

实验介绍

1.实验内容

本实验介绍如何使用贝叶斯算法解决一个实际问题：新闻分类。

2.实验目标

通过本实验进一步掌握贝叶斯算法的原理，掌握如何使用贝叶斯算法解决实际问题，掌握实际世界中贝叶斯算法的解决流程。

3. 实验知识点

- 贝叶斯算法
- 数据预处理
- 梯度下降
- sklearn

4. 实验环境

- python 3.6.5
- jieba

5.预备知识

- 概率论与数理统计
- Linux命令基本操作
- Python编程基础

准备工作

点击屏幕右上方的下载实验数据模块，选择下载bayes_news.tgz到指定目录下，然后再依次选择点击上方的File->Open->Upload,上传刚才下载的数据集压缩包，再使用如下命令解压：

```
In [1]: !tar -zxvf bayes_news.tgz
```

```
bayes_news/  
bayes_news/Sample/  
bayes_news/Sample/C000008/  
bayes_news/Sample/C000008/10.txt  
bayes_news/Sample/C000008/11.txt  
bayes_news/Sample/C000008/12.txt  
bayes_news/Sample/C000008/13.txt  
bayes_news/Sample/C000008/14.txt  
bayes_news/Sample/C000008/15.txt  
bayes_news/Sample/C000008/16.txt  
bayes_news/Sample/C000008/17.txt  
bayes_news/Sample/C000008/18.txt  
bayes_news/Sample/C000008/19.txt  
bayes_news/Sample/C000010/  
bayes_news/Sample/C000010/10.txt  
bayes_news/Sample/C000010/11.txt  
bayes_news/Sample/C000010/12.txt  
bayes_news/Sample/C000010/13.txt  
bayes_news/Sample/C000010/14.txt  
bayes_news/Sample/C000010/15.txt  
bayes_news/Sample/C000010/16.txt  
bayes_news/Sample/C000010/17.txt  
bayes_news/Sample/C000010/18.txt  
bayes_news/Sample/C000010/19.txt  
bayes_news/Sample/C000013/  
bayes_news/Sample/C000013/10.txt  
bayes_news/Sample/C000013/11.txt  
bayes_news/Sample/C000013/12.txt  
bayes_news/Sample/C000013/13.txt  
bayes_news/Sample/C000013/14.txt  
bayes_news/Sample/C000013/15.txt  
bayes_news/Sample/C000013/16.txt  
bayes_news/Sample/C000013/17.txt  
bayes_news/Sample/C000013/18.txt  
bayes_news/Sample/C000013/19.txt  
bayes_news/Sample/C000014/  
bayes_news/Sample/C000014/10.txt  
bayes_news/Sample/C000014/11.txt  
bayes_news/Sample/C000014/12.txt  
bayes_news/Sample/C000014/13.txt  
bayes_news/Sample/C000014/14.txt  
bayes_news/Sample/C000014/15.txt  
bayes_news/Sample/C000014/16.txt  
bayes_news/Sample/C000014/17.txt  
bayes_news/Sample/C000014/18.txt  
bayes_news/Sample/C000014/19.txt  
bayes_news/Sample/C000016/  
bayes_news/Sample/C000016/10.txt  
bayes_news/Sample/C000016/11.txt  
bayes_news/Sample/C000016/12.txt  
bayes_news/Sample/C000016/13.txt  
bayes_news/Sample/C000016/14.txt  
bayes_news/Sample/C000016/15.txt  
bayes_news/Sample/C000016/16.txt  
bayes_news/Sample/C000016/17.txt  
bayes_news/Sample/C000016/18.txt  
bayes_news/Sample/C000016/19.txt  
bayes_news/Sample/C000020/  
bayes_news/Sample/C000020/10.txt  
bayes_news/Sample/C000020/11.txt  
bayes_news/Sample/C000020/12.txt  
bayes_news/Sample/C000020/13.txt  
bayes_news/Sample/C000020/14.txt  
bayes_news/Sample/C000020/15.txt  
bayes_news/Sample/C000020/16.txt  
bayes_news/Sample/C000020/17.txt  
bayes_news/Sample/C000020/18.txt  
bayes_news/Sample/C000020/19.txt  
bayes_news/Sample/C000022/  
bayes_news/Sample/C000022/10.txt
```

