折交叉验证朴素贝叶斯分类结果

- 把词频权重改成tfidf

```
from sklearn.naive_bayes import MultinomialNB
mnb = MultinomialNB(alpha=1)
mnb.fit(tv_train_features, train_label_names)
mnb_bow_tv_scores = cross_val_score(mnb, tv_train_features, train_label_names, cv=5)
mnb_bow_tv_mean_score = np.mean(mnb_bow_tv_scores)
print('TV Accuracy (5-fold):', mnb_bow_tv_scores)
print('Mean TV Accuracy:', mnb_bow_tv_mean_score)
mnb_bow_test_score = mnb.score(tv_test_features, test_label_names)
print('Test Accuracy:', mnb_bow_test_score)
```

TV Accuracy (5-fold): [0.7049513 0.70621698 0.7007329 0.71207178 0.72344771]

Mean TV Accuracy: 0.7094841346496311

Test Accuracy: 0.7184658621259712

测试集准确率提升5%!

调参

对tfidf_ngram_range和mnf_alpha这两个参数进行不同组合的参数设置

```
# 对朴素贝叶斯分类器进行调参
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import TfidfVectorizer
mnf_pipeline = Pipeline([('tfidf', TfidfVectorizer()), ('mnf', MultinomialNB()) ])
param_grid = {'tfidf_ngram_range': [(1, 1), (1, 2)], 'mnf_alpha': [1e-5, 1e-4, 1e-2, 1e-1, 1] }
gs_mnf = GridSearchCV(mnf_pipeline, param_grid, cv=5, verbose=2)
gs_mnf = gs_mnf.fit(train_corpus, train_label_names)
```

GridSearch 网格搜索

查看参数搜索结果

```
cv_results = gs_mnb.cv_results_  
results_df = pd.DataFrame({'rank': cv_results['rank_test_score'], 'params': cv_results['params'], \  
                           'cv score (mean)': cv_results['mean_test_score'], 'cv score (std)': cv_results['std_test_score']})  
results_df = results_df.sort_values(by=['rank'], ascending=True)  
pd.set_option('display.max_colwidth', 100)  
results_df
```

	rank	params	cv score (mean)	cv score (std)
5	1	{'mnf__alpha': 0.01, 'tfidf__ngram_range': (1, 2)}	0.764433	0.006358
4	2	{'mnf__alpha': 0.01, 'tfidf__ngram_range': (1, 1)}	0.761909	0.005188
6	3	{'mnf__alpha': 0.1, 'tfidf__ngram_range': (1, 1)}	0.750916	0.008254
7	4	{'mnf__alpha': 0.1, 'tfidf__ngram_range': (1, 2)}	0.750346	0.008194
3	5	{'mnf__alpha': 0.0001, 'tfidf__ngram_range': (1, 2)}	0.748066	0.007650
1	6	{'mnf__alpha': 1e-05, 'tfidf__ngram_range': (1, 2)}	0.740331	0.006917
2	7	{'mnf__alpha': 0.0001, 'tfidf__ngram_range': (1, 1)}	0.733572	0.005754
0	8	{'mnf__alpha': 1e-05, 'tfidf__ngram_range': (1, 1)}	0.719893	0.006279
8	9	{'mnf__alpha': 1, 'tfidf__ngram_range': (1, 1)}	0.709877	0.007741
9	10	{'mnf__alpha': 1, 'tfidf__ngram_range': (1, 2)}	0.707679	0.011376

mnf__alpha
=0.01得到比较
好的结果，比
mnf__alpha=1时
又提升5%!

对分类结果做进一步定性分析

除了量化评估，深入到具体实例，进行不同角度的定性分析有助于对分类结果进行解释和分析。

加载自定义函数model_evaluation_utils
(与章节其他代码放在同一目录下)

```
import model_evaluation_utils as meu
```

结果分析和解释-混淆矩阵

Predicted:																					
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Actual:	0	158	2	2	0	0	0	0	3	0	1	2	4	1	2	7	40	10	9	3	8
	1	1	236	8	13	5	20	4	2	2	1	3	3	3	5	6	1	1	0	1	0
	2	0	18	211	34	7	24	3	1	2	0	0	6	5	2	2	2	2	0	1	0
	3	1	9	29	238	21	5	7	3	0	0	0	3	9	1	0	1	0	0	1	0
	4	0	4	5	19	224	2	6	1	0	0	0	5	7	2	1	1	1	0	0	0
	5	0	26	9	2	3	252	1	0	1	0	1	2	1	0	3	2	0	0	0	0
	6	0	2	2	24	11	1	251	12	2	3	0	4	9	4	3	1	1	1	1	0
	7	4	4	1	0	4	1	4	247	17	1	1	4	7	3	0	6	7	0	3	0
	8	2	0	0	1	2	1	2	15	258	2	1	3	2	4	2	4	8	2	6	1
	9	1	2	2	0	0	3	1	0	3	290	13	1	2	2	0	1	7	1	2	0
	10	0	0	1	0	2	4	1	1	3	5	288	0	2	1	0	3	1	1	1	0
	11	1	6	2	3	1	3	1	1	1	1	1	279	0	4	4	3	5	4	1	2
	12	1	10	3	21	7	2	8	10	2	0	0	11	212	2	12	3	1	0	0	1
	13	2	5	0	1	1	0	0	2	0	1	1	1	2	251	5	9	0	1	2	0
	14	2	5	1	3	3	2	0	0	2	0	0	2	7	6	273	6	2	2	5	1
	15	6	3	1	0	0	1	0	0	0	1	1	2	2	7	2	318	2	3	1	1
	16	3	0	0	0	0	1	0	4	3	2	2	6	1	1	0	2	228	4	16	0
	17	6	1	0	0	1	2	0	2	2	0	2	1	0	1	3	12	2	261	4	0
	18	3	3	0	2	0	1	1	2	1	1	1	7	1	3	12	12	43	18	155	0
	19	19	1	0	0	0	1	0	2	0	0	2	2	0	5	5	87	23	6	10	58

