KNN 算法识别手写数字

——实验报告

作者：罗海王

2020 年 3 月 16 日星期一

项目代码地址：https://github.com/Neptune-Haiwang/AliPythonDataAnalysis/blob/master/Week2/KNN_classifier2.py

项目说明文档：https://github.com/Neptune-Haiwang/AliPythonDataAnalysis/blob/master/Week2/README_Writen_Number_Recognition.md

目　　录

一、实验目的及要求

- 基础算法：KNN 算法学习，Python sklearn 环境搭建
- 场景建模：了解手写数字的场景，问题定义，以及如何使用 KNN 解决该问题
- 算法评估：了解如何评估一个算法的性能

二、实验的理论知识与算法模型

2.1 KNN 算法定义

- k-近邻（k-Nearest Neighbour，简称 KNN），常用于有监督学习。
- 如果一个样本 x 在特征空间中的 K 个最相似的（即特征空间中最邻近）的样本大多属于类别 A, 则该样本 x 也属于类别 A。

2.2 KNN 算法核心思想

- 根据你的'邻居'来推断你的类别
- 整个计算过程分为三步：
    - 计算待分类物体与其他物体之间的距离：一般用欧式距离，即平方差距离
    - 统计距离最近的 K 个邻居；
    - 对于 K 个最近的邻居，它们属于哪个分类最多，待分类物体就属于哪一类

2.3 KNN 算法的流程

- 获取数据
- 数据集划分
- 数据预处理、特征工程
    - 通过特定的统计方法（数学方法）将数据转换成算法要求的数据
    - 无量纲化：把不同规格的数据转换到同一规格：
        - 标准化：把原始数据变换到均值为 0，方差为 1 的范围内
        - 归一化：对原始数据进行变换，把数据映射到 [0, 1] 之间
- 构建 KNN 算法的模型
- 评估算法的性能效果
    - 调整 KNeighborsClassifier() 函数的超参数，查看模型预测准确率的变化，从而找出最优的模型参数。

三、基础实验过程展示

Note: 以下内容的伪代码，为步骤分析时从完整项目代码中提取出来的实现特定功能的代码。

3.1 获取数据

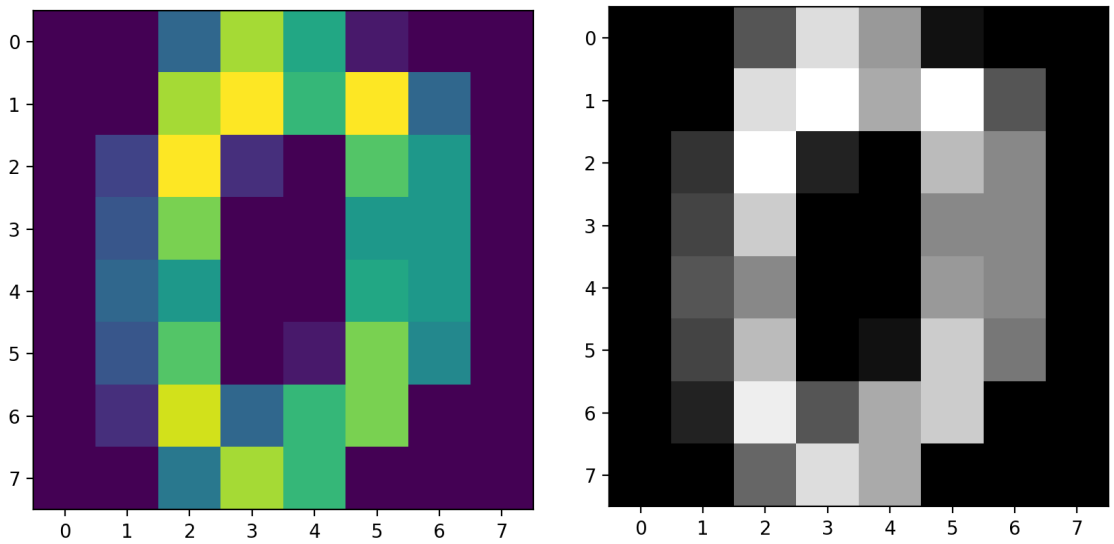
- `from sklearn.datasets import load_digits`