bayes_news/Sample/C000022/11.txt
bayes_news/Sample/C000022/12.txt
bayes_news/Sample/C000022/13.txt
bayes_news/Sample/C000022/14.txt
bayes_news/Sample/C000022/15.txt
bayes_news/Sample/C000022/16.txt
bayes_news/Sample/C000022/17.txt
bayes_news/Sample/C000022/18.txt
bayes_news/Sample/C000022/19.txt
bayes_news/Sample/C000023/
bayes_news/Sample/C000023/10.txt
bayes_news/Sample/C000023/11.txt
bayes_news/Sample/C000023/12.txt
bayes_news/Sample/C000023/13.txt
bayes_news/Sample/C000023/14.txt
bayes_news/Sample/C000023/15.txt
bayes_news/Sample/C000023/16.txt
bayes_news/Sample/C000023/17.txt
bayes_news/Sample/C000023/18.txt
bayes_news/Sample/C000023/19.txt
bayes_news/Sample/C000024/
bayes_news/Sample/C000024/10.txt
bayes_news/Sample/C000024/11.txt
bayes_news/Sample/C000024/12.txt
bayes_news/Sample/C000024/13.txt
bayes_news/Sample/C000024/14.txt
bayes_news/Sample/C000024/15.txt
bayes_news/Sample/C000024/16.txt
bayes_news/Sample/C000024/17.txt
bayes_news/Sample/C000024/18.txt
bayes_news/Sample/C000024/19.txt
bayes_news/ClassList.txt
bayes_news/stopwords_cn.txt

框架

本实验使用Python3编程实现一个简单的新闻分类算法。

1.朴素贝叶斯理论


朴素贝叶斯是贝叶斯决策理论的一部分，所以在讲述朴素贝叶斯之前有必要快速了解一下贝叶斯决策理论。

2.新闻分类

接下来我们开始用朴素贝叶斯算法进行新闻分类实战。

中文语句切分

考虑一个问题，英文的语句可以通过非字母和非数字进行切分，但是汉语句子呢？就比如我打的这一堆字，该如何进行切分呢？我们自己写个规则？幸运地是，这部分的工作不需要我们自己做了，可以直接使用第三方分词组件，即jieba，没错就是“结巴”。新闻分类数据集可在实验目录下找到，数据集已经做好分类，分文件夹保存，分类结果如下：



C000008 财经
C000010 IT
C000013 健康
C000014 体育
C000016 旅游
C000020 教育
C000022 招聘
C000023 文化
C000024 军事

数据集已经准备好，接下来，让我们直接进入正题。切分中文语句，编写如下代码：

```

In [3]: # -*- coding: UTF-8 -*-
import os
import jieba
def TextProcessing(folder_path):
    folder_list = os.listdir(folder_path) #查看folder_path下的文件
    data_list = [] #训练集
    class_list = []

    #遍历每个子文件夹
    for folder in folder_list:
        new_folder_path = os.path.join(folder_path, folder) #根据子文件夹,生成新的路径
        files = os.listdir(new_folder_path) #存放子文件夹下的txt文件的列表

        j = 1
        #遍历每个txt文件
        for file in files:
            if j > 100: #每类txt样本数最多100个
                break
            with open(os.path.join(new_folder_path, file), 'r', encoding='utf-8') as f: #打开txt文件
                raw = f.read()

            word_cut = jieba.cut(raw, cut_all=False) #精简模式,返回一个可迭代的generator
            word_list = list(word_cut) #generator转换为list

            data_list.append(word_list)
            class_list.append(folder)
            j += 1
        print(data_list)
        print(class_list)

if __name__ == '__main__':
    #文本预处理
    folder_path = 'bayes_news/Sample' #训练集存放地址
    TextProcessing(folder_path)

```

惯', '形成', '了', '鲜明', '的', '对比', '。', '\n', '\u3000', '\u3000', '然而', '这种', '支付', '模式', '和', '传统', '的', '网络', '支付', '并', '无', '本质', '的', '区别', '。', '因为', '每', '一个', '手机号码', '和', '邮件地址', '背后', '都', '会', '对应', '着', '一个', '账户', '—', '—', '这个', '账户', '可以', '是', '信用卡', '账户', '、', '借记卡', '账户', '、', '也', '包括', '邮局', '汇款', '、', '手机', '代收', '、', '电话', '代收', '、', '预付费', '卡', '和', '点卡', '等', '多种形式', '。', '\n', '\u3000', '\u3000', '“', '快', '钱', '的', '功能', '其实', '就', '相当于', '融合', '了', '很多', '交易', '工具', '的', 'VISA卡', '、', '所以', '又', '被', '称为', '网络', 'VISA', '。', '”', '关国光', '说', '、', '“', '从', '本质', '上', '讲', '、', '我们', '和', 'VISA', '等', '采用', '的', '底层', '技术', '是', '没有', '差别', '的', '、', '我们', '和', '它', '的', '区别', '在于', 'VISA卡', '面对', '的', '交易', '工具', '比较', '单一', '、', '而', '快', '钱', '面对', '的', '是', '多种', '分散', '的', '交易', '工具', '。', '”', '\n', '\u3000', '\u3000', '因为', '“', '信用', '缺位', '”', '、', '网络', '支付', '一直', '是', '困扰', '中国', '电子商务', '发展', '的', '瓶颈', '之一', '。', '网络', '支付', '平台', '相当于', '“', '信用', '缺位', '”', '条件', '下', '的', '“', '补位', '产物', '”', '、', '它', '把', '众多', '的', '银行卡', '整合', '到', '一个', '页面', '端口', '、', '以', '支付', '公司', '作为', '信用', '中介', '、', '在', '买家', '确认', '收到', '商品', '前', '、', '代替', '买卖双方', '暂时', '保管', '货款', '。', '\n', '\u3000', '\u3000', '目前', '最', '知名', '的', '网络', '支付', '平台', '包括', '阿里巴巴', '的', '支付宝', '和', 'eBay', '的', 'Paypal', '（', '贝宝', '）', '。', '关国光', '表示', '、', '快', '钱', '最大', '的', '特点', '是', '第三方', '的', '支付', '平台', '、', '主要', '客户', '为', '那些', '中小', '公司', '。', '这些', '网络', '支付', '平台', '的', '主要', '业务', '是', '针对', '母公司', '的', '、', '不可能', '被', '其', '母公司', '同行', '借用', '。', '\u3000', '\u3000', '“

