## 备选方案：

### 朴素贝叶斯

使用多项式事件模型（multinomial event model）和Laplace平滑。

* 两种特征向量表达方式：
  + （1）向量的每一维表示筛选出的特征词典中的每个特征，以TF-IDF或其他权重表示该维的值；
  + （2）向量的长度等于文本中包含的特征数，每一维表示该特征在特征词典中的位置。

### 支持向量机SVM

* 选用多种核进行测试，比较效果：（1）线性核 （2）RBF核 （3）polynomial核 （4）sigmoid核 等
* 由于支持向量机对训练集正负数据比例非常敏感，因此在训练模型之前，可以先对训练集数据进行比例调整，对比例较大的一类数据进行随机抽样，保证训练集中两个类别的比例大约在1:1.

### Boosting vs. bagging

两者都属于集成学习方法，都是将已有的分类或回归算法通过一定方式组合起来，形成一个性能更加强大的分类器，即将弱分类器组装成强分类器。

**【Boosting】**

* 初始化时对每一个训练例赋相等的权重1／n，然后用该算法对训练集训练t轮，每一轮结束后，都提高那些在前一轮被弱分类器分错类别的样例的权值，减小前一轮分对类别的样例的权值，也就是让学习算法在后续的学习中集中对比较难的训练例进行学习，从而得到一个预测函数序列h。
* 预测效果好的预测函数权重较大，反之较小。**最终的预测函数H对分类问题采用有权重的投票方式，对回归问题采用加权平均的方法对新示例进行判别。**( 类似Bagging方法，但是训练是串行进行的，第k个分类器训练时关注对前k-1分类器中错分的文档，即不是随机取，而是加大取这些文档的概率；而bagging随机抽取样本，可以并行进行).
* 弱的分类器bias高，而强的分类器bias低，所以说boosting起到了**降低bias**的作用。variance不是boosting的主要考虑因素。

**【Bagging】**是bootstrap aggregating的缩写。

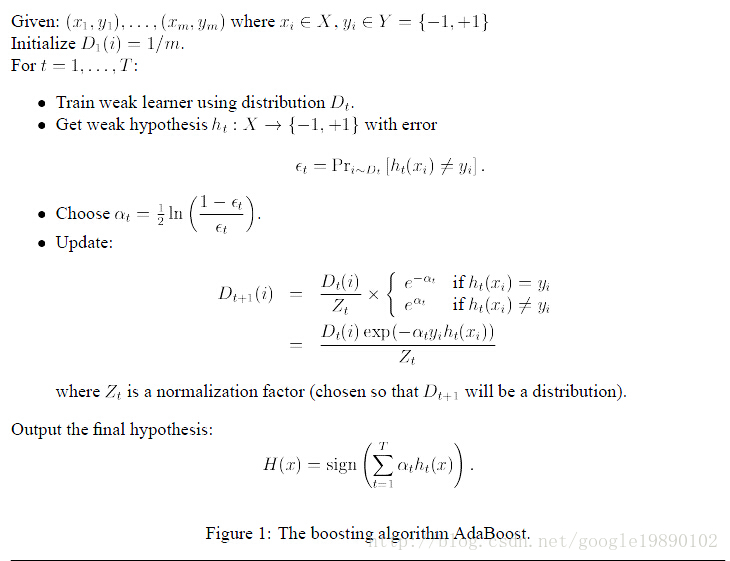
* 从原始样本集中抽取训练集。每轮从原始样本集中使用Bootstraping的方法抽取n个训练样本（一种有放回的抽样：在训练集中，有些样本可能被多次抽取到，而有些样本可能一次都没有被抽中）。共进行k轮抽取，得到k个训练集。（k个训练集之间是相互独立的）
* 每次使用一个训练集得到一个模型，k个训练集共得到k个模型。（注：这里并没有具体的分类算法或回归方法，我们可以根据具体问题采用不同的分类或回归方法，如决策树、感知器等）。
* **最终的预测函数H对分类问题采用投票方式，对回归问题采用简单平均方法对新示例进行判别。**
* 这种方式是对许多强（甚至过强）的分类器求平均。在这里，每个单独的分类器的bias都是低的，平均之后bias依然低；而每个单独的分类器都强到可能产生overfitting的程度，也就是variance高，求平均的操作起到的作用就是**降低这个variance**。

