

实验三 分类 KNN 算法

【实验目的】

1. 了解 Anaconda 和 python 的使用与设置。
2. 掌握分类 KNN 算法的基本原理和实现方法。

【实验学时】

建议 2 学时

【实验环境配置】

- 1、Windows 环境
- 2、Anaconda
- 3、Pandas

【实验原理】

数据挖掘中分类 KNN 算法的实现：

- 1、Pandas 数据导入
- 2、KNN 算法
- 3、sk-learn 库的 KNN 模块使用

【实验步骤】

1. 打开 Anaconda 的 jupyter notebook，创建实验三，明确标题和步骤。

1.导入数据

```
4]: import os

5]: import pandas as pd

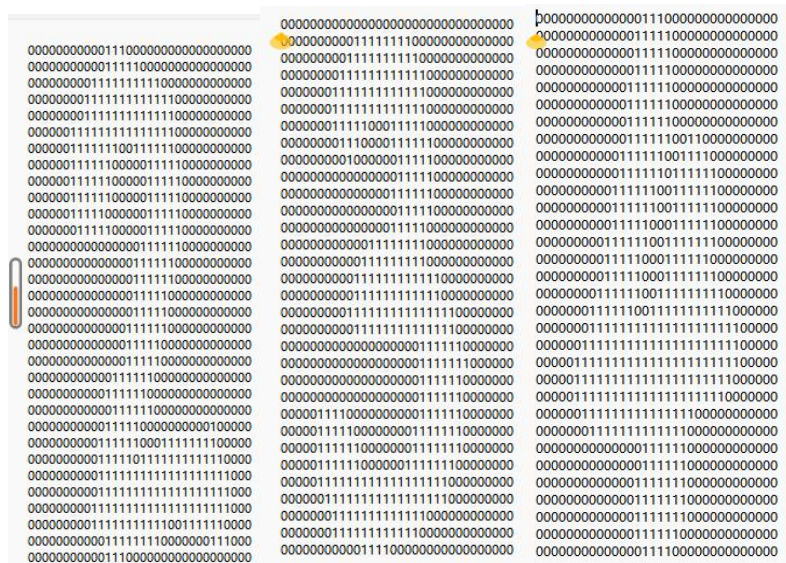
6]: dir_root = os.path.join(os.path.curdir, 'digits')

7]: dir_training = os.path.join(dir_root, 'trainingDigits') # 训练集

8]: dir_test = os.path.join(dir_root, 'testDigits') # 测试集


: def df_exact(dir_name):
    for root, dirs, files in os.walk(dir_name):
        df_digit = pd.DataFrame(columns=['X', 'Y'])
        for f in files:
            list_f = f.split('_')
            y = int(list_f[0])
            df_data = pd.read_csv(os.path.join(dir_name, f), header=None)
            list_x = df_data.iloc[:, 0]
            s = ''
            x = []
            for d in list_x:
                s = s + d
            for i in s:
                x.append(int(i))
            new = pd.DataFrame({'X': [x], 'Y': [y]})
            df_digit = df_digit.append(new)
        print(df_digit)
    return df_digit
```

提取训练集和测试集的函数，根据文件名分割出其代表的数字 y，文件里面的内容为[x]，根据测试集中 X 的内容判断其文件代表的是 0~9 中的什么数字。那么形成的 DataFrame 集合中第一列代表的是数据形态内容，第二列代表的是数据的标签。



观察数据文件可以发现，文本文件中的内容是由 0 和 1 组成的提取出来的图形矩阵，代表不同的数字。通过对训练集来对测试集进行分类。

2. 导入 pandas 库和 os 库，从 digits 数据文件夹中，导入手写数字实验训练数据和测试数据。

（注：可能涉及的函数 `os.path.join`, `os.walk`, `pandas.read_csv`）

```
ds_training = df_exact(dir_training)
```

		X	Y
0	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, ...	0	
0	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, ...	0	
0	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, ...	0	
0	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, ...	0	
0	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, ...	0	
..	
0	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	9	
0	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	9	
0	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	9	
0	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, ...	9	
0	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, ...	9	

[1934 rows x 2 columns]

```

: ds_test = df_exact(dir_test)

                                X Y
0  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, ... 0
0  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ... 0
0  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, ... 0
0  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, ... 0
0  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ... 0
..
0  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ... 9
0  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, ... 9
0  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, ... 9
0  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, ... 9
0  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ... 9

[946 rows x 2 columns]

```

```

: ds_training = ds_training.reset_index()
  ds_test = ds_test.reset_index()

```

调用 `df_exact` 函数，分别从训练集文件夹 `dir_training` 和测试集 `dir_test` 中提取出测试数据。

- 对导入的实验数据，根据《机器学习实战》教程第二章，完成 KNN 算法，并完成对于手写数字数据的分类识别。

2、采用sklearn完成knn算法

```

: import numpy as np
  from sklearn.neighbors import KNeighborsClassifier

: array_x = np.array([i for i in ds_training.X])
  array_y = np.array([i for i in ds_training.Y])
  arr_testx = np.array([i for i in ds_test.X])
  arr_testy = np.array([i for i in ds_test.Y])

: neigh = KNeighborsClassifier(n_neighbors=15)

: neigh.fit(array_x, array_y)

: KNeighborsClassifier(n_neighbors=15)

: y_p = neigh.predict(arr_testx)

```


4. 对导入的实验数据，根据 sk-learn 的 KNN 算法模块 `sklearn.neighbors.KNeighborsClassifier`，完成对于手写数字数据的分类识别。

3、设计KNN算法函数

```
def distEclud(vecA, vecB):
    return np.linalg.norm(vecA - vecB)

def knn(ds_training, ds_test, k = 10):
    Y_p = []
    for p in ds_test.X:
        ds = ds_training.copy(deep=True)
        D = []
        for q in ds_training.X:
            vecA = np.array(p)
            vecB = np.array(q)
            d = distEclud(vecA, vecB)
            D.append(d)
        ds['D'] = np.array(D)
        ds = ds.sort_values(by=['D'])
        ds = ds.reset_index()
        Y = ds.loc[0:k-1, ['Y']]
        # 打印Y
        rank_y = Y['Y'].value_counts()
        # 打印rank_y
        print(rank_y.index[0], end=',')
        y_p = rank_y.index[0]
        Y_p.append(y_p)
    return Y_p
```

`distEclud` 函数是用来计算两点的欧式距离，算出每两个点之间的距离，然后进行排序，找出最大的前 `K` 个点，依次类推。输入训练集和测试集，算出测试集每一个点到训练集所有点的距离，算出前 15 个点有哪些。找出这些点中标号最多的数字，就代表其属于此种类型，算出的结果和 `sklearn` 差不多。

```
Y_p = knn(ds_training, ds_test, k = 15)
```

[illegible]

【实验总结】

通过实验发现，knn 算法对内存要求较高，因为该算法存储了所有训练数据，预测阶段可能很慢。当需要使用分类算法，且数据比较大的时候就可以尝试使用 KNN 算法进行分类了。