可以看到，我们已经顺利将每个文本进行切分，并进行了类别标记。

词频统计

我们将所有文本分成训练集和测试集，并对训练集中的所有单词进行词频统计，并按降序排序。也就是将出现次数多的词语在前，出现次数少的词语在后进行排序。编写代码如下：

In [4]: # -*- coding: UTF-8 -*-

```
import os
import random

import jieba

"""
函数说明: 中文文本处理

Parameters:
    folder_path - 文本存放的路径
    test_size - 测试集占比, 默认占所有数据集的百分之20
Returns:
    all_words_list - 按词频降序排序的训练集列表
    train_data_list - 训练集列表
    test_data_list - 测试集列表
    train_class_list - 训练集标签列表
    test_class_list - 测试集标签列表
"""

def TextProcessing(folder_path, test_size=0.2):
    folder_list = os.listdir(folder_path) #查看folder_path下的文件
    data_list = [] #数据集数据
    class_list = [] #数据集类别

    #遍历每个子文件夹
    for folder in folder_list:
        new_folder_path = os.path.join(folder_path, folder) #根据子文件夹, 生成新的路径
        files = os.listdir(new_folder_path) #存放子文件夹下的txt文件的列表

        j = 1
        #遍历每个txt文件
        for file in files:
            if j > 100: #每类txt样本数最多100个
                break
            with open(os.path.join(new_folder_path, file), 'r', encoding='utf-8') as f: #打开txt文件
                raw = f.read()

            word_cut = jieba.cut(raw, cut_all=False) #精简模式, 返回一个可迭代的generator
            word_list = list(word_cut) #generator转换为list

            data_list.append(word_list) #添加数据集数据
            class_list.append(folder) #添加数据集类别
            j += 1

    data_class_list = list(zip(data_list, class_list)) #zip压缩合并, 将数据与标签对应压缩
    random.shuffle(data_class_list) #将data_class_list乱序
    index = int(len(data_class_list) * test_size) + 1 #训练集和测试集切分的索引值
    train_list = data_class_list[index:] #训练集
    test_list = data_class_list[:index] #测试集
    train_data_list, train_class_list = zip(*train_list) #训练集解压缩
    test_data_list, test_class_list = zip(*test_list) #测试集解压缩

    all_words_dict = {} #统计训练集词频
    for word_list in train_data_list:
        for word in word_list:
            if word in all_words_dict.keys():
                all_words_dict[word] += 1
            else:
                all_words_dict[word] = 1

    #根据键的值倒序排序
    all_words_tuple_list = sorted(all_words_dict.items(), key=lambda f: f[1], reverse=True)
    all_words_list, all_words_nums = zip(*all_words_tuple_list) #解压缩
    all_words_list = list(all_words_list) #转换成列表
    return all_words_list, train_data_list, test_data_list, train_class_list, test_class_list

if __name__ == '__main__':
    #文本预处理
    folder_path = 'bayes_news/Sample' #训练集存放地址
```

```
all_words_list, train_data_list, test_data_list, train_class_list, test_class_list = TextProcessing(folder, test_s
print(all_words_list)
```

`all_words_list`就是将所有训练集的切分结果通过词频降序排列构成的单词合集。观察一下打印结果，不难发现，这里包含了很多标点符号，很显然，这些标点符号是不能作为新闻分类的特征的。为了降低这些高频的符号对分类结果的影响，需要将这些标点符号删除。除了这些，还有“在”，“了”这样对新闻分类没有帮助的词。此外，还有一些数字，数字显然也不能作为分类新闻的特征。所以要消除它们对分类结果的影响，可以定制一个规则。

