# 信 息 内 容 安 全 实 验 报 告

实验项目名称： 基于朴素贝叶斯的垃圾邮件过滤

班级： SC011701

学号： 2017302208；2017302209；2017302211

姓名： 高亦菲；胡梦莹；袁坤焱

指导教师： 杨黎斌

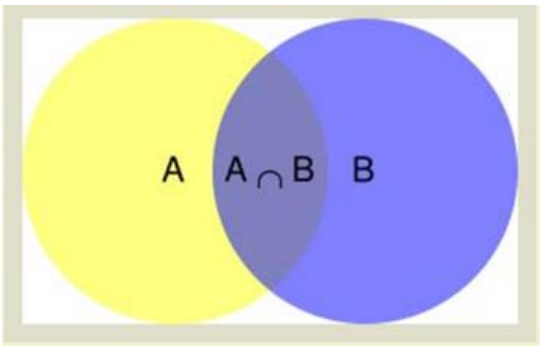
实验时间： 2020.3.29

# 目录

# 1.朴素贝叶斯原理

## 1.1 条件概率公式

所谓"条件概率"（Conditional probability），就是指在事件 B 发生的情况下，事件 A 发生的概率，用 P(A|B)来表示。




根据文氏图，如图 1 所示，可以很清楚地看到在事件 B 发生的情况下，事件 A 发生的概率就是 P(A∩B)除以 P(B)。

即$P(A|B)=\frac{P(AB)}{P(A)}$

同理可得$P(B|A)=\frac{P(AB)}{P(B)}$

可得条件概率计算公式：$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$

## 1.2 贝叶斯后验概率公式

对条件概率公式进行变形，可以得到如下形式：

$$P(A|B)=P(A)\cdot\frac{P(B|A)}{P(B)}$$

我们把 P(A)称为"先验概率"，即在 B 事件发生之前，我们对 A 事件概率的一个判断。P(A|B)称为"后验概率"，即在 B 事件发生之后，我们对 A 事件概率的重新评估。P(B|A)/P(B)称为"可能性函数"，这是一个调整因子，使得预估概率更接近真实概率。

所以，条件概率可以理解成下面的式子：

后验概率=先验概率×调整因子

这就是贝叶斯推断的含义。我们先预估一个"先验概率"，然后加入实验结果，看这个实验到底是增强还是削弱了"先验概率"，由此得到更接近事实的"后验概率"。

在这里，如果"可能性函数"P(B|A)/P(B)>1，意味着"先验概率"被增强，事

件 A 的发生的可能性变大；如果"可能性函数"=1，意味着 B 事件无助于判断事件 A 的可能性；如果"可能性函数"<1，意味着"先验概率"被削弱，事件 A 的可能性变小。

## 1.3 朴素贝叶斯

朴素贝叶斯算法是应用最为广泛的分类算法之一。朴素贝叶斯的思想基础是这样的：对于给出的待分类项，求解在此项出现的条件下各个类别出现的概率，哪个最大，就认为此待分类项属于哪个类别。

整个朴素贝叶斯分类分为三个阶段：

第一阶段——准备工作阶段，这个阶段的任务是为朴素贝叶斯分类做必要的准备，主要工作是根据具体情况确定特征属性，并对每个特征属性进行适当划分，然后由人工对一部分待分类项进行分类，形成训练样本集合。这一阶段的输入是所有待分类数据，输出是特征属性和训练样本。这一阶段是整个朴素贝叶斯分类中唯一需要人工完成的阶段，其质量对整个过程将有重要影响，分类器的质量很大程度上由特征属性、特征属性划分及训练样本质量决定。

第二阶段——分类器训练阶段，这个阶段的任务就是生成分类器，主要工作是计算每个类别在训练样本中的出现频率及每个特征属性划分对每个类别的条件概率估计，并将结果记录。其输入是特征属性和训练样本，输出是分类器。这一阶段是机械性阶段，根据前面讨论的公式可以由程序自动计算完成。

第三阶段——应用阶段。这个阶段的任务是使用分类器对待分类项进行分类，其输入是分类器和待分类项，输出是待分类项与类别的映射关系。这一阶段也是机械性阶段，由程序完成。

# 2. 贝叶斯过滤器模型

## 2.1 多项式模型

### 2.1.1Laplace 平滑处理

多项式模型在计算先验概率和条件概率时，会做一些平滑处理，具体公式为：

(1)先验概率：P（$C_i$）=（类别 $C_i$ 下各文档单词总数+$\alpha$）/（训练样本中各文档单词总数+k$\alpha$）,其中：k 为分类数，$\alpha$是平滑值。