- `digits = load_digits()`
- *`# 展示图片（以第一张图片为例）`*

```
import matplotlib.pyplot as plt
plt.imshow(digits.images[0])
plt.show()
```

- *`# 也可以改变参数，图形以灰度图显示`*

```
# plt.imshow(digits.images[0], cmap='gray')
# plt.show()
```



3.2 数据集划分

- `from sklearn.model_selection import train_test_split`

- *`# 对训练集特征值，目标值、测试集的特征值和目标值的划分，另外设定了测试样本比例为 30%， 随机状态为2。`*
- *`# random_state 相当于随机数种子random.seed()，这里的random_state 就是为了保证程序每次运行都分割一样的训练集和测试集。否则，同样的算法模型在不同的训练集和测试集上的效果不一样。`*

- `x_train, x_test, y_train, y_test = train_test_split(digits.data, digits.target, test_size=0.3, random_state=2)`

3.3 特征工程：标准化

```
from sklearn.preprocessing import StandardScaler
transfer = StandardScaler()

# 对训练集进行标准化: fit 在做计算， transform 在做转换
x_train = transfer.fit_transform(x_train)
# 用训练集的平均值和标准差对测试集进行标准化
x_test = transfer.transform(x_test)
```

3.4 构建基础预测模型

```
from sklearn.neighbors import KNeighborsClassifier

estimator = KNeighborsClassifier(n_neighbors=3)
# 把训练集特征值和目标值丢给 KNN 算法 进行计算，训练
estimator.fit(x_train, y_train)
```

3.5 基础预测模型的评估

```
# 方法1：直接比对真实值和预测值
y_predict = estimator.predict(x_test)
print('真实值: %s 和预测值: %s ， 正确与否：%s' % (y_test, y_predict,
y_test == y_predict))

# 方法2：计算准确率 ： 特征值，目标值
score1 = estimator.score(x_train, y_train)
score2 = estimator.score(x_test, y_test)
print('训练集的准确率: %s， 而测试集的准确率为：%s' % (score1, score2))
```

3.6 基础预测模型的输出结果

```
真实结果:
[2 8 2 6 6 7 1 9 8 5 2 8 6 6 6 6 1 0 5 8 8 7 8 4 7 5 4 9 2 9 4 7 6 8 9 4 3
 1 0 1 8 6 7 7 1 0 7 6 2 1 9 6 7 9 0 0 5 1 6 3 0 2 3 4 1 9 2 6 9 1 8 3 5 1
 2 8 2 2 9 7 2 3 6 0 5 3 7 5 1 2 9 9 3 1 7 7 4 8 5 8 5 5 2 5 9 0 7 1 4 7 3
 4 8 9 7 9 8 2 6 5 2 5 8 4 8 7 0 6 1 5 9 9 9 5 9 9 5 7 5 6 2 8 6 9 6 1 5 1
 5 9 9 1 5 3 6 1 8 9 8 7 6 7 6 5 6 0 8 8 9 8 6 1 0 4 1 6 3 8 6 7 4 5 6 3 0
 3 3 3 0 7 7 5 7 8 0 7 8 9 6 4 5 0 1 4 6 4 3 3 0 9 5 9 2 1 4 2 1 6 8 9 2 4
 9 3 7 6 2 3 3 1 6 9 3 6 3 2 2 0 7 6 1 1 9 7 2 7 8 5 5 7 5 2 3 7 2 7 5 5 7
 0 9 1 6 5 9 7 4 3 8 0 3 6 4 6 3 2 6 8 8 8 4 6 7 5 2 4 5 3 2 4 6 9 4 5 4 3
 4 6 2 9 0 1 7 2 0 9 6 0 4 2 0 7 9 8 5 4 8 2 8 4 3 7 2 6 9 1 5 1 0 8 2 1 9
 5 6 8 2 7 2 1 5 1 6 4 5 0 9 4 1 1 7 0 8 9 0 5 4 3 8 8 6 5 3 4 4 4 8 8 7 0
 9 6 3 5 2 3 0 8 3 3 1 3 3 0 0 4 6 0 7 7 6 2 0 4 4 2 3 7 8 9 8 6 8 5 6 2 2
 3 1 7 7 8 0 3 3 2 1 5 5 9 1 3 7 0 0 7 0 4 5 9 3 3 4 3 1 8 9 8 3 6 2 1 6 2
 1 7 5 5 1 9]
```

