中山大学计算机学院

人工智能

本科生实验报告

（2023 学年春季学期）

课程名称：Artificial Intelligence

教学班级	**刘咏梅老师班级**	专业（方向）	**信息与计算科学**
学号	**22336313**	姓名	**郑鸿鑫**

一、 实验题目

- 二选一实现：k-NN 分类器、朴素贝叶斯分类器
- 在给定文本数据集完成文本情感分类训练，在测试集完成测试，计算准确率。

实验要求：

- 不可直接调用机器学习库中的分类器算法（仅可用于和自己的方法对比准确率）
- 可使用各种提取文本特征的辅助工具，如 OneHotEncoder、 TfidfVectorizer 等。

二、 实验内容

1. 算法原理

本实验采用 K-NN 分类器作为实现的方法，故分析 K-NN 的算法原理：

- 原理：k-NN 算法是一种基于实例的学习方法，它通过计算测试样本与训练样本之间的距离来进行分类。对于每个测试样本，找到与其距离最近的 k 个训练样本，然后根据这 k 个样本的类别进行投票来确定测试样本的类别。
- 步骤：
  - 将训练数据转换为特征向量表示（如 TF-IDF 特征）。

- 计算测试样本与每个训练样本之间的距离（可以使用闵氏距离或余弦相似度）。
- 根据距离找到最近的 k 个训练样本。
- 根据这 k 个样本的类别进行投票，选择票数最多的类别作为测试样本的类别。

2. 伪代码

算法伪代码如下：

```
Procedure preprocess(text)
    Begin
        Return text Converted To Lowercase
    EndProcedur
Procedure knn_predict(X_test, k)
    Begin
        num_correct = 0
        For i = 0 To Length(X_test) - 1
            test_vector = X_test[i]
            distances = Euclidean Distance Between X_train And test_vector
            nearest_indices = Sort(distances) And Get First k Elements
            nearest_labels = Get labels Corresponding To nearest_indices
            predicted_label = Most Frequent Label In nearest_labels
            If predicted_label Equals test_labels[i]
                Predict successfly.
                num_correct = num_correct + 1
            Else
                Predict failed.
            EndIf
        EndFor
        Return num_correct / Length(X_test)
    EndProcedure
Procedure Main()
    Begin
        // 读取和预处理训练数据
        Initialize train_data And labels As Empty Lists
        For Each line In 'train.txt'
            Append preprocess(line.split(' ')[2]) To train_data
            Append line.split(' ')[1] To labels
        EndFor
        // 读取和预处理测试数据
```

```
        Initialize test_data And test_labels As Empty Lists
        For Each line In 'test.txt'
            Append preprocess(line.split(' ')[2]) To test_data
            Append line.split(' ')[1] To test_labels
        EndFo
        // 特征提取
        Initialize vectorizer With TfidfVectorizer(ngram_range=(1, 2), max_features=500)
        X_train = vectorizer.fit_transform(train_data).toarray()
        X_test = vectorizer.transform(test_data).toarray()
        // 评估模型
        accuracy = knn_predict(X_test, k=3)
        Print "Accuracy: ", accuracy * 100, "%"
    EndProcedure
```

3. 关键代码展示（带注释）

a. 预处理函数，将文本全部转化为小写字母，以便后续识别单词不受干扰：

```
def preprocess(text):
    # Basic preprocessing to lowercase text
    return text.lower()
#将文本数据转化为小写字母
```

b. 打开文件处理训练数据和测试数据：

```
# Read and preprocess data
with open('train.txt', 'r') as file:
    lines = file.readlines()
    train_data = [preprocess(line.strip().split(' ', 2)[-1]) for
line in lines]
    #存储训练数据
    labels = [line.strip().split(' ')[1] for line in lines]
    #存储感情标签，用于训练分类器
with open('test.txt', 'r') as file:
    lines = file.readlines()
    test_data = [preprocess(line.strip().split(' ', 3)[-1]) for
line in lines]
    #存储测试数据
    test_labels = [line.strip().split(' ')[1] for line in lines]
    #存储测试数据的情感标签，用于比对预测结果是否正确
```

c. 处理文本为 TF-IDF 特征矩阵：

```
#使用 TdidfVectorizer 将文本数据为 TF-IDF 特征矩阵
vectorizer = TfidfVectorizer(ngram_range=(1, 2),
max_features=500)
#ngram_range=(1, 2) 表示考虑从单个词（unigram）到两个词（bigram）的 n-
gram
#max_features=500 表示选择最多的特征数为 500。
X_train = vectorizer.fit_transform(train_data).toarray()
X_test = vectorizer.transform(test_data).toarray()
# X_train 是转换后的训练数据特征矩阵。
# X_test 是转换后的测试数据特征矩阵
```