(2)条件概率：P（$d_{ji}|C_i$）=（类别 $C_i$ 中单词 $D_{ji}$ 在各文档出现次数之和+$\alpha$）/（类别 $C_i$ 中单词总数+n$\alpha$）,其中：n 为特征的维数，$\alpha$是平滑值。

(3)在垃圾邮件处理的案例中，分类数 k=2,，即 spam 与 ham，特征维数 n 为训练文档中出现的单词数（无重复），也就是语料库 V。

当$\alpha$=1 时，称作 Laplace 平滑。如果不做平滑，当某一维特征的值没在训练样本中出现过时，会导致其条件概率为 0，从而导致后验概率为 0，加上平滑就可以克服这个问题。

### 2.1.2 多项式模型基本原理

多项式模型以单词为粒度，已知类别 $\{C_1, C_2, C_3, \cdots, C_k\}$ 与文档集合 $D\{D_1, D_2, \cdots, D_n\}$，设某一文档 $D_j$ 的词向量为 $D_j=\{d_{j1}, d_{j2}, d_{ji}\}$（可重复），设训练文档中出现的单词（单词出现多次只算一次）即语料库 V，对于待分类文档 $A=\{A_1, A_2, \cdots, A_M\}$，有：

(1)计算文档类别的先验概率

P（$C_i$）=（类别 $C_i$ 下各文档单词总数+1）/（训练样本中各文档单词总数+2）

P（$C_i$）可以认为是类别 $C_i$ 在整体上占多大比例（有多大可能性）

(2)某单词 $d_{ji}$ 在类别 $C_i$ 下的条件概率

P（$d_{ji}|C_i$）=（类别 $C_i$ 中单词 $D_{ji}$ 在各文档出现次数之和+1）/（类别 $C_i$ 中单词总数+V）

P（$d_{ji}|C_i$）可以看作是单词 $D_{ji}$ 在证明 $D_j$ 属于类别 $C_i$ 上提供了多大证据

(3)后验概率：对于待分类文档 A 被判定为类别 $C_i$ 的概率

假设文档中的词即 $A_1, A_2, \ldots A_m$ 相互独立，则有

$$P(C_i|A) = \frac{P(C_i)P(A|C_i)}{P(A)} = \frac{P(C_i)P(A_1|C_i)P(A_2|C_i)\cdots P(A_m|C_i)}{P(A)}$$

对于同一文档 P（A）一定，因此中需要计算分子的值。

多项式模型基于以上三步，最终以第三步中计算出的后验概率最大者为文档

A 所属类别。

## 2.2 伯努利模型

### 2.2.1 建立历史资料库

首先，我们必须预先提供两组已经识别好的邮件，一组是正常邮件，另一组是垃圾邮件。我们用这两组邮件，对过滤器进行"训练"。这两组邮件的规模越大，训练效果就越好。在这里，假设正常邮件与垃圾邮件各 4000 封。

"训练"过程很简单。首先，解析所有邮件，提取每一个词。然后，计算每个词语在正常邮件和垃圾邮件中的出现频率。比如，我们假定"sex"这个词，在 4000 封垃圾邮件中，有 200 封包含这个词，那么它的出现频率就是 5%；而在 4000 封正常邮件中，只有 2 封包含这个词，那么出现频率就是 0.05%。如果某个词只出现在垃圾邮件中，就假定它在正常邮件的出现频率是 1%，反之亦然。这样做是为了避免概率为 0。随着邮件数量的增加，计算结果会自动调整。

有了这个初步的统计结果，过滤器就可以投入使用了。

### 2.2.2 过滤器的使用过程

现在，我们收到了一封新邮件。在未经统计分析之前，我们假定它是垃圾邮件的概率为 50%。

我们用 S 表示垃圾邮件（spam），H 表示正常邮件（healthy）。因此，P(S)和 P(H)的先验概率，都是 50%。然后，对这封邮件进行解析，发现其中包含了 sex 这个词，我们用 W 表示"sex"这个词，那么问题就变成了如何计算 P(S|W)的值，即在某个词语（W）已经存在的条件下，垃圾邮件（S）的概率有多大。

根据条件概率公式，马上可以写出：

$$P(S|W) = \frac{P(W|S)P(S)}{P(W|S)P(S) + P(W|H)P(H)}$$

公式中，P(W|S)和 P(W|H)的含义是，这个词语在垃圾邮件和正常邮件中，分别出现的概率。这两个值可以从历史资料库中得到，对 sex 这个词来说，上文假定它们分别等于 5%和 0.05%。另外，P(S)和 P(H)的值，前面说过都等于 50%。所以，马上可以计算 P(S|W)=99.0%