预测结果：

```
[2 8 2 6 6 7 1 9 8 5 2 8 6 6 6 6 1 0 5 8 8 7 8 4 7 5 4 9 2 9 4 7 6 8 9 4 3
 1 0 1 8 6 7 7 1 0 7 6 2 1 9 6 7 9 0 0 5 1 6 3 0 2 3 4 1 9 3 6 9 1 8 3 5 1
 2 8 2 2 9 7 2 3 6 0 5 3 7 5 1 2 9 9 3 1 7 7 4 8 5 8 5 5 2 5 9 0 7 1 4 7 3
 4 8 9 7 9 8 2 6 5 2 5 3 4 1 7 0 6 1 5 9 9 9 5 9 9 5 7 5 6 2 8 6 7 6 1 5 1
 5 9 9 1 5 3 6 1 8 9 8 7 6 7 6 5 6 0 8 8 9 3 6 1 0 4 1 6 3 8 6 7 4 9 6 3 0
 3 3 3 0 4 7 5 7 8 0 7 8 9 6 4 5 0 1 4 6 4 3 3 0 9 5 9 2 1 4 2 1 6 8 9 2 4
 9 3 7 6 2 3 3 1 6 9 8 6 3 2 2 0 7 6 1 1 9 7 2 7 8 5 5 7 5 2 3 7 2 7 5 5 7
 0 9 1 6 5 9 7 4 3 8 0 3 6 4 6 3 1 6 8 8 8 4 6 7 5 2 4 5 3 2 4 6 9 4 5 4 3
 4 6 2 9 0 1 7 2 0 9 6 0 4 2 0 7 5 8 5 7 8 2 8 4 3 7 2 6 9 1 5 1 0 8 2 1 9
 5 6 8 2 7 2 1 5 1 6 4 5 0 9 4 1 1 7 0 8 9 0 5 4 3 8 8 6 5 3 4 4 4 8 8 7 0
 9 6 3 5 2 3 0 8 3 3 1 3 3 0 0 4 6 0 7 7 6 2 0 4 4 2 3 7 8 9 8 6 8 5 6 2 2
 3 1 7 7 8 0 3 3 2 1 5 5 9 1 3 7 0 0 7 0 7 5 9 3 3 4 3 1 8 9 8 3 6 2 1 6 2
 1 7 5 5 1 9]
```

正确与否

```
[ True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True False  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True False  True False  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True True False
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True False  True  True  True  True  True  True  True  True  True  True
  True False  True  True  True  True  True  True  True False  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True False  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True False
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
```

训练集的准确率：0.9866369710467706， 而测试集的准确率为：0.9733333333333334

四、探索：针对 KNN 算法模型超参数的调优

4.1 目的及操作方法：

** 分析不同的超参数的变化对K 近邻算法预测精度和泛化能力的影响*

** 使用控制变量法，固定其他变量 只改变一个变量 就是一张折线图*

4.2 调整 KNN 算法模型的超参数及对超参数概念的分析

- ```
  train_accuracy1, train_accuracy2, train_accuracy3,
  train_accuracy4 = [], [], [], []
  test_accuracy1, test_accuracy2, test_accuracy3, test_accuracy4 =
  [], [], [], []
  ```

