机器学习实验报告

学　　院：电子信息与通信学院

班　　级：电子信息工程 2101 班

姓　　名：林子垚

学　　号：U202113904

实验时间：2024 年 3 月 24 号

一、实验名称

贝叶斯模型

二、实验任务

1. 使用朴素贝叶斯过滤垃圾邮件

现有 50 封电子邮件，存放在数据集中，试基于朴素贝叶斯分类器原理，用 Python 编程实现对垃圾邮件和正常邮件的分类。采用交叉验证方式并且输出分类的错误率及分类错误的文档。

- 收集数据：这里数据直接提供，一般情况下需要自己手动采集及预分类；
- 数据管理：拿到手的数据并不能直接使用，我们需要对数据进行预处理，变成向量的形式（这里采用**词袋模型**）。可以首先把所有字符转换成小写，去掉大小字符不统一的影响；然后构建一个包含在所有文档中出现的不重复的词的列表；获得词汇表后，根据某个单词在一篇文档中出现的次数获得文档向量（这三步的代码可参考给出的 demo）；最后利用构建的分类器进行训练。
- 训练和测试：导入文件夹 spam 与 ham 下的文本文件，并将它们解析为词列表。接下来构建一个训练集和测试集，两个集合中的邮件都是随机选出的。经过一次迭代就能输出分类的错误率及分类错误的文档。（注：由于测试集的选择是随机的，所以测试算法时每次的输出结果可能有些差别，如果想要更好的估计错误率，最好将上述过程重复多次，然后求平均值。）

需要编写实验报告进行简述其原理，编程思路，数据集的划分方式以及错误率等。

2. 使用朴素贝叶斯对搜狗新闻语料库进行分类

2.1. 数据集讲解

- 数据集存放在 `Database` 中，新闻一共分为 9 个类：财经、IT、健康、体育、旅游、教育、招聘、文化、军事，不同种类的新闻分别放在相应代号的文件夹中。NBC.py 中，对新闻进行预处理的模块已经给出，请尝试补全分类器模块的代码（NBC.py 中函数 TextClassifier），自行划分训练集和测试集，对这些新闻进行训练和分类。

2.2. 提示

1. 数据预处理：对语料库中的文本进行分词，并通过词频找到特征词，从而生成相应的训练集和测试集。这部分代码我们已经给出，请尝试看懂并理解它们。

2. 机器学习库：我们鼓励你手写朴素贝叶斯分类器，但是在本任务中，我们允许你使用机器学习库 sklearn 快速实现分类器，以方便研究不同的参数设置对实验结果的影响，探讨影响朴素贝叶斯分类器分类效果的原因。

3. 分词工具：本任务需要你安装分词库 jieba。安装方法：pip install jieba。

提交要求：请尝试在不同的参数下多做几组实验：例如可以在构建词典中尝试删去不同个数的高频词，观察实验结果的变化；也可以在更改不同的特征词数量，观察对分类准确率的影响；或者更改训练集和测试集划分比例，观察不同训练集规模对实验结果的影响；也可以更换特征词的提取方式，如 TF-IDF 等等。编写实验报告，报告中需要包含：实验设置，编程思路，实验结果展示以及自己的思考等。

3. 使用朴素贝叶斯对电影评论分类

该数据集是 IMDB 电影数据集的一个子集，已经划分好了测试集和训练集，训练集包括 25000 条电影评论，测试集也有 25000 条，该数据集已经经过预处理，将每条评论的具体单词序列转化为词库里的整数序列，其中每个整数代表该单词在词库里的位置。例如，整数 104 代表该单词是词库的第 104 个单词。为实验简单，词库仅仅保留了 10000 个最常出现的单词，低频词汇被舍弃。每条评论都具有一个标签，0 表示为负面评论，1 表示为正面评论。

训练数据在 `train_data.txt` 文件下，每一行为一条评论，训练集标签在 `train_labels.txt` 文件下，每一行为一条评论的标签；测试数据在 `test_data.txt` 文件下，测试数据标签未给出。

3.1. 步骤提示

每个文档可以看成由n个特征构成的文档向量，每个单词表示一个特征维度。统计正样本数和负样本数，可以得到文档分布的先验概率；统计每类样本中，某个单词出现的次数和总单词数的比值，可以得到该特征的条件概率。

