

南京大学

实验报告

姓 名： 华广松

日 期： 2021 年 11 月 20 日

1. 实验内容

本实验将会从电影题材分类的例子入手，讲述 K 近邻算法的原理。在这之后，我们将会使用该算法实现手写数字识别系统

2. 实验目的

学习使用 K 近邻算法并实现手写数字识别

3. 算法描述

简单地说，K 近邻算法采用测量不同特征值之间的距离方法进行分类。

4. 算法实现

```
5. import numpy as np
import operator
from os import listdir

def createDataSet():
    group = np.array([[1.0, 1.1], [1.0, 1.0], [0, 0], [0, 0.1]])
    labels = ['A', 'A', 'B', 'B']
    return group, labels

def img2vector(filename):
    # 创建向量
    returnVect = np.zeros((1, 1024))
    # 打开数据文件，读取每行内容
    fr = open(filename)
    for i in range(32):
        # 读取每一行
        lineStr = fr.readline()
        # 将每行前 32 字符转成 int 存入向量
        for j in range(32):
            returnVect[0, 32 * i + j] = int(lineStr[j])
```

```

        return returnVect

def classify0(inX, dataSet, labels, k):
    """
    参数:
    - inX: 用于分类的输入向量
    - dataSet: 输入的训练样本集
    - labels: 样本数据的类标签向量
    - k: 用于选择最近邻居的数目
    """

    # 获取样本数据数量
    dataSetSize = dataSet.shape[0]

    # 矩阵运算，计算测试数据与每个样本数据对应数据项的差值
    diffMat = np.tile(inX, (dataSetSize, 1)) - dataSet

    # sqDistances 上一步骤结果平方和
    sqDiffMat = diffMat ** 2
    sqDistances = sqDiffMat.sum(axis=1)

    # 取平方根，得到距离向量
    distances = sqDistances ** 0.5

    # 按照距离从低到高排序
    sortedDistIndicies = distances.argsort()
    classCount = {}

    # 依次取出最近的样本数据
    for i in range(k):
        # 记录该样本数据所属的类别
        voteIlabel = labels[sortedDistIndicies[i]]
        classCount[voteIlabel] = classCount.get(voteIlabel, 0) + 1

    # 对类别出现的频次进行排序，从高到低
    sortedClassCount = sorted(
        classCount.items(), key=operator.itemgetter(1), reverse=True)

    # 返回出现频次最高的类别
    return sortedClassCount[0][0]

def handwritingClassTest():

```

```

# 样本数据的类标签列表
hwLabels = []

# 样本数据文件列表
trainingFileList = listdir('digits/trainingDigits')
m = len(trainingFileList)

# 初始化样本数据矩阵 (M*1024)
trainingMat = np.zeros((m, 1024))

# 依次读取所有样本数据到数据矩阵
for i in range(m):
    # 提取文件名中的数字
    fileNameStr = trainingFileList[i]
    fileStr = fileNameStr.split('.')[0]
    classNumStr = int(fileStr.split('_')[0])
    hwLabels.append(classNumStr)

    # 将样本数据存入矩阵
    trainingMat[i, :] = img2vector(
        'digits/trainingDigits/%s' % fileNameStr)

# 循环读取测试数据
testFileList = listdir('digits/testDigits')

# 初始化错误率
errorCount = 0.0
mTest = len(testFileList)

# 循环测试每个测试数据文件
for i in range(mTest):
    # 提取文件名中的数字
    fileNameStr = testFileList[i]
    fileStr = fileNameStr.split('.')[0]
    classNumStr = int(fileStr.split('_')[0])

    # 提取数据向量
    vectorUnderTest = img2vector('digits/testDigits/%s' %
        fileNameStr)

    # 对数据文件进行分类
    classifierResult = classify0(vectorUnderTest, trainingMat,
        hwLabels, 3)