一个简单的规则可以这样制定：

- 首先去掉高频词，至于去掉多少个高频词，我们可以通过观察去掉高频词个数和最终检测准确率的关系来确定。
- 然后，去除数字，不把数字作为分类特征。
- 最后，去除一些特定的词语，比如：“的”，“一”，“在”，“不”，“当然”，“怎么”这类的对新闻分类无影响的介词、代词、连词。

数据清洗

可以使用已经整理好的stopwords_cn.txt文本来去除这些词，stopwords_cn.txt位于当前实验的数据集目录下。

可以使用head命令查看该文件内容的前40行:

```
In [5]: !head -n 40 bayes_news / stopwords_cn.txt
```

'head' 不是内部或外部命令，也不是可运行的程序
或批处理文件。

所以可以根据这个文档，将这些单词去除，不作为分类的特征。先去除前100个高频词汇，然后编写代码如下：

In [6]:

```
"""
函数说明:读取文件里的内容,并去重

Parameters:
    words_file - 文件路径
Returns:
    words_set - 读取的内容的set集合
"""

def MakeWordsSet(words_file):
    words_set = set() #创建set集合
    with open(words_file, 'r', encoding='utf-8') as f: #打开文件
        for line in f.readlines(): #一行一行读取
            word = line.strip() #去回车
            if len(word) > 0: #有文本,则添加到words_set中
                words_set.add(word)
    return words_set #返回处理结果

"""
函数说明:文本特征选取

Parameters:
    all_words_list - 训练集所有文本列表
    deleteN - 删除词频最高的deleteN个词
    stopwords_set - 指定的结束语
Returns:
    feature_words - 特征集
"""

def words_dict(all_words_list, deleteN, stopwords_set=None):
    if stopwords_set is None:
        stopwords_set = set()
    feature_words = [] #特征列表
    n = 1
    for t in range(deleteN, len(all_words_list), 1):
        if n > 1000: #feature_words的维度为1000
            break
        #如果这个词不是数字,并且不是指定的结束语,并且单词长度大于1小于5,那么这个词就可以作为特征词
        if not all_words_list[t].isdigit() and all_words_list[t] not in stopwords_set and 1 < len(all_words_list[t]) < 5:
            feature_words.append(all_words_list[t])
        n += 1
    return feature_words

if __name__ == '__main__':
    #文本预处理
    folder_path = 'bayes_news/Sample' #训练集存放地址
    all_words_list, train_data_list, test_data_list, train_class_list, test_class_list = TextProcessing(folder_path, test_size=0.2)

    #生成stopwords_set
    stopwords_file = 'bayes_news/stopwords.cn.txt'
    stopwords_set = MakeWordsSet(stopwords_file)

    feature_words = words_dict(all_words_list, 100, stopwords_set)
    print(feature_words)
```

['支付', '黄金周', '五一', '成为', '仿制', '发展', '增长', '选择', '学校', '远程', '记者', '主要', '复习', '可能', '问题', '品牌', '比赛', '企业', '一定', '分析', '建设', '射程', '学习', '通过', '银行', '完全', '部署', '工作', '辅导班', '重要', '亿美元', '专业', '期间', '部分', '时候', '需要', '很多', '管理', '今年', '表示', '填报', '使用', '考试', '文章', '达到', '能力', '开始', '产品', 'VS', '情况', '万人次', '军事', '阵地', '技术', '表现', '希望', '几乎', '相对', '部队', '影响', '拥有', '服务', '用户', '基础', '科学', '现在', '提高', '国家', '日本', '收入', '活动', '新浪', '接待', '必须', '来源', '最后', '提供', '老师', '资料', '显示', '印度', '知道', '实验室', '网络', '发现', '比较', '考古', '一直', '计划', '经济', '最大', '进入', '重点', '电话', '不用', '上海', '了解', '考研', '专利', '公里', '角度', '人数', '机会', '参加', '这是', '要求', '应该', '距离', '方面', '作用', '员工', '相关', '全国', '准备', '压制', '阿里', '岛屿', '去年', '休闲', '方式', '手机', '非常', '力气', '国内', '彻底', '睡眠', '沿海', '大批', '摧毁', '左右', '告诉', '文化', '一家', '介绍', '平台', '训练', '系统', '孩子', '不同', '出现', '我国', '台湾', '项目', '置于', '挑衅', 'MBA', '全面', '指挥', '全军', '不会', '不能', '香港', '考虑', '医院', '纳斯', '自寻死路', '世界领先', '型号', '开战', '金贵', '海量', '之内', '费多', '廉价', '发展观', '消费者', '利用', '吸引', '增加', '目标', '战场', '正在', '包