4.2.1 n_neighbors 超参数

- ** n_neighbors：即 KNN 中的 K 值，代表的是邻居的数量。K 值如果比较小，会造成过拟合。如果 K 值比较大，无法将未知物体分类出来。一般我们使用默认值 5。*

- ```
  neighbours_range = range(1, 11)
  ```
- ```
  # n_neighbors 超参数对KNeighbors 预测效果的影响
  for n in neighbours_range:
      # 构建模型
      estimator = KNeighborsClassifier(n_neighbors=n)
      estimator.fit(x_train, y_train)
      # 记录训练集精度S
      train_accuracy1.append(estimator.score(x_train, y_train))
      # 记录泛化能力
      test_accuracy1.append(estimator.score(x_test, y_test))
  ```

4.2.2 weights 超参数

- ** weights：是用来确定邻居的权重，有三种方式：*
- *weights=uniform，代表所有邻居的权重相同；*
- *weights=distance，代表权重是距离的倒数，即与距离成反比；*
- *自定义函数，你可以自定义不同距离所对应的权重。大部分情况下不需要自己定义函数。*

- ```
  weights_range = ['uniform', 'distance']
  ```
- *# 超参数带入代码同n_neighbors*

4.2.3 algorithm 超参数

- *# algorithm：用来规定计算邻居的方法，它有四种方式：*
- *# algorithm=auto，根据数据的情况自动选择适合的算法，默认情况选择 auto*
- *# algorithm=kd_tree，也叫作 KD 树，是多维空间的数据结构，方便对关键数据进行检索，不过 KD 树适用于维度少的情况，一般维数不超过 20，如果维数大于 20 之后，效率反而会下降；*

- *# algorithm=ball_tree，也叫作球树，它和 KD 树一样都是多维空间的数据结果，不同于 KD 树，球树更适用于维度大的情况；*
- *# algorithm=brute，也叫作暴力搜索，它和 KD 树不同的地方是在于采用的是线性扫描，而不是通过构造树结构进行快速检索。当训练集大的时候，效率很低。*

- `algorithm_range = ['auto', 'kd_tree', 'ball_tree', 'brute']`
- *# 超参数带入代码同n_neighbors*

4.2.4 leaf_size 超参数

- *# leaf_size：代表构造 KD 树或球树时的叶子数，默认是 30，调整 leaf_size 会影响到树的构造和搜索速度。创建完 KNN 分类器之后，我们就可以输入训练集对它进行训练，*
- *# 这里我们使用 fit() 函数，传入训练集中的样本特征矩阵和分类标识，会自动得到训练好的 KNN 分类器。*
- *# 然后可以使用 predict() 函数来对结果进行预测，这里传入测试集的特征矩阵，可以得到测试集的预测分类结果。*