	Label Name	Label Number
0	alt.atheism	0
1	comp.graphics	1
2	comp.os.ms-windows.misc	2
3	comp.sys.ibm.pc.hardware	3
4	comp.sys.mac.hardware	4
5	comp.windows.x	5
6	misc.forsale	6
7	rec.autos	7
8	rec.motorcycles	8
9	rec.sport.baseball	9
10	rec.sport.hockey	10
11	sci.crypt	11
12	sci.electronics	12
13	sci.med	13
14	sci.space	14
15	soc.religion.christian	15
16	talk.politics.guns	16
17	talk.politics.mideast	17
18	talk.politics.misc	18
19	talk.religion.misc	19

结果分析和解释

id为19的类别 (talk.religion.misc) 中confidence 最高的几个分错的实例

	Article	Clean Article	Target Label	Target Name	Predicted Name	Predicted Confidence
7068	\n\nI think the passage you're looking for is the following.\n\nMatthew 5:17 "Think not that I have come to abolish the law and the \nprophets; I have come not to abolish them but to fulfil the...	think passag ' look following. matthew 5 : 17 " think come abolish law prophet ; come abolish fulfil them. matthew 5 : 18 truli , say , till heaven earth pass away , iota , dot , pass law accompli...	19	talk.religion.misc	soc.religion.christian	1.000000
13789	: \n: I am a Mormon. I believe in Christ, that he is alive. He raised himself\n: [Text deleted]\n\n: I learned that the concept of the Holy Trinity was never taught by Jesus\n: Christ, that it ...	: : mormon. believ christ , alive. rais : [text delet] : : learn concept holi triniti wa never taught jesu : christ , wa " agre " council clergymen long christ : wa ascend , men author speak him...	19	talk.religion.misc	soc.religion.christian	0.999881
14539	iank@microsoft.com (Ian Kennedy) writes...\n\n\nMore along the lines of Hebrews 12:25-29, I reckon...\n\n\n\tSee that you refuse not him that speaks. For if they\n\n\tescaped not who refused him that ...	iank@microsoft.com (ian kennedi) write ... along line hebrew 12 : 25-29 , reckon ... 	 ; see refus speaks. 	 ; escap refus spake earth , much 	 ; shall escap , turn away 	 ; speak heaven...	19	talk.religion.misc	soc.religion.christian	0.999678
6512	Russell Turpin, as is his wont, has raised some interesting issues \n\nin his struggle with the Christian texts and the Christians. \n\nUnfortunately, he seems to be hoping for simplicity where it is ...	russel turpin , hi wont , ha rais interest issu hi struggl christian text christians. unfortun , seem hope simplic available. lukewarm stew detect may well inevit result divin mix bunch loser huma...	19	talk.religion.misc	soc.religion.christian	0.999641
16705	\nJesus did not say that he was the fulfillment of the Law, and, unless\nI'm mistaken, heaven and earth have not yet passed away. Am I mistaken?\n\nAnd, even assuming that one can just gloss over th...	jesu say wa fulfil law , , unless ' mistaken , heaven earth yet pass away. mistaken ? , even assum one gloss portion word jesu , realli think " accomplish ? " whi ' jesu say " ani jew annul ... " ...	19	talk.religion.misc	soc.religion.christian	0.999393

scikit learn其他分类模型

- 逻辑回归 LogisticRegression

```
from sklearn.linear_model import LogisticRegression
```

```
lr = LogisticRegression(penalty='l2', max_iter=100, C=1,  
random_state=42)
```

- SVM

```
from sklearn.svm import LinearSVC
```

```
svm = LinearSVC(penalty='l2', C=1, random_state=42)
```

组合方法:

```
from sklearn.ensemble import AdaBoostClassifier
```

```
from sklearn.ensemble import GradientBoostingClassifier
```


20newsgroups手动下载，转自[这里](#)

把钉钉群里面的20news-bydate.tar.gz下载后放到：
C:\Users\用户\scikit_learn_data\20news_home

注意：文件名中的.gz可能不显示，不要自己多加。

找到 C:\Users\用户\AppData\Roaming\Python\Python37\site-packages\sklearn\datasets文件夹找到
twenty_newsgroups.py文件，打开后注释

```
1. logger.info("Downloading dataset from %s (14 MB)", ARCHIVE.url)
2. archive_path = _fetch_remote(ARCHIVE, dirname=target_dir)
```

红色部分路径看具体情况。

并添加

```
1. archive_path = os.path.join(target_dir, r'20newsbydate.tar.gz')
```

运行以下代码（仍然需要几分钟）

```
from sklearn.datasets import fetch_20newsgroups
data = fetch_20newsgroups(subset='all', shuffle=True, remove=('headers', 'footers', 'quotes'))
```

成功后得到 C:\Users\用户\scikit_learn_data\ 20news-bydate_py3