因此，这封新邮件是垃圾邮件的概率等于 99%。这说明，sex 这个词的推断能力很强，将 50%的"先验概率"一下子提高到了 99%的"后验概率"。

如果有的词是第一次出现，无法计算 P(S|W)，Paul Graham 就假定这个值等于 0.4。因为垃圾邮件用的往往都是某些固定的词语，所以如果你从来没见过某个词，它多半是一个正常的词。

2.2.3 联合概率的计算

所谓联合概率，就是指在多个事件发生的情况下，另一个事件发生概率有多大。比如，已知 $W_1$ 和 $W_2$ 是两个不同的词语，它们都出现在某封电子邮件之中，那么这封邮件是垃圾邮件的概率，就是联合概率。在已知 $W_1$ 和 $W_2$ 的情况下，无非就是两种结果：垃圾邮件（事件 E1）或正常邮件（事件 E2），假设 $W_1$ 和 $W_2$ 的出现是两个独立事件，则

$$P(E_1) = P(S|W_1)P(S|W_2)P(S)$$

$$P(E_2) = \big(1 - P(S|W_1)\big)\big(1 - P(S|W_2)\big)\big(1 - P(S)\big)$$

因此在 $W_1$ 和 $W_2$ 已经发生的情况下，垃圾邮件的概率等于

$$P = \frac{P(E_1)}{P(E_1) + P(E_2)}$$

将 P(S)=0.5 代入得

$$P = \frac{P(S|W_1)P(S|W_2)}{P(S|W_1)P(S|W_2) + \big(1 - P(S|W_1)\big)\big(1 - P(S|W_2)\big)}$$

将$P(S|W_1)$记为 $P_1$，$P(S|W_2)$记为 $P_2$，公式变为

$$P = \frac{P_1P_2}{P_1P_2 + (1 - P_1)(1 - P_2)}$$

将两个词扩展为多个词，就得到了最终的计算公式：

$$P = \frac{P_1 P_2 \cdots P_n}{P_1 P_2 \cdots P_n + (1-P_1)(1-P_2)\cdots(1-P_n)}$$

最后，我们设定一个门槛值，即 P>0.9 是认定该邮件为垃圾邮件。

# 3. linearSVC 线性分类支持向量机

## 3.1 原理

SVM(Support Vector Machine)：支持向量机，是常见的一种判别方法。在机器学习领域，是一个有监督的学习模型，通常用来进行模式识别、分类以及回归分析。

支持向量机，其基本思路就是对于给定的数据样本，试图找到位于两类（或多类）样本“正中间”的一个划分超平面，把数据样本划分为不同的类。

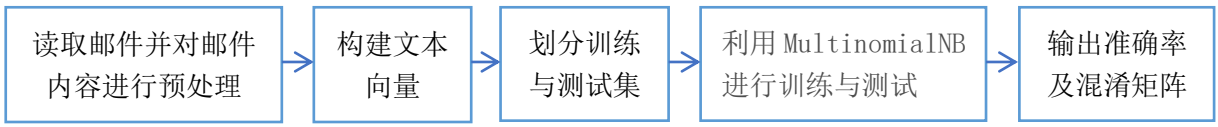
(1)分类器：机器学习的分类器，均可以看成一个或一组超平面，将 label 不同的数据点在数据空间中分开。对于线性可分问题，属于相同 label 的数据点在数据空间中可以看成是“类聚”的，即具有相同 label 的点会聚在一起。这样，分类效果最好的超平面应该满足：对于其分割的两种 label，距离最近的两个不同 label 的数据点距离超平面的距离都足够大，即超平面离两个类聚的空间都足够远。

(2)支持向量：对于支持向量机来说，最关心的并不是所有数据的分布情况，而是所谓类聚空间边界的相互位置，这些边界上的数据点，即两个空间间隔最小的两个数据点被称为支持向量，支持向量机分类器就是针对这些点优化的分类器。