- `leaf_size_range = [10, 20, 30, 40, 50]`
- *# 超参数带入代码同n_neighbors*

4.3 使用 matplotlib 绘图查看效果

4.3.1 绘图功能代码的实现

- `import matplotlib.pyplot as plt`

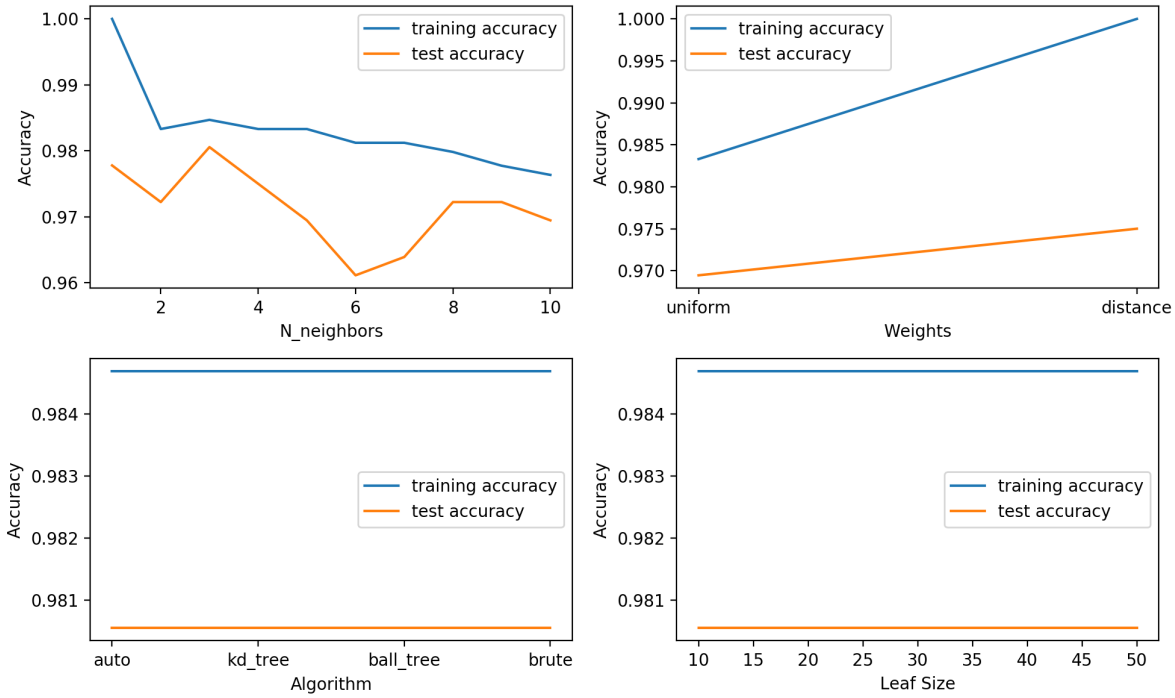
- *# 图大小*
  ```
  plt.figure(figsize=(10, 6))
  ```

- *# 221 > 2 行2 列第1 个*
  ```
  figure1 = plt.subplot(221)
  plt.plot(neighbours_range, train_accuracy1, label='training accuracy')
  plt.plot(neighbours_range, test_accuracy1, label='test accuracy')
  plt.xlabel('N_neighbors')
  plt.ylabel('Accuracy')
  plt.legend()
  ```

- *# 其他三个图 222，223，224 参照 221 的代码部分*

- *# 展示 画图的结果*
  ```
  plt.tight_layout()
  plt.show()
  ```

4.3.2 折线图可视化以及对运行结果的分析



- 对模型超参数变化对模型预测准确率折线图的说明解释及分析：
  - n_neighbors 取 3，weights 取 distance 时，KNN 算法模型的预测准确率较高。
  - 经过断点调试，后两个参数 预测的准确率结果是完全一样的数字。我觉得可能原因是 ：algorithm 超参数是用于确定 “邻居” 的搜索方式；leaf_size 影响的是存储和查询树的速度。因此这两个超参与预测结果无关，只与运行速度有关。

五、实验结果与结论

本次实验通过使用 sklearn 库中的 KNN 算法对 实验数据 load_digits() 这一手写数字图形的识别，实现了较高的预测准确率。使用控制变量法，通过对 KNN 算法的超参数进行调整，找出来最佳的超参数，以及不同超参数对模型准确率的影响，也发现了 algorithm 超参数和 leaf_size 超参数对模型的实际意义。对 KNN 的算法模型原理的理解更深入了。

六、参考资源

- 【黑马程序员】python 机器学习-视频教程：
  https://www.bilibili.com/video/av39137333?p=21
- 查看 neighbors 大小对 K 近邻分类算法预测准确度和泛化能力的影响：
  https://www.cnblogs.com/yszd/p/9298214.html
- 用 Python 中的 matplotlib 画出一个 3 行 2 列的饼图：
  https://blog.csdn.net/qq_33221533/article/details/81568244