1. KNN (K nearest neighbor algorithm) recognizes handwritten numbers

1. KNN (K nearest neighbor algorithm) recognizes handwritten numbers
1.1. Introduction to KNN
1.2.Recognize handwritten digits and determine the K value
1.3. Recognize your own handwritten numbers

1.1. Introduction to KNN

1) KNN (K-Nearest Neighbor) is a supervised learning method. Its working mechanism is very simple and does not require training sets. It is one of the simpler classic machine learning algorithms. It can handle regression and classification problems.

2) Method ideas

In the feature space, if most of the k nearest samples (i.e. the nearest ones in the feature space) near a sample belong to a certain category, then the sample also belongs to this category.

In official terms, the so-called K nearest neighbor algorithm is to find the K nearest instances (i.e. the K neighbors mentioned above) to a new input instance in the training data set given a training data set. If most of these K instances belong to a certain class, the input instance is classified into this class.

3) Working principle

There is a sample data set, also called a training sample set, and each data in the sample set has a label, that is, we know the relationship between each data in the sample set and its corresponding classification. After inputting unlabeled data, each feature in the new data is compared with the corresponding feature of the data in the sample set, and the classification label of the data with the most similar features (nearest neighbor) in the sample set is extracted. Generally speaking, we only select the first K most similar data in the sample data set, which is the origin of K in the K nearest neighbor algorithm. Usually K is an integer not greater than 20. Finally, the classification with the most occurrences in the K most similar data is selected as the classification of the new data.

4) KNN advantages and disadvantages

- Advantages

Flexible usage, convenient for small sample prediction, high accuracy, insensitive to outliers, no data input assumptions

- Disadvantages

Lack of training phase, unable to cope with multiple samples, high computational complexity, high spatial complexity

5) KNN implementation steps:

- Calculate distance

Euclidean distance, that is

$$L_2(x_i, x_j) = (\sum_{l=1}^{n} |x_i{}^{(l)} - x_j{}^{(l)}|^2)^{\frac{1}{2}}$$

- Sort by increasing distance
- Select K points with the smallest distance (generally no more than 20)
- Determine the frequency of occurrence of the category of the first K points, frequency of occurrence = a certain category/k
- Return the category with the highest frequency of occurrence among the first K points as the predicted classification of the test data

1.2.Recognize handwritten digits and determine the K value

Source code path:

/home/pi/project_demo/06.Open_source_cv_fundamentals_course/E.Machine_Learning/01_KNN_Digit_Recog/01_K_in_KNN_Digits.ipynb

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import numpy
import operator
import os
import matplotlib.pyplot as plt
```

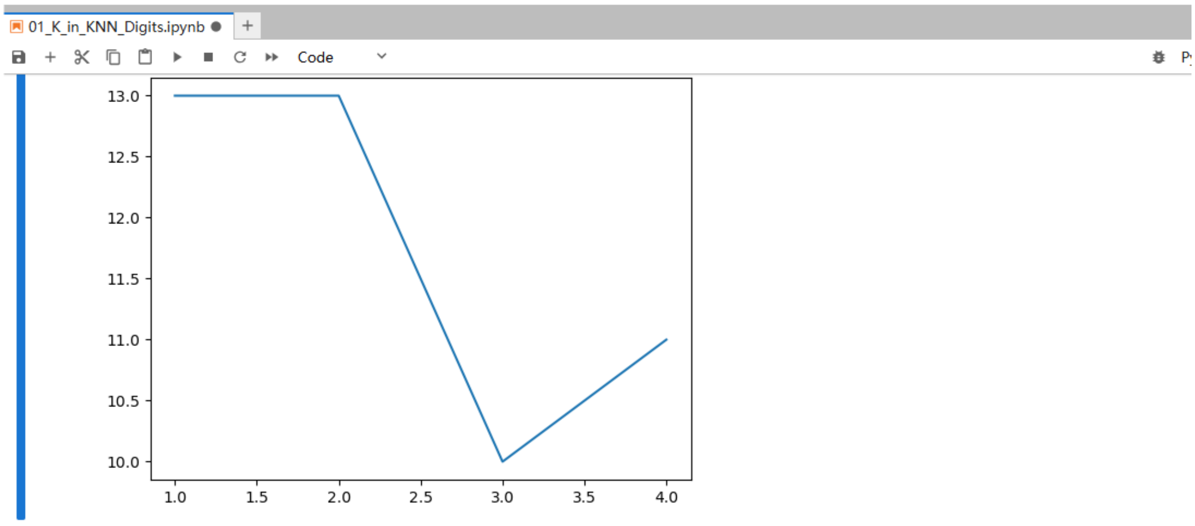
```
# 将图像数据转换为（1，1024）向量 Convert the image data to a (1, 1024) vector
def img2vector(filename):
    returnVect = numpy.zeros((1, 1024))
    file = open(filename)
    for i in range(32):
        lineStr = file.readline()
        for j in range(32):
            returnVect[0, 32 * i + j] = int(lineStr[j])
    return returnVect
```