# 4.实现

## 4.1 基于贝叶斯公式展开的多项式模型实现

4.1.1 实现流程




### 4.1.2 具体代码分析

(1)读取邮件内容并对邮件进行预处理

read_mails()对邮件内容过滤符号数字，将大写变为小写并进行分词

```
def read_mails():#读取邮件类型，名称，内容
    path="D:\PycharmProject\spam filtering\ham"
    path_list=os.listdir(path)
    for filename in path_list:
        label.append('ham')
        num.append(filename)
        f=open(os.path.join(path, filename),'r')
        text = f.read().lower()  # 小写
        token = RegexpTokenizer('[a-zA-Z]+').tokenize(text)  # 过滤符号数字，分词
        message.append(token)
        f.close()
```

make_panda_list 将邮件读取与 DataFrame 形式，即：{labe,num,message }：其中 label 为 spam/ham，num 标记邮件名称，如：1.TXT，message 为分词形式的邮件内容。

```
def make_panda_list():
    read_mails()
    data = {'label':label,
        'num':num,
        'message':message}
    df=pd.DataFrame(data)
    df['label'] = df['label'].replace(['spam', 'ham'], [1, 0])#spam 为垃圾邮件，标记为 1
    return df
```

(2)构建文本向量

```
#将分词后的邮件内容连接成句子
df['message'] = [' '.join(text) for text in df['message']]
# CountVectorizer 是一个词频统计的类，可以算出不同的单词在所给的文档中出现的频率
# 创建一个 CountVectorizer 对象
cv = CountVectorizer()
X = cv.fit_transform(df.message)
y = df.label
```

(3)划分训练集与测试集

一共 50 封邮件，其中 30 封划分为训练集，另外 20 封划分为测试集

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0)
```

(4)利用 MultinomialNB 进行训练与测试

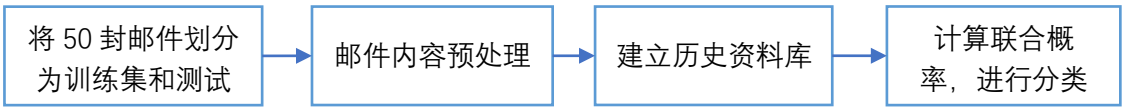
```
mnb = MultinomialNB()
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
mnb.fit(X_train, y_train)
y_pred = mnb.predict(X_test)
```

(5)输出准确率以及混淆矩阵

```
print("准确率:", accuracy_score(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
print('混淆矩阵:\n',cm)
```

## 4.2 基于联合概率公式的伯努利模型实现

### 4.2.1 实现流程




### 4.2.2 具体代码分析

(1)划分训练集和测试集

在正常邮件和垃圾邮件均为 25 封的情况下，随机抽取十封作为测试集，其它四十封作为训练集。

- 读取 email 文件夹下的目录路径

```
def fileWalker(path):
    fileArray = []
    for root, dirs, files in os.walk(path):
        for fn in files:
            eachpath = str(root+'\\'+fn)
            fileArray.append(eachpath)
    return fileArray
```

- 随机抽取十封放入测试集 test 文件夹

```
files = fileWalker(filepath)
    random.shuffle(files)
    top10 = files[:10]
```

```
    for ech in top10:
        ech_name = testpath+'\\'+('_'.join(ech.split('\\')[-2:]))
        shutil.move(ech, testpath)
        os.rename(testpath+'\\'+ech.split('\\')[-1], ech_name)
```

(2)邮件内容预处理

将邮件内容进行分词，按垃圾邮件和正常邮件的分类放入不同集合

• 对路径名下的邮件进行分词

```
def email_parser(email_path):
    punctuations = """,.<>()*&^%$#@!";~`[]{}|、\\/~+_-=?"""#删除邮件中的符号
    content_list = readtxt(email_path)
    content = (' '.join(content_list)).replace('\r\n', ' ').replace('\t', ' ')
    clean_word = []
    for punctuation in punctuations:
        content = (' '.join(content.split(punctuation))).replace('  ', ' ')
        clean_word = [word.lower()
                      for word in content.split(' ') if len(word) > 2]
    return clean_word#得到分词列表
```

•对正常邮件和垃圾邮件文件夹下的分词结果进行统计，得到词汇的集合

```
def get_word(email_file):
    word_list = []
    word_set = []
    email_paths = fileWalker(email_file)
    for email_path in email_paths:
        clean_word = email_parser(email_path)
        word_list.append(clean_word)
        word_set.extend(clean_word)
    return word_list, set(word_set)
```

(3)建立资料库

计算每个词语在正常邮件和垃圾邮件中的出现频率

```
    for word in union_set:
        counter = 0
        for email in email_list:#遍历每一封邮件
            if word in email:
                counter += 1
            else:
                continue
        prob = 0.0
        if counter != 0:
            prob = counter/len(email_list)
        else:
            prob = 0.01#该词只出现在垃圾邮件中,就假定它在正常邮件的出现频率是 1%
```