朴素贝叶斯用各个特征的条件概率连乘表示某个样本在某个类别的条件概率。然而，如果一个单词没有出现在某个类别的样本中，那么它的条件概率就是 0，导致最后的连乘结果也为 0，从而将不再有文档被分到这一类。因此在训练的过程中，注意使用拉普拉斯平滑处理。

3.2. 具体要求

将测试数据预测结果，与训练数据标签存储方式相同，存储为 txt 文件，每一行为一条评论的标签。将测试集预测结果的 txt 文件由刘青霞同学负责统一发给助教。实验报告中需要写明具体实验流程，思路。

三、开发环境

- **Python 3.8**

Python是一种高级、通用、解释型的编程语言，由Guido van Rossum于1991年创造。它在软件开发领域得到了广泛应用，包括Web开发、数据科学、人工智能、网络编程、自动化脚本等方面。

- **Anaconda**

Anaconda是一个开源的Python和R编程语言的发行版本，用于科学计算、数据科学、机器学习和大数据处理等领域。它包含了许多常用的工具、库和包，旨在简化数据分析和科学计算的流程。

- **scikit-learn 1.30**

scikit-learn是一个用于机器学习的Python库，提供了丰富且易于使用的工具，涵盖了各种常见的机器学习算法和数据处理功能。它建立在NumPy、SciPy和matplotlib之上，是Python生态系统中机器学习的主要组成部分之一。

- **pandas 2.30**

Pandas是一个Python库，提供了丰富的数据结构和数据分析工具，特别适用于数据清洗、数据处理、数据分析和数据可视化等任务。它构建在NumPy之上，因此具有快速、高效处理大型数据集的能力。

- **jieba 1.24.3**

jieba（"结巴"）是一个用于中文文本处理的 Python 库，主要用于中文分词。它采用了基于前缀词典构建有向无环图（DAG）的分词算法，结合了动态规划和词频统计，能够较好地处理各种文本中的中文分词任务。

四、实验过程

1. 使用朴素贝叶斯过滤垃圾邮件

本次实验，我使用了 sklearn，pandas 和 matplotlib 库来实现作业要求。

• 数据预处理

本次实验任务的数据预处理较为复杂，我单独开了一个 preprocess.py 文件专门处理数据。我使用 pandas 作为数据预处理的工具。其主函数如下

```
def preprocess():
    ham_mail_paths = find_mail_files('ham')
    spam_mail_paths = find_mail_files('spam')
    ham_strings = []
    spam_strings = []
    for ham_path in ham_mail_paths:
        ham_strings.append(read_txt_as_string(ham_path))
    for spam_path in spam_mail_paths:
        spam_strings.append(read_txt_as_string(spam_path))
    strings = ham_strings + spam_strings
    vocab_list = []
    for long_string in strings:
        vocab_list.append(textParse(long_string))
    vocab_list = createVocabList(vocab_list)

    ham_vectors = []
    spam_vectors = []
    for ham_string in ham_strings:
        ham_vectors.append(bagOfWords2VecMN(vocab_list, textParse(ham_string)))
    for spam_string in spam_strings:
        spam_vectors.append(bagOfWords2VecMN(vocab_list, textParse(spam_string)))
    ham_vectors = pd.DataFrame(ham_vectors)
    spam_vectors = pd.DataFrame(spam_vectors)
    return ham_vectors, spam_vectors
```

首先设计了 find_mail_files 函数，用于读取两种不同邮件的路径，代码如下。传入参数 path 用于指定路径名，然后建立一个空列表，用 for 循环找出指定路径下所有 txt 文件的路径，以 list 形式返回。

```python
def find_mail_files(path):
    mail_paths = []
    # 遍历当前目录下的所有文件和子目录
    for root, dirs, files in os.walk(path):
        for file in files:
            if file.endswith('.txt'):
                # 构建完整的文件路径并添加到列表中
                mail_paths.append(os.path.join(root, file))
    return mail_paths
```

在主函数中，分别传入参数 ham 和 spam 来获取两种邮件的所有邮件路径，然后调用函数 read_txt_as_string 来读取邮件。其代码如下。

```python
def read_txt_as_string(file_path):
    try:
        with open(file_path, 'r', encoding='cp1252') as file:
            content = file.read()
        return content
    except FileNotFoundError:
        print("File not found.")
        return None
```