括', '两个', '数学', '地方', '历史', '沈阳市', '教育', '药厂', '牛奶', '一次', '这种', '消费', '喜欢', '信息', '化', '专家', '一批', '决定', '推出', '图库', '一下', '容易', '是否', '价值', '关系', '赔偿', '连续', '建议', '录取', '全球', '内容', '预期', '治疗', '军队', '东引岛', '止痛药', '世界', '东南亚', '指出', '网上', '不断', '理由', '消息', '排名', '顾客', '电脑', '注意', '获得', '之间', '明显', '成功', '著名', '分期付款', '知识点', '越来越', '数字', '泰国', '基本', '旅行社', '未来', '努力', '帮助', '景区', '同比', '面对', '得到', '本场', '完成', '英语', '院校', '组织', '辽宁队', '最佳', '火力', '掌握', '过程', '社会', '药物', '医疗', '能够', '营养', '武器', '镇痛药', '我军', '公布', '标志', '代表', '旅游者', '预计', '俄罗斯', '学员', '实现', '一场', '研究', '中心', '小时', '事情', '领导', '之后', '相当', '第一', '交易', '对手', '每个', '大学', '特点', '销售', '经验', '补充', '方法', '条件', '业务', '结果', '上市', '协议', '患者', '詹姆斯', '写作', '词汇', '利苑', '概念', '分钟', '认证', '埃及', '超过', '往往', '韩国', '因素', '数量', '欧洲', '演练', '环境', '功能', '加强', '各型', '回家', '三个', '整个', '上午', '客场', '此前', '其实', '领域', '建立', '关键', '客户', '万元', '支持', '媒体', '之前', '免息', '商机', '不少', '感觉', '过去', '综合', '装备', '一起', '信息', '过年', '不要', '一样', '城市', '提前', '这家', '知名', '产生', '更加', '国际', '复试', '阅读', '十分', '开通', '我省', '职业', '稳定', '最近', '伯德', '思路', '备考', '关国光', '东莞', '疼痛', '旅游业', '预测', '数据', '第一次', '一位', '共同', '关注', '投入', '进攻', '满足', '资源', '知识', '同事', '同学', '刚刚', '主场', '特别', '今天', '本报记者', '设计', '一年', '带来', '进一步', '理解', '昨天', '实施', '南京', '应用', '原因', '本科', '办法', '方向', '市民', '电视', '振保', '采取', '元老', '密码', '批次', '大学生', '晋升', '大量', '评选', '口技', '食物', '失眠', '酒家', '解题', '公式', '遗址', '标题', 'H股', '股东', '熟悉', '出境', '公民', '景点', '变得', '协会', '展开', '促进', '安排', '战争', '蓝军', '状态', '起来', '集团', '正式', '人士', '意味着', '广东', '统计', '亿元', '报道', '联想', '竞争', '购买', '以下', '收益', '效果', '有效', '有限公司', '生活', '很大', '压力', '之一', '当时', '每天', '坚持', '很快', '迅速', 'CEO', '主动', '这次', '攻击', '具有', '补报', '单位', '家长', '三分', '一半', '制药', '曹国伟', 'gt', '透露', '吸收', '传统', '第三方', '听课', '文物', '关键字', '美国在线', '东部', '内容摘要', '下载', '广播', '举办', '负责人', '价格', '宣布', '看到', '发布', '战略', '不再', '法国', '发生', '几年', '心理', '本报', '模拟', '开展', '人才', '力量', '电子', '红军', '突出', '核心', '培养', '参与', '十大', '根本', '充足', '优势', '俱乐部', '第二', '实力', '提升', '董事长', '发出', '不足', '商业银行', '只能', '广告', '发挥', '增幅', '积极', '经理', '诉讼', '找到', '初盘', '新型', '依然', '保证', '总结', '形成', '程度', '辽足', '马林', '唐尧东', '几个', '海上', '这一', '沈阳', '围棋', '首次', '运动', '网站', '可选报', '结束', '一般', '规则', '销售额', '每股', '最好', '米勒', '罚球', '人体', '尤其', '网上支付', '基地', '题型', '设立', '吸烟', '万人', '戒烟', '场位', '巨大', '阿片类', '埃弗顿', '矩阵', '牙膏', '口语', '国防', '敏华', '具体', '水平', '突破', '出境游', '航线', '专门', '成立', '东北亚', '每年', '有望', '月份', '费用', '主题', '说明', '充分', '保障', '双方', '联系', '先后', '变成', '机票', '避免', '展示', '举行', '挑战', '创造', '胜利', '更是', '困难', '类似', '人口', '动力', '标准', '存在', '外界', '同期', '属于', '行动', '采用', '最终', '直接', '足彩', '切沃', '佛罗伦萨', '报告', '考场', '练习', '听力', '一天', '时代', '乡村', '相比', '下降', '投诉', '比例', '联赛', '失去', '意义', '至少', '打击', '留下', '家教', '学生', '降价', '空间', '表明', '业绩', '安妮', '呼叫', '改革', '汪力', '涉及', '平均', '美元', '泰华', '合作', '女兵', '工具', '原则', '优秀', '教材', '身高', '生长', '安全性', '短程', '战术导弹', '点穴', '考前', '语法', '概率', '参看', '本书', '研究所', '去年同期', '高清晰', '迎来', '地区', '目的地', '总经理', '一体化', '旅游圈', '拉动', '修改', '速度', '学院', '各级', '精神', '官兵', '现实', '终于', '评估', '真实', '增强', '战术', '学科', '培训', '注重', '春节', '感到', '做到', '汽车', '感受', '健康', '不好', '真正', '集中', '兄弟', '广州', '各地', '免费', '自然', '即将', '内部', '政府', '经营', '一页', '大多数', '为主', '大部分', '活力', '利润', '鼓励', '高级', '贯彻', '本报讯', '继续', '主队', '阶段', '轻松', 'of', '独立', '调剂', '战胜', '二外', '相互', '招生', '需求', '机制', '纳入', '这部分', '行业', '商业', '持续']