```
        word_prob[word] = prob#得到该词出现频率的字典
```

(4)计算联合概率，为测试邮件分类

- 联合概率的计算

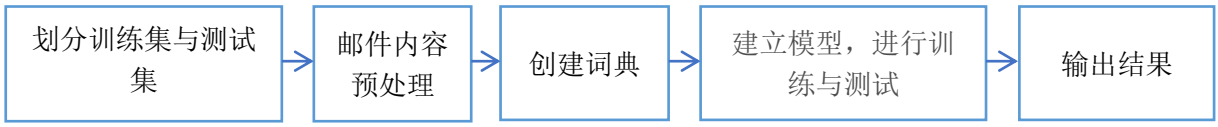
```
        for k, v in prob_dict.items():
            numerator *= v
            denominator_h *= (1-v)
        email_spam_prob = numerator/(numerator+denominator_h)#联合概率
```

- 输出各测试邮件的结果和概率

```
        if email_spam_prob > 0.9:
            print(file_name, 'spam', email_spam_prob)#该邮件为垃圾邮件
        else:
            print(file_name, 'ham', email_spam_prob)#该邮件为正常邮件
```

## 4.3 基于支持向量机的线性分类模型实现

### 4.3.1 实现流程




### 4.3.2 具体代码实现

(1)划分训练集与测试集

一共 50 封邮件，其中 30 封划分为训练集，另外 20 封划分为测试集，训练集与测试集中垃圾邮件与正常邮件数量一致。

(2)邮件内容预处理

- 读取邮件内容保存在字符串 text 中：

```
text = ''
for mail in emails:
    with open(mail) as m:
        f = m.read()
        text += f
```

- 处理非字母的字符，并单词替换为小写：

```
string = re.sub('[^a-zA-Z]', ' ', text)
string = re.sub(r'\s+', ' ', string).lower()
```

- 处理停用词，将句子分词，并将单词进行词性还原：

```
stop_words = set(stopwords.words('english'))
tokenizer = RegexpTokenizer('[a-z]+')
lemmatizer = WordNetLemmatizer()
token = tokenizer.tokenize(string)  # 分词
token =  [lemmatizer.lemmatize(w) for w in token if  lemmatizer.lemmatize(w)
             not  in
       stop_words]  # 停用词+词形还原
```

(3)创建词典

```
from collections import Counter
dic = Counter(token)
dic = dic.most_common(400)
```

(4)建立模型，进行训练与测试

- 建立模型

```
model = LinearSVC()
```

- 训练

训练、测试时同样要对邮件进行预处理，与第（2）步类似。

```
train_labels = np.zeros(30)
train_labels[15:30] = 1model.fit(train_matrix, train_labels)
train_matrix = extract_features(train_dir)
```

- 测试

```
test_matrix = extract_features(test_dir)
test_labels = np.zeros(20)
test_labels[10:20] = 1
result = model.predict(test_matrix)
```

(5)输出结果

```
print(confusion_matrix(test_labels, result))
```

# 5.结果与分析

## 5.1 结果展示

通过以上三种方法，训练集中的邮件中有一封邮件都识别错误，即垃圾邮件spam17.txt:


提取该邮件的关键词为：

['home', 'based', 'business', 'opportunity', 'knocking', 'door', 'rude', 'let', 'chance', 'go', 'earn', 'great', 'income', 'find', 'financial', 'life', 'transformed', 'learn', 'success', 'work', 'home', 'finder', 'expert']

分析可知，该封邮件中的关键词在词典中出现频率低，不如其它垃圾邮件的关键词的指向性明显。

## 5.2 朴素贝叶斯与支持向量机性能分析

朴素贝叶斯分类器结构简单，实现多分类的性能较好，尤其是在样本数量较大的时候优势更加明显；支持向量机已被证明在小样本情况下表现突出，并成功应用于多个分类领域。而在相同数据量的情况下，SVM 用时较长。

在本次实验中，因为样本数量不大，且大部分垃圾邮件与正常邮件内容对比明显，SVM 与朴素贝叶斯的效果对比并不是很明显。

## 备注

实验任务分工：

高亦菲：朴素贝叶斯——多项式模型代码实现，及相关报告内容的书写

胡梦莹：支持向量机的分类模型代码实现，及相关报告内容的书写

袁坤焱：朴素贝叶斯——伯努利模型代码实现，及相关报告内容的书写