通过这种方式，把得到的路径转换为了字符串列表。然后在数据预处理主函数中，先把两种邮件的字符串列表喝起来，然后调用实验任务提供的 textParse 函数和 createVocabList 函数，求出词汇表，记为 vocab_list

最后利用词汇表和 bagOfWord2VecMN 函数，将所有字符串转换为词袋表示，并以 pandas 提供的 DataFrame 格式返回以供主函数训练

- **训练**

在训练贝叶斯分类器的部分，我调用了 sklearn 库进行决策树的训练。具体代码如下

```
def read_txt_as_string(file_path):
    try:
        with open(file_path, 'r', encoding='cp1252') as file:
            content = file.read()
        return content
    except FileNotFoundError:
        print("File not found.")
        return None
```

首先将数据预处理中得到 ham 和 spam 进行标记，将 ham 标记为 0，将 spam 标记为 1，然后将 DataFrame 调用 train_test_split 进行训练集和测试集的划分。其中指定了参数 `test_size=0.2, random_state=0`。前者用于划分数据集中测试集的比例，后者用于控制随机性，保证可复制性。

然后创造一个决策树 `MultinomialNB` 实例 `clf`，并且进行训练，然后用测试集进行性能衡量。输出结果为“Ham Accuracy: 100.0 %，Spam Accuracy: 100.0 %”

- **性能**

经过测试，正确率为 100.0 %

2. 使用朴素贝叶斯对搜狗新闻语料库进行分类

• 数据预处理

由于数据预处理部分已经基本给出，因此不需要做大的修改。但是值得注意的是，题目给出的原始代码包含了一些 Python 2 的语法，这些语法在 Python 3 中已经被删除。因此我花了一些时间对这些用法进行修改。如对 str 进行 .decode()，对 zip 返回的迭代器求长度 len() 和

```
# Sen-Yao
def TextClassifier(train_feature_list, test_feature_list, train_class_list, test_class_list):
    """
    函数说明:分类器
    Parameters:
        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
    """

    clf = MultinomialNB()
    clf.fit(train_feature_list, train_class_list)
    test_accuracy = clf.score(test_feature_list, test_class_list)
    return test_accuracy
```

下标等。

• 训练结果和性能

有趣的是，在不同的训练-测试中，正确率表现出了较大的差异性。按照题目的默认设置，经过 100 次独立的训练-测试，发现平均正确率约为 64 %。

将测试集比例从 0.1 改变到 0.5，没有发现明显的性能变化。

将 deleteN 从 1 变化到 800，没有发现明显的性能变化。

将 feature_word 的维度从 100 变化到 2000，发现在维度小于 300 时，性能明显较差，当维度大于 500 后，正确率基本稳定。

综上得出结论，影响性能最明显的指标是 feature_word 的维度，如果维度过小，贝叶斯分类器可能无法提取到关于文本足够多的信息，导致影响分类的正确率。

3. IMDB 数据集电影评测分类（二分类问题）

本次实验，我使用了 sklearn，pandas, numpy 库来实现作业要求。

- 性能

我使用 pandas 作为数据预处理的工具。代码如下

```
def preprocess(data_path, **kwargs):
    print('正在读取数据')
    vectors = one_hot(data_path)
    if 'label_path' in kwargs:
        labels = []
        with open(kwargs['label_path'], 'r') as file:
            for line in file:
                labels.append(int(line))
        vectors = pd.DataFrame(vectors)
        vectors['Label'] = labels
    return vectors
```

其中，`one_hot` 函数为独热编码的过程

```
def one_hot(path, vector_size=10000):
    vectors = []
    with open(path, 'r') as file:
        for line in file:
            # 将每行按空格切分并转换为整数
            numbers = list(map(int, line.split()))
            # 创建一个长度为 10000 的零向量
            vector = np.zeros(10000, dtype=int)
            # 对每个数字进行编码
            for number in numbers:
                if 0 <= number < 10000:  # 确保索引在合理范围内
                    vector[number] += 1
            vectors.append(vector)
    return np.array(vectors)
```

此函数首先为每句话创建一个长度为 10000 的零向量，对应了词库中的 10000 个词。如果这句话中出现了某词汇，则在对应的下标加一。由此建立了句子到词向量的映射关系。将此映射关系以 Numpy 数组的形式返回。