可以看到，我们已经滤除了那些没有用的词组，这个feature_words就是我们最终选出的用于新闻分类的特征。随后，我们就可以根据feature_words，将文本向量化，然后用于训练朴素贝叶斯分类器。

Sklearn接口说明

数据已经处理好了，接下来就可以使用sklearn构建朴素贝叶斯分类器了。

官方英文文档地址：http://scikit-learn.org/dev/modules/generated/sklearn.naive_bayes.MultinomialNB.html (http://scikit-learn.org/dev/modules/generated/sklearn.naive_bayes.MultinomialNB.html)

朴素贝叶斯是一类比较简单的算法，scikit-learn中朴素贝叶斯类库的使用也比较简单。相对于决策树，KNN之类的算法，朴素贝叶斯需要关注的参数是比较少的，这样也比较容易掌握。在scikit-learn中，一共有3个朴素贝叶斯的分类算法类。分别是GaussianNB，MultinomialNB和BernoulliNB。其中GaussianNB就是先验为高斯分布的朴素贝叶斯，MultinomialNB就是先验为多项式分布的朴素贝叶斯，而BernoulliNB就是先验为伯努利分布的朴素贝叶斯。上篇文章讲解的先验概率模型就是先验概率为多项式分布的朴素贝叶斯。

sklearn.naive_bayes: Naive Bayes

The `sklearn.naive_bayes` module implements Naive Bayes algorithms. These are supervised learning methods based on applying Bayes' theorem with strong (naive) feature independence assumptions.

User guide: See the [Naive Bayes](#) section for further details.

<code>naive_bayes.BernoulliNB</code> ([alpha, binarize, ...])	Naive Bayes classifier for multivariate Bernoulli models.
<code>naive_bayes.GaussianNB</code> ([priors])	Gaussian Naive Bayes (GaussianNB)
<code>naive_bayes.MultinomialNB</code> ([alpha, ...])	Naive Bayes classifier for multinomial models

对于新闻分类，属于多分类问题。我们可以使用MultinomialNB()完成我们的新闻分类问题。另外两个函数的使用暂且不再进行扩展，可以自行学习。MultinomialNB假设特征的先验概率为多项式分布，即如下式：

$$P(X_j = x_{jl} | Y = C_k) = \frac{x_{jl} + \lambda}{m_k + n\lambda}$$

其中， $P(X_j = x_{jl} | Y = C_k)$ 是第k个类别的第j维特征的第l个取值条件概率。 m_k 是训练集中输出为第k类的样本个数。 λ 为一个大于0的常数，尝尝取值为1，即拉普拉斯平滑，也可以取其他值。

接下来，我们看下MultinomialNB这个函数，只有3个参数：

```
class sklearn.naive_bayes. MultinomialNB (alpha=1.0, fit_prior=True, class_prior=None) ¶
```

[\[source\]](#)

参数说明如下：

- alpha：浮点型可选参数，默认为1.0，其实就是添加拉普拉斯平滑，即为上述公式中的 λ ，如果这个参数设置为0，就是不添加平滑；
- fit_prior：布尔型可选参数，默认为True。布尔参数fit_prior表示是否要考虑先验概率，如果是false,则所有的样本类别输出都有相同的类别先验概率。否则可以自己用第三个参数class_prior输入先验概率，或者不输入第三个参数class_prior让MultinomialNB自己从训练集样本来计算先验概率，此时的先验概率为 $P(Y=C_k)=m_k/m$ 。其中m为训练集样本总数量， m_k 为输出为第k类别的训练集样本数。
- class_prior：可选参数，默认为None。