```

```

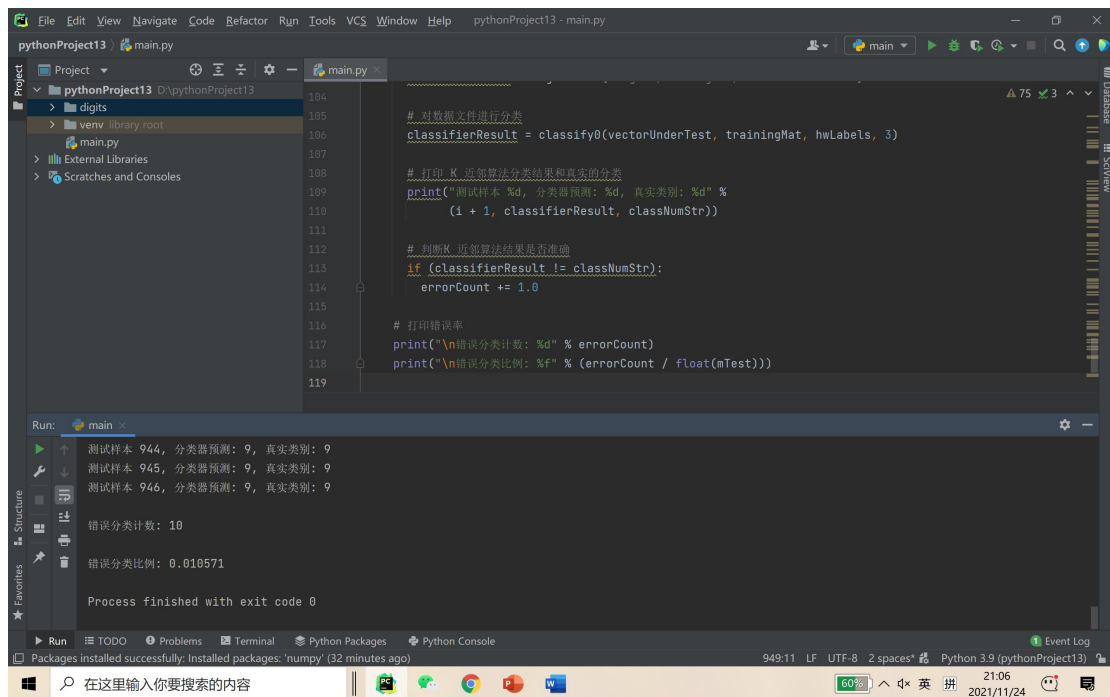
# 打印 K 近邻算法分类结果和真实的分类
print("测试样本 %d, 分类器预测: %d, 真实类别: %d" %
      (i + 1, classifierResult, classNumStr))

# 判断 K 近邻算法结果是否准确
if (classifierResult != classNumStr):
    errorCount += 1.0

# 打印错误率
print("\n 错误分类计数: %d" % errorCount)
print("\n 错误分类比例: %f" % (errorCount / float(mTest)))

```

6. 程序运行结果



```

# 对数据文件进行分类
classifierResult = classify0(vectorUnderTest, trainingMat, hwLabels, 3)

# 打印 K 近邻算法分类结果和真实的分类
print("测试样本 %d, 分类器预测: %d, 真实类别: %d" %
      (i + 1, classifierResult, classNumStr))

# 判断 K 近邻算法结果是否准确
if (classifierResult != classNumStr):
    errorCount += 1.0

# 打印错误率
print("\n 错误分类计数: %d" % errorCount)
print("\n 错误分类比例: %f" % (errorCount / float(mTest)))

```

Run: main

```

测试样本 944, 分类器预测: 9, 真实类别: 9
测试样本 945, 分类器预测: 9, 真实类别: 9
测试样本 946, 分类器预测: 9, 真实类别: 9

错误分类计数: 10
错误分类比例: 0.010571

Process finished with exit code 0

```

上面的代码中，将 `trainingDigits` 目录中的文件内容存储在列表中，然后可以得到目录中有多少文件，并将其存储在变量 `m` 中。接着，代码创建一个 `m` 行 1024 列的训练矩阵，该矩阵的每行数据存储一个图像。

我们可以从文件名中解析出分类数字。该目录下的文件按照规则命名，如文件 `9_45.txt` 的分类是 9，它是数字 9 的第 45 个实例。然后我们可以将类代码存储在 `hwLabels` 向量中，使用前面讨论的 `img2vector` 函数载入图像。

在下一步中，我们对 `testDigits` 目录中的文件执行相似的操作，不同之处是我们并不将这个目录下的文件载入矩阵中，而是使用 `classify0()` 函数测试该目录下的每个文件。

7. 实验结果分析

K 近邻算法识别手写数字数据集，错误率为 1.05%。改变变量 k 的值、修改函数 `handwritingClassTest` 随机选取训练样本、改变训练样本的数目，都会对 K 近邻算法的错误率产生影响