d. 训练并测试数据：

```
#K-NN 情感分类器
def knn_predict(X_test, k=5):
    num_correct = 0 #计算正确预测的次数
    for i, test_vector in enumerate(X_test):
        distances = np.linalg.norm(X_train - test_vector, axis=1)
        # 计算训练集与测试向量之间的欧氏距离。
        nearest_indices = np.argsort(distances)[:k]
        #获取最近 k 个训练样本的索引
        nearest_labels = [labels[idx] for idx in nearest_indices]
        #存储这些的邻居的标签
        predicted_label = max(set(nearest_labels),
key=nearest_labels.count)
        #根据多数投票原则预测的标签
        if predicted_label == test_labels[i]:
            print('第',i+1,'组数据，结果正确',"预测标签为
",predicted_label,'正确标签为',test_labels[i],'数据为',test_data[i])
            num_correct += 1
        else: print('第',i+1,'组数据，结果错误',"预测标签为
",predicted_label,'正确标签为',test_labels[i],'数据为',test_data[i])
    return num_correct / len(X_test)
# Evaluate model
accuracy = knn_predict(X_test)
print(f'正确率: {正确率* 100:.2f}%')
```

4. 创新点&优化

在 k-NN 算法中，k 是一个参数，代表在进行分类决策时考虑的最近邻居的数量。具体来说，先给出本实验中的 k 值的定义如下：

k 值：这个数字 k 表示在预测一个新样本的类别时，会考虑距离该样本最近的 k 个训练样本（邻居）。在 k-NN 算法中，一个新样本的类别是通过对这些邻居的类别进行投票或计算加权平均来确定的。

在上述代码中，k 默认为 5，我们尝试通过修改 k 的值来提高算法的分类能力，详见下文实验结果分析中的评测指标及分析。

5. 实验结果及分析

1. 实验结果展示示例

此处只给出部分测试结果，完整的结果详见 Result 文件夹：

```
第 990 组数据，结果错误 预测标签为 4 正确标签为 5 数据为 u s onlin love broker ey china
第 991 组数据，结果错误 预测标签为 5 正确标签为 1 数据为 u s divert troop to fight taliban
第 992 组数据，结果正确 预测标签为 5 正确标签为 5 数据为 hous of card actor ian richardson dead
第 993 组数据，结果正确 预测标签为 5 正确标签为 5 数据为 taliban leader kill in airstrik
第 994 组数据，结果正确 预测标签为 4 正确标签为 4 数据为 escap to pragu without the summer hord
第 995 组数据，结果错误 预测标签为 4 正确标签为 6 数据为 kathmandu first snow in year
第 996 组数据，结果错误 预测标签为 4 正确标签为 5 数据为 nasdaq fail in bid for lse
第 997 组数据，结果正确 预测标签为 5 正确标签为 5 数据为 ex pastor get death sentenc
第 998 组数据，结果错误 预测标签为 5 正确标签为 6 数据为 babi born on turnpik after dad miss exit
第 999 组数据，结果错误 预测标签为 4 正确标签为 6 数据为 studi link chimp and hammer
第 1000 组数据，结果正确 预测标签为 4 正确标签为 4 数据为 un googl earth map climat chang
正确率：38.00%

进程已结束，退出代码为 0
```

2. 评测指标展示及分析

对于每次的测试，我们进行十次测试并记录准确率的最大最小值，并计算平均准确值：

小 k 值：当 k 较小时，模型对噪声和异常值比较敏感，容易过拟合。也就是说，如果 k 过小，分类器可能会对训练数据中的随机误差进行过度拟合，从而影响其泛化能力。

大 k 值：当 k 较大时，模型的复杂度降低，容易欠拟合。如果 k 过大，分类器可能会忽视数据中的一些重要特征，导致它对新数据的分类不够敏感。

- 以 k = 3 测试十次结果为：

  平均正确率：31.96 %　准确率最大值：34.8% 准确率最小值：29.6%

- 以 k = 5 测试十次结果为：

  平均准确率：34.7% 准确率最大值：35.5% 准确率最小值：33.8%

- 以 k = 7 测试十次结果为：

  平均正确率：37.05% 准确率最大值：37.8% 准确率最小值：35.5%

- 以 k = 9 测试十次结果为：

  平均正确率：36.52% 准确率最大值：38% 准确率最小值：35.3%

- 以 k = 11 测试十次结果为：

  平均正确率：36.17% 准确率最大值：36.7% 准确率最小值：34.8%

由此可看出，k = 7 为测试中较好的一个 k 值，可以使平均准确率达到 37%以上。

对于每次使用相同样本训练和测试仍会出现结果略有不同的解释：

由于 KNN 是一种基于实例的学习方法，它对训练数据的具体分布和密度非常敏

感。因此，即使使用相同的训练和测试数据，由于 KNN 算法的随机性和对邻居的选择敏感性，可能会导致在每次运行时得到略有不同的结果。

6. 参考资料

1. 【人工智能-神经网络】Numpy 实现单层感知机对情感文本多分类_感知机实现文本分类-CSDN 博客
2. sklearn: TfidfVectorizer 中文处理及一些使用参数_tfidfvectorizer 的参数 vocabulary_-CSDN 博客
3. sklearn 文本特征提取——TfidfVectorizer - 冬色 - 博客园 (cnblogs.com)
4. 一个例子来使用 sklearn 中的 TfidfVectorizer_tfidvectorizer 单汉字-CSDN 博客
5. np.linalg.norm()用法总结-CSDN 博客