总结如下：

fit_prior	class_prior	最终先验概率
False	填或不填没有意义	$P(Y = C_k) = 1 / k$
True	不填	$P(Y = C_k) = m_k / m$
True	填	$P(Y = C_k) = \text{class_prior}$

除此之外，MultinomialNB也有一些方法供我们使用：

Methods

<code>fit</code> (X, y[, sample_weight])	Fit Naive Bayes classifier according to X, y
<code>get_params</code> ([deep])	Get parameters for this estimator.
<code>partial_fit</code> (X, y[, classes, sample_weight])	Incremental fit on a batch of samples.
<code>predict</code> (X)	Perform classification on an array of test vectors X.
<code>predict_log_proba</code> (X)	Return log-probability estimates for the test vector X.
<code>predict_proba</code> (X)	Return probability estimates for the test vector X.
<code>score</code> (X, y[, sample_weight])	Returns the mean accuracy on the given test data and labels.
<code>set_params</code> (**params)	Set the parameters of this estimator.

MultinomialNB一个重要的功能是有partial_fit方法，这个方法的一般用在如果训练集数据量非常大，一次不能全部载入内存的时候。这时我们可以把训练集分成若干等分，重复调用partial_fit来一步步的学习训练集，非常方便。GaussianNB和BernoulliNB也有类似的功能。在使用MultinomialNB的fit方法或者partial_fit方法拟合数据后，可以进行预测。

此时预测有三种方法，包括predict，predict_log_proba和predict_proba。predict方法就是我们最常用的预测方法，直接给出测试集的预测类别输出。predict_proba则不同，它会给出测试集样本在各个类别上预测的概率。容易理解，predict_proba预

测出的各个类别概率里的最大值对应的类别，也就是predict方法得到类别。predict_log_proba和predict_proba类似，它会给出测试集样本在各个类别上预测的概率的一个对数转化。转化后predict_log_proba预测出的各个类别对数概率里的最大值对应的类别，也就是predict方法得到类别。具体细节可参照官网手册。