8. 结论

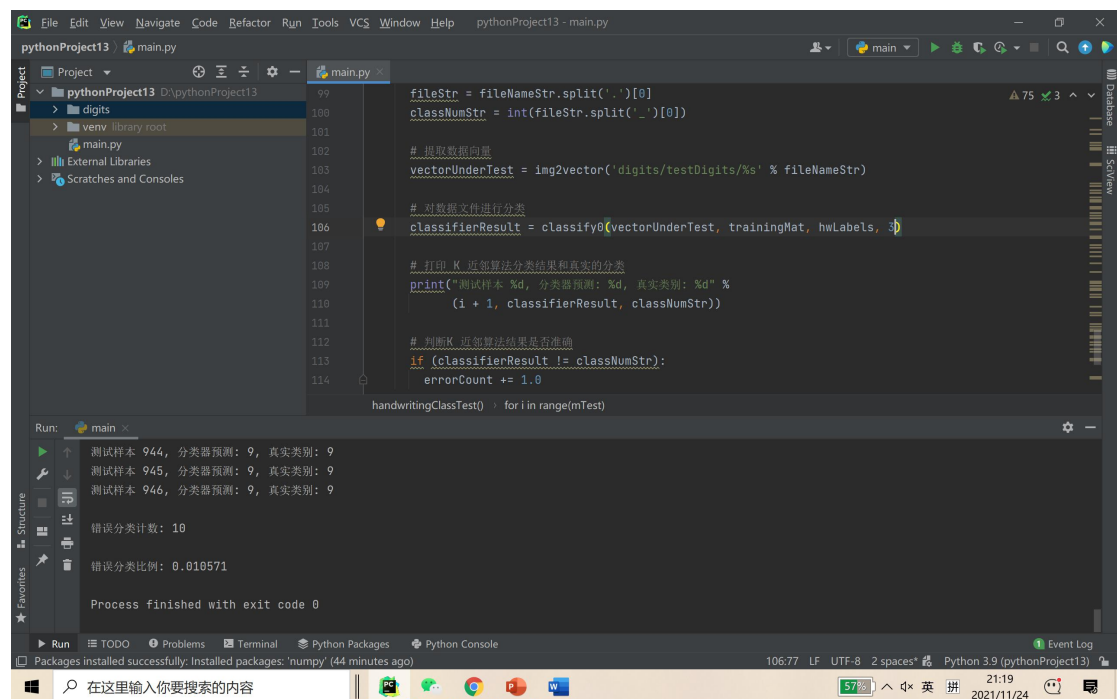
K 近邻算法是分类数据最简单有效的算法。K 近邻算法是基于实例的学习，使用算法时我们必须有接近实际数据的训练样本数据。K 近邻算法必须保存全部数据集，如果训练数据集很大，必须使用大量的存储空间。此外，由于必须对数据集中的每个数据计算距离值实际使用是可能非常耗时。是否存在一种算法减少存储空间和计算时间的开销呢？K 决策树就是 K 近邻算法的优化版，可以节省大量的计算开销。

9. 探索

K 邻近算法的使用场景：文本分类，多分类领悟，模式识别

改变 k 的值，观察错误率的变化：

$K = 3$:



```
pythonProject13 - main.py
pythonProject13 D:\pythonProject13
> digits
> venv library root
> main.py
> External Libraries
> Scratches and Consoles

99 fileStr = fileNameStr.split('.')[0]
100 classNumStr = int(fileNameStr.split('.')[0])
101
102 # 提取数据向量
103 vectorUnderTest = img2vector('digits/testDigits/%s' % fileNameStr)
104
105 # 对数据进行分类
106 classifierResult = classify0(vectorUnderTest, trainingMat, hwLabels, 3)
107
108 # 打印 K 近邻算法分类结果和真实的分类
109 print("测试样本 %d, 分类器预测: %d, 真实类别: %d" %
110       (i + 1, classifierResult, classNumStr))
111
112 # 判断 K 近邻算法结果是否准确
113 if (classifierResult != classNumStr):
114     errorCount += 1.0

handwritingClassTest() for i in range(mTest)

Run: main
测试样本 944, 分类器预测: 9, 真实类别: 9
测试样本 945, 分类器预测: 9, 真实类别: 9
测试样本 946, 分类器预测: 9, 真实类别: 9
错误分类计数: 10
错误分类比例: 0.010571
Process finished with exit code 0

Packages installed successfully: Installed packages: 'numpy' (44 minutes ago)
106.77 LF UTF-8 2 spaces* Python 3.9 (pythonProject13)
21:19 2021/11/24
```