然后数据预处理函数再读取 `train_label`，将其中的标签补充到刚刚得到的词向量上，并且命名为 `label`。由此完成了数据预处理部分。

• 训练

在训练贝叶斯分类器的部分，我调用了 sklearn 库进行贝叶斯分类器的训练。具体代码如下

这里设置了参数 `vaild` 来判断是否划分验证集，用于监控决策树性能。首先对数据预处理中得到的 DataFrame 进行分割，选出特征和标签。对于需要划分验证集的情况，默认的训练集和验证集的划分比例为 7:3。

```
def train(train_vectors, valid=False):
    start_time = time.time()
    acc = None  # 为了在函数最后能返回acc，要保证acc被定义

    X = train_vectors.iloc[:, :train_vectors.shape[1] - 1]
    y = train_vectors.iloc[:, train_vectors.shape[1] - 1]

    if valid:
        x_train, x_valid, y_train, y_valid = train_test_split(X, y, test_size=0.3, random_state=42)

        clf = MultinomialNB()
        clf.fit(x_train, y_train)
        y_pred = clf.predict(x_valid)
        acc = accuracy_score(y_valid, y_pred)
        print('正确率为', 100 * acc, '%')
    else:
        clf = MultinomialNB()
        clf.fit(X, y)

    end_time = time.time() - start_time
    print('训练耗时:', end_time, 's')
    return clf, acc
```

然后创造一个决策树 `DecisionTreeClassifier` 实例 `clf`，并且进行训练，然后用测试集进行性能衡量。

• 主函数

综上，本次实验的代码的主函数为

```
def main():
    train_data = preprocess('train_data.txt', label_path='train_labels.txt')
    test_data = preprocess('test_data.txt')
    clf, _ = train(train_data, True)
    test_y_pred = clf.predict(test_data)
    np.savetxt('test_predictions.txt', test_y_pred, fmt='%d')
```

即先分别读取训练数据，训练标签和测试集数据，然后进行相应的数据预处理函数。调用 `train` 函数进行训练，并输出正确率，最后应用训练得到的决策树，对给定的测试集数据进行预测，最后将结果保存在 `test_predictions.txt`

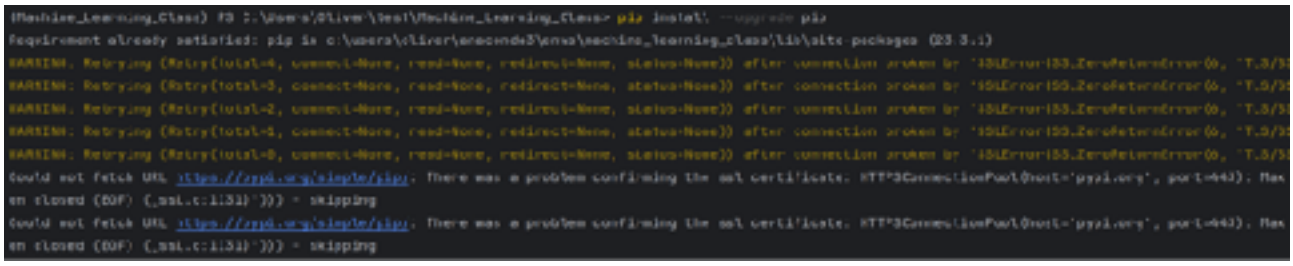
- **性能**

经过测试，决策树在验证集上的正确率约为 83 %

七、实验小结

在本次实验中，我学会了使用贝叶斯模型实现简单的功能。对于不同形式和格式的数据集，我也学会了利用 Numpy, pandas 等各种常用 Python 库，将其转换为模型容易接受的数据形式。经过“任务分析-数据预处理-模型选择-训练模型-反复调优-提交模型”的机器学习解决问题过程，将自己在《机器学习》的贝叶斯模型和模型评估中学到的理论知识应用到了实践中，这给我的成就感非常大。我将我的代码放在了 GitHub 上。https://github.com/Sen-Yao/Machine_Learning_Class

但与此同时，也不可避免的遇到了一些问题。第二个任务中，在尝试安装 jieba 时遇到了某种 SSL 错误，如下



最后通过调整网络配置得以解决。此外，还遇到了给出的源代码用到了大量早期 Python 版本特有的语法，因此遇到了一些兼容性问题，处理这一部分也花费了一些时间。

最后，任务二通过调整各个参数，观察到了参数对训练性能的影响。很主观的感受到了特征长度对于一个样本提供的信息多少有着重要意义。