【练习】基于Sklearn实现分类

了解了这些，我们就可以编写代码，通过观察取不同的去掉前deleteN个高频词的个数与最终检测准确率的关系，确定deleteN的取值：

In [32]:

```
# -*- coding: UTF-8 -*-
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB

def TextFeatures(train_data_list, test_data_list, feature_words):
    """根据feature_words将文本向量化

    Args:
        train_data_list: 训练集
        test_data_list: 测试集
        feature_words: 特征集

    Returns:
        train_feature_list - 训练集向量化列表 test_feature_list - 测试集向量化列表
    """
    def text_features(text, feature_words): #出现在特征集中, 则置1
        text_words = set(text)
        features = [1 if word in text_words else 0 for word in feature_words]
        return features

    train_feature_list = [text_features(text, feature_words) for text in train_data_list]
    test_feature_list = [text_features(text, feature_words) for text in test_data_list]
    return train_feature_list, test_feature_list #返回结果

def TextClassifier(train_feature_list, test_feature_list, train_class_list, test_class_list):
    """新闻分类器

    Args:
        train_feature_list - 训练集向量化的特征文本
        test_feature_list - 测试集向量化的特征文本
        train_class_list - 训练集分类标签
        test_class_list - 测试集分类标签

    Returns:
        test_accuracy - 分类器精度
    """
    #创建朴素贝叶斯对象
    model = MultinomialNB()
    #使用fit函数进行模型训练
    model.fit(train_feature_list, train_class_list)
    #调用score函数计算模型在测试集上的得分test_accuracy
    test_accuracy = model.score(test_feature_list, test_class_list)

    return test_accuracy

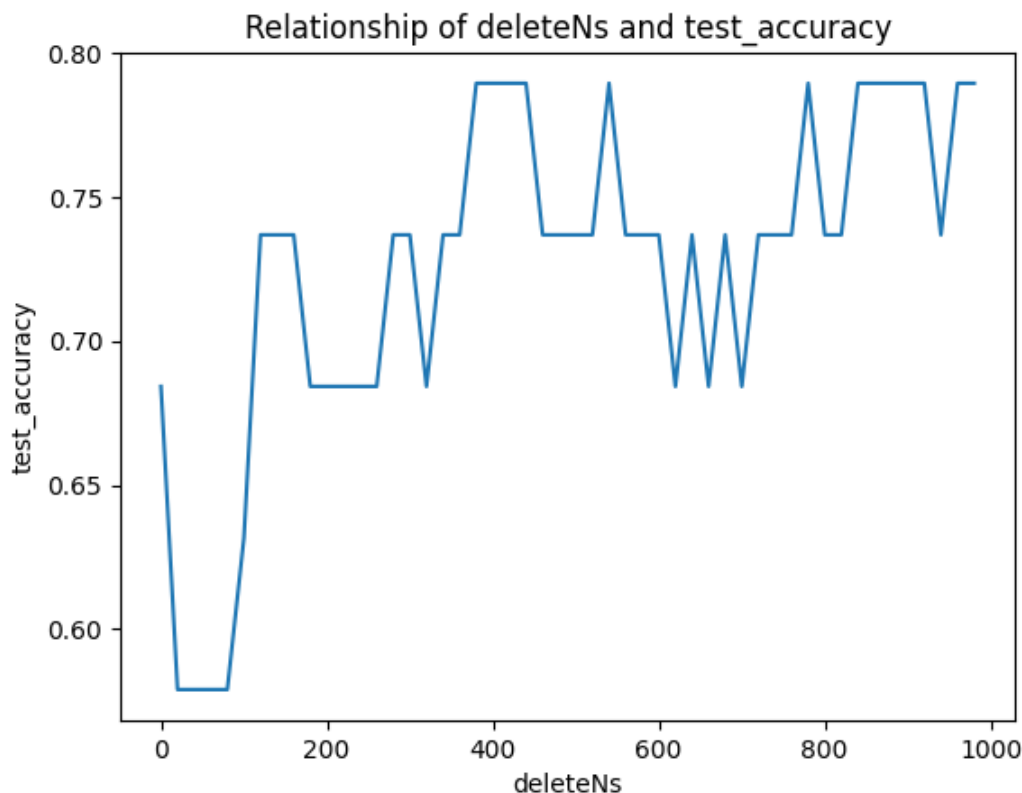
if __name__ == '__main__':
    #文本预处理
    folder_path = 'bayes_news/Sample' #训练集存放地址
    all_words_list, train_data_list, test_data_list, train_class_list, test_class_list = TextProcessing(folder_path, test_s

    # 生成stopwords_set
    stopwords_file = 'bayes_news/stopwords_cn.txt'
    stopwords_set = MakeWordsSet(stopwords_file)

    test_accuracy_list = []
    deleteNs = range(0, 1000, 20) #0 20 40 60 ... 980
    for deleteN in deleteNs:
        feature_words = words_dict(all_words_list, deleteN, stopwords_set)
        train_feature_list, test_feature_list = TextFeatures(train_data_list, test_data_list, feature_words)
        test_accuracy = TextClassifier(train_feature_list, test_feature_list, train_class_list, test_class_list)
        test_accuracy_list.append(test_accuracy)

    plt.figure()
    plt.plot(deleteNs, test_accuracy_list)
    plt.title('Relationship of deleteNs and test_accuracy')
    plt.xlabel('deleteNs')
    plt.ylabel('test_accuracy')
```

```
plt.show()
```



我们绘制出了deleteNs和test_accuracy的关系，这样我们就可以大致确定去掉前多少的高频词汇了。每次运行程序，绘制的图形可能不尽相同，我们可以通过多次测试，来决定这个deleteN的取值，然后确定这个参数，这样就可以顺利构建出用于新闻分类的朴素贝叶斯分类器了。

可以看到450还不错，最差的分类准确率也可以达到百分之50以上。

将if name == 'main'下的代码修改如下：

```
In [25]: if __name__ == '__main__':
#文本预处理
folder_path = 'bayes_news/Sample' #训练集存放地址
all_words_list, train_data_list, test_data_list, train_class_list, test_class_list = TextProcessing(folder_path, test_s

# 生成stopwords_set
stopwords_file = 'bayes_news/stopwords_cn.txt'
stopwords_set = MakeWordsSet(stopwords_file)

test_accuracy_list = []
feature_words = words_dict(all_words_list, 500, stopwords_set)
train_feature_list, test_feature_list = TextFeatures(train_data_list, test_data_list, feature_words)
test_accuracy = TextClassifier(train_feature_list, test_feature_list, train_class_list, test_class_list)
test_accuracy_list.append(test_accuracy)
ave = lambda c: sum(c) / len(c)

print(ave(test_accuracy_list))
```

0.7894736842105263

实验总结

通过本实验，您应该能达到以下两个目标：

- 1. 掌握朴素贝叶斯算法原理。
- 2. 熟悉朴素贝叶斯算法的初步应用。

参考文献及延伸阅读

参考资料:

- 1.哈林顿, 李锐. 机器学习实战 : Machine learning in action[M]. 人民邮电出版社, 2013.
- 2.周志华. 机器学习:Machine learning[M]. 清华大学出版社, 2016.

延伸阅读:

- 1.李航. 统计学习方法[M]. 清华大学出版社, 2012.