### Adaboost

AdaBoost是一种迭代型的算法，其核心思想是针对同一个训练集训练不同的分类器，即弱分类器，然后将这些弱分类器集合起来，构造一个强分类器。

在AdaBoost算法中，有两个权重：训练集中每个样本有一个权重，称为样本权重，用向量D表示；每一个弱学习算法具有一个权重，用向量α表示。初始时，设定每个样本的权重是相等的。依次利用每一个弱学习算法，对样本集进行学习，学习完成后进行错误率的统计，并利用错误率计算该弱学习算法的权重αt。每一次学习完成后，需要重新调整样本的权重，以使得在上一次分类中被错分的样本可以在接下来的学习中被重点学习。（**这就要求弱分类器可以进行带权重的样本训练**）当学习完T轮后，将所有弱分类器进行加权相加，得到最终的强分类器。

其中，错误率的定义为：。



“一个弱分类器”：

一次运用某一维特征并且选取某阈值及判决方向**使判错样本权值之和最小**的过程。这句话包含的意思是：一个弱分类器的判决用的只是所有训练样本n维特征中的一维而已，在那一维特征下选取适当的阈值及方向，可以得到当前样本权重分配情况下的最小判错样本权重之和（注意，与错误率是两个不同概念！二者只有在样本权重平均分配时才是相同概念）。

某一维特征在前面判决中已经用过了并不影响后面的弱分类器还是用该维特征来判，只要该维特征在那一轮的判决中可以使判错样本权重之和最小。也就是说训练样本只有n维特征，但是也可以训练出无数个弱分类器（无数次地从这n维特征中找最优）。

Adaboost的优点:

　　1)精度很高

　　2)可以使用各种方法构建子分类器，adaboost算法提供的是框架

　　3)弱分类器构造极其简单

　　4)简单，不用做特征筛选

　　5)不用担心overfitting！

Adaboost算法中的弱分类器一般选择**决策树/决策桩**， 也可以选择**朴素贝叶斯**作为弱分类器。

* **选择单层决策树（决策桩）作为弱分类器**

单层决策树仅基于单个特征来做决策。

步骤如下：

将**最小的加权错误率minError**设为正无穷；

对数据集中的**每一个特征**：

建立一棵单层决策树，并利用当前的加权样本集对其进行训练，计算其***加权错误率***；

若该单层决策树的加权错误率低于minError，则将其设为最佳单层决策树;

返回最佳单层决策树（作为该轮训练出的弱分类器）；

### Adaboost && Naïve Bayes

假设邮件的内容中包含的词汇为Wi，垃圾邮件Spam，正常邮件ham。 判断一份邮件，内容包含的词汇为Wi，判断该邮件是否是垃圾邮件，即计算P（S|Wi）这个条件概率。根据Bayes' theorem：

[Bayes' theorem](https://camo.githubusercontent.com/680fa43d97cc667266ec5ec7b3309f96abe4d559/68747470733a2f2f75706c6f61642e77696b696d656469612e6f72672f6d6174682f612f362f652f61366537663863353231646366303138623634383061383936373737336163332e706e67)

其中：

* Pr(S|Wi) 出现词汇Wi的邮件是垃圾邮件的条件概率（即后验概率）；
* Pr(S) 训练阶段邮件数据集中垃圾邮件的概率，或实际调查的垃圾邮件的概率（即先验概率）；
* Pr(Wi|S) 垃圾邮件中词汇Wi出现的概率；
* Pr(H) 训练阶段邮件数据集中正常邮件的概率，或实际调查的正常邮件的概率；
* Pr(Wi|H) 正常邮件中词汇Wi出现的概率；