$K = 4$:

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help pythonProject13 - main.py
pythonProject13 main.py
Project
  pythonProject13
    digits
    venv library root
    main.py
    External Libraries
    Scratches and Consoles
Run: main
测试样本 938, 分类器预测: 9, 真实类别: 9
测试样本 939, 分类器预测: 9, 真实类别: 9
测试样本 940, 分类器预测: 9, 真实类别: 9
测试样本 941, 分类器预测: 9, 真实类别: 9
测试样本 942, 分类器预测: 9, 真实类别: 9
测试样本 943, 分类器预测: 9, 真实类别: 9
测试样本 944, 分类器预测: 9, 真实类别: 9
测试样本 945, 分类器预测: 9, 真实类别: 9
测试样本 946, 分类器预测: 9, 真实类别: 9
错误分类计数: 11
错误分类比例: 0.011628
Process finished with exit code 0
Packages installed successfully: Installed packages: 'numpy' (45 minutes ago)
954:1 LF UTF-8 2 spaces Python 3.9 (pythonProject13)
```

K = 6:

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help pythonProject13 - main.py
pythonProject13 main.py
Project
  pythonProject13
    digits
    venv library root
    main.py
    External Libraries
    Scratches and Consoles
Run: main
测试样本 938, 分类器预测: 9, 真实类别: 9
测试样本 939, 分类器预测: 9, 真实类别: 9
测试样本 940, 分类器预测: 9, 真实类别: 9
测试样本 941, 分类器预测: 9, 真实类别: 9
测试样本 942, 分类器预测: 9, 真实类别: 9
测试样本 943, 分类器预测: 9, 真实类别: 9
测试样本 944, 分类器预测: 9, 真实类别: 9
测试样本 945, 分类器预测: 9, 真实类别: 9
测试样本 946, 分类器预测: 9, 真实类别: 9
错误分类计数: 17
错误分类比例: 0.017970
Process finished with exit code 0
Packages installed successfully: Installed packages: 'numpy' (46 minutes ago)
106:77 LF UTF-8 2 spaces Python 3.9 (pythonProject13)
```

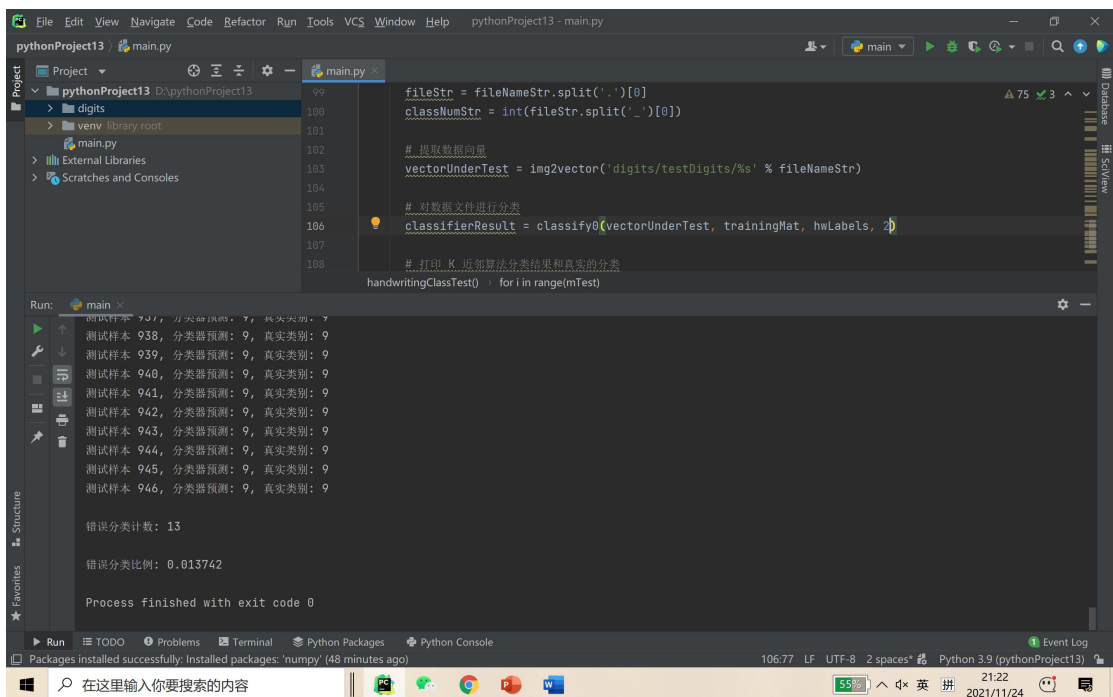
K = 10:

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help pythonProject13 - main.py
pythonProject13 main.py
Project
  pythonProject13
    digits
    venv library root
    main.py
    External Libraries
    Scratches and Consoles
Run: main
  测试样本 938, 分类器预测: 9, 真实类别: 9
  测试样本 939, 分类器预测: 9, 真实类别: 9
  测试样本 940, 分类器预测: 9, 真实类别: 9
  测试样本 941, 分类器预测: 9, 真实类别: 9
  测试样本 942, 分类器预测: 9, 真实类别: 9
  测试样本 943, 分类器预测: 9, 真实类别: 9
  测试样本 944, 分类器预测: 9, 真实类别: 9
  测试样本 945, 分类器预测: 9, 真实类别: 9
  测试样本 946, 分类器预测: 9, 真实类别: 9
  错误分类计数: 19
  错误分类比例: 0.02085
  Process finished with exit code 0
Packages installed successfully: Installed packages: 'numpy' (46 minutes ago)
106.78 LF UTF-8 2 spaces Python 3.9 (pythonProject13)
```

K = 20:

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help pythonProject13 - main.py
pythonProject13 main.py
Project
  pythonProject13
    digits
    venv library root
    main.py
    External Libraries
    Scratches and Consoles
Run: main
  测试样本 938, 分类器预测: 9, 真实类别: 9
  测试样本 939, 分类器预测: 9, 真实类别: 9
  测试样本 940, 分类器预测: 9, 真实类别: 9
  测试样本 941, 分类器预测: 9, 真实类别: 9
  测试样本 942, 分类器预测: 9, 真实类别: 9
  测试样本 943, 分类器预测: 9, 真实类别: 9
  测试样本 944, 分类器预测: 9, 真实类别: 9
  测试样本 945, 分类器预测: 9, 真实类别: 9
  测试样本 946, 分类器预测: 9, 真实类别: 9
  错误分类计数: 26
  错误分类比例: 0.027484
  Process finished with exit code 0
Packages installed successfully: Installed packages: 'numpy' (47 minutes ago)
106.78 LF UTF-8 2 spaces Python 3.9 (pythonProject13)
```

K = 2:



```
pythonProject13 - main.py
pythonProject13 \main.py
Project
  pythonProject13
    digits
    venv \library root
    main.py
    External Libraries
    Scratches and Consoles
Run: main
测试样本 938, 分类器预测: 9, 真实类别: 9
测试样本 939, 分类器预测: 9, 真实类别: 9
测试样本 940, 分类器预测: 9, 真实类别: 9
测试样本 941, 分类器预测: 9, 真实类别: 9
测试样本 942, 分类器预测: 9, 真实类别: 9
测试样本 943, 分类器预测: 9, 真实类别: 9
测试样本 944, 分类器预测: 9, 真实类别: 9
测试样本 945, 分类器预测: 9, 真实类别: 9
测试样本 946, 分类器预测: 9, 真实类别: 9
错误分类计数: 13
错误分类比例: 0.013742
Process finished with exit code 0
Packages installed successfully: Installed packages: 'numpy' (48 minutes ago)
106.77 LF UTF-8 2 spaces* Python 3.9 (pythonProject13)
```

fileStr = fileNameStr.split('.')[0]
classNumStr = int(fileNameStr.split('_')[0])
提取数据向量
vectorUnderTest = img2vector('digits/testDigits/%s' % fileNameStr)
对数据文件进行分类
classifierResult = classify0(vectorUnderTest, trainingMat, hmLabels, 2)
打印 K 近邻算法分类结果和真实的分类
handwritingClassTest() for i in range(mTest)

可见，随着 k 值的增大，其错误率不断上升直至平缓

实验感受与收获：

两个实验都使我受益匪浅，让我对机器学习有了更加深入的了解，同时，我感受到了编码的神奇之处，原来看似特别神奇的程序背后是工程师们精巧的计算与设计。像 numpy 包等工具，无不饱含程序设计师们辛勤的汗水与智慧，辛苦了。