**注：**程序中，通过计算出Pr(S|W)和Pr(H|W)，比较Pr(S|W)和Pr(H|W)的大小，判断是垃圾邮件还是正常邮件。我们发现Pr(S|W)和Pr(H|W)计算的分母相同，所以我们只需要比较分子即可。

**但存在两个问题：**

1. 当词汇不存在时，即ni=0，此时Pr(S|Wi) = 0，会造成P=0，无法比较
2. 当Pr(S|Wi)较小时，连乘操作会造成下溢出问题

**解决方案：**

* 计算P(Wi|S)和P(Wi|H)时，将所有词汇初始化出现的次数为1，并将分母初始化为2（或根据样本/实际调查结果调整分母的值）。

# 统计语料库中词汇在S和H中出现的次数

wordsInSpamNum = np.ones(numWords)

wordsInHealthNum = np.ones(numWords)

spamWordsNum = 2.0

healthWordsNum = 2.0

* 计算P(Wi|S)和P(Wi|H)时，对概率取对数

pWordsSpamicity = np.log(wordsInSpamNum / spamWordsNum)

pWordsHealthy = np.log(wordsInHealthNum / healthWordsNum)

所以最终比较的是，P(W1|S)P(W2|S)....P(Wn|S)P(S)和P(W1|H)P(W2|H)....P(Wn|H)P(H)的大小。

ps = sum(testWordsMarkedArray \* pWordsSpamicity) + np.log(pSpam)

ph = sum(testWordsMarkedArray \* pWordsHealthy) + np.log(1 - pSpam)

**我们在计算ps和ph联合后验概率时，可引入一个调整因子DS，其作用是调整词汇表中某一词汇的“垃圾程度”(spamicity)，**

ps = sum(testWordsMarkedArray \* pWordsSpamicity \* DS) + np.log(pSpam)

ph = sum(testWordsMarkedArray \* pWordsHealthy) + np.log(1 - pSpam)

其中DS通过Adaboost算法迭代获取最佳值。原理如下：

设定adaboost循环的次数count

交叉验证随机选择1000个样本

DS初始化为和词汇列表大小相等的全一向量

迭代循环count次：

设定最小分类错误率为inf

对于每一个样本：

在当前DS下对样本分类

如果分类出错：

计算出错的程度，即比较ps和ph的相差alpha

如果样本原本是spam，错分成ham：

DS[样本包含的词汇] = np.abs(DS[样本包含的词汇] + np.exp(alpha) / DS[样本包含的词汇])

如果样本原本是ham，错分成spam：

DS[样本包含的词汇] = DS[样本包含的词汇] - np.exp(alpha) / DS[样本包含的词汇]

计算错误率

保存最小的错误率和此时的词汇列表、P(Wi|S)和P(Wi|H)、DS等信息，即保存训练好的最佳模型的信息

### 决策树

决策树是一个类似于流程图的树结构，其中每个节点代表一个属性上的测试，每个分支代表一个测试输出，最后的叶结点代表类别。决策树方便改写为形如if- then的分类规则，易于理解和调试。

但是，决策树有以下缺点：

1. 对连续性的字段比较难预测
2. 不适应大规模的数据集，在此种情况下，决策树的构造会变得效率非常低下
3. 易出现过拟合问题

因此，本项目中不考虑决策树算法。

### Semi-supervised NB with EM

详见文献47.

### 隐含主题模型 / multi-domain方法

会花费大量额外的时间将文本分为多个种类（将二分类问题变成了多分类问题），成本高，不考虑了。

### Semi-supervised clustering方法

该方法要求训练集的文本较长，使得每个文本可被分为多个组件（component），故不太适用于每则消息都很短的聊天消息分类。

### 多层grams特征抽取方法 （文献46）

* Step1：将文本内容按照是否为中文字符分隔成若干个字符串，每个字符串为全部中文字符或全部非中文字符，并将其分成全部中文字符串集合和全部非中文字符串集合。
* Step2：对**中文字符串集合**中每个字符串使用n-grams特征抽取方法，但滑动窗口每次**往后移动2个字节**。例如，字符串“元人民币”一共为8个字符，通过n-grams方法抽取的特征为：“元人，人民，民币”。
* Step3：对**非中文字符串集合**中的每个字符串也使用n-grams方法抽取特征，但滑动窗口每次**往后移动1个字节**。
* 将两个集合中抽取的特征组成特征空间向量。

实验中，n-grams中的n均为4。