```
# kNN分类器 kNN Classifier
def classifier(inX, dataSet, labels, k):
    #numpy中shape[0]返回数组的行数，shape[1]返回列数 In numpy, shape[0] returns the number of rows in the array, and shape[1] returns the number of columns
    #MDS降维操作 MDS Dimensionality Reduction Operation
    dataSetSize = dataSet.shape[0]
    #去逆矩阵 De-invert matrix
    diffMat = numpy.tile(inX, (dataSetSize, 1)) - dataSet
    #二维特征相减后乘方 Subtract two-dimensional features and then square them
    sqDiffMat = diffMat ** 2
    #计算距离 Calculating distance
    sqDistances = sqDiffMat.sum(axis=1)
    distances = sqDistances ** 0.5
    print ("distances:",distances)
    #返回distance中元素从小到大排序后的索引 Returns the index of the elements in
distance sorted from small to large
    sortedDistIndicies = distances.argsort()
    print ("sortDistance:",sortedDistIndicies)
    classCount = {}
    for i in range(k):
        #取出前k个元素的类别 Take out the categories of the first k elements
        voteIlabel = labels[sortedDistIndicies[i]]
        classCount[voteIlabel] = classCount.get(voteIlabel, 0) + 1
    #reverse降序排序字典 reverse sort dictionary in descending order
    sortedClassCount = sorted(classCount.items(), key=operator.itemgetter(1),
reverse=True)
    return sortedClassCount[0][0]
```

```
# 测试手写数字识别代码 Test the handwritten digit recognition code
def handwritingClassTest(k):
    #训练部分 Training
    hwLabels = []
    trainingFileList = os.listdir('knn-digits/trainingDigits')
    m = len(trainingFileList)
    trainingMat = numpy.zeros((m, 1024))
    for i in range(m):
        fileNameStr = trainingFileList[i]
        fileStr = fileNameStr.split('.')[0]
        try:
            classNumStr = int(fileStr.split('_')[0])
        except Exception as e:
            print('Error:', e)

        hwLabels.append(classNumStr)
        trainingMat[i, :] = img2vector("knn-digits/trainingDigits/%s" %
fileNameStr)

    #测试数据分类结果 Test data classification results
    testFileList = os.listdir('knn-digits/testDigits')
    errorCount = 0.0
    mTest = len(testFileList)
    for i in range(mTest):
        fileNameStr = testFileList[i]
        fileStr = fileNameStr.split('.')[0]
        try:
            classNumStr = int(fileStr.split('_')[0])
        except:
            print(fileStr.split('_')[0])
            continue
        vectorTest = img2vector("knn-digits/testDigits/%s" % fileNameStr)
        result = classifier(vectorTest, trainingMat, hwLabels, k)
        print("The classification result is: %d, the true result is: %d" %
(result, classNumStr))
        if result != classNumStr:
            errorCount += 1.0
    print("Total Errors:%d" % errorCount)
    print("Error rate:%f" % (errorCount / mTest))
    return errorCount
```

```
# 这里是为了测试取不同的k值，识别的效果如何 Here we want to test the recognition effect
with different k values.
def selectK():
    x = list()
    y = list()
    for i in range(1, 5):
        x.append(int(i))
        y.append(int(handWritingClassTest(i)))
    plt.plot(x, y)

    plt.show()
```

```
# 开始测试，会生成折线图 Start the test and a line chart will be generated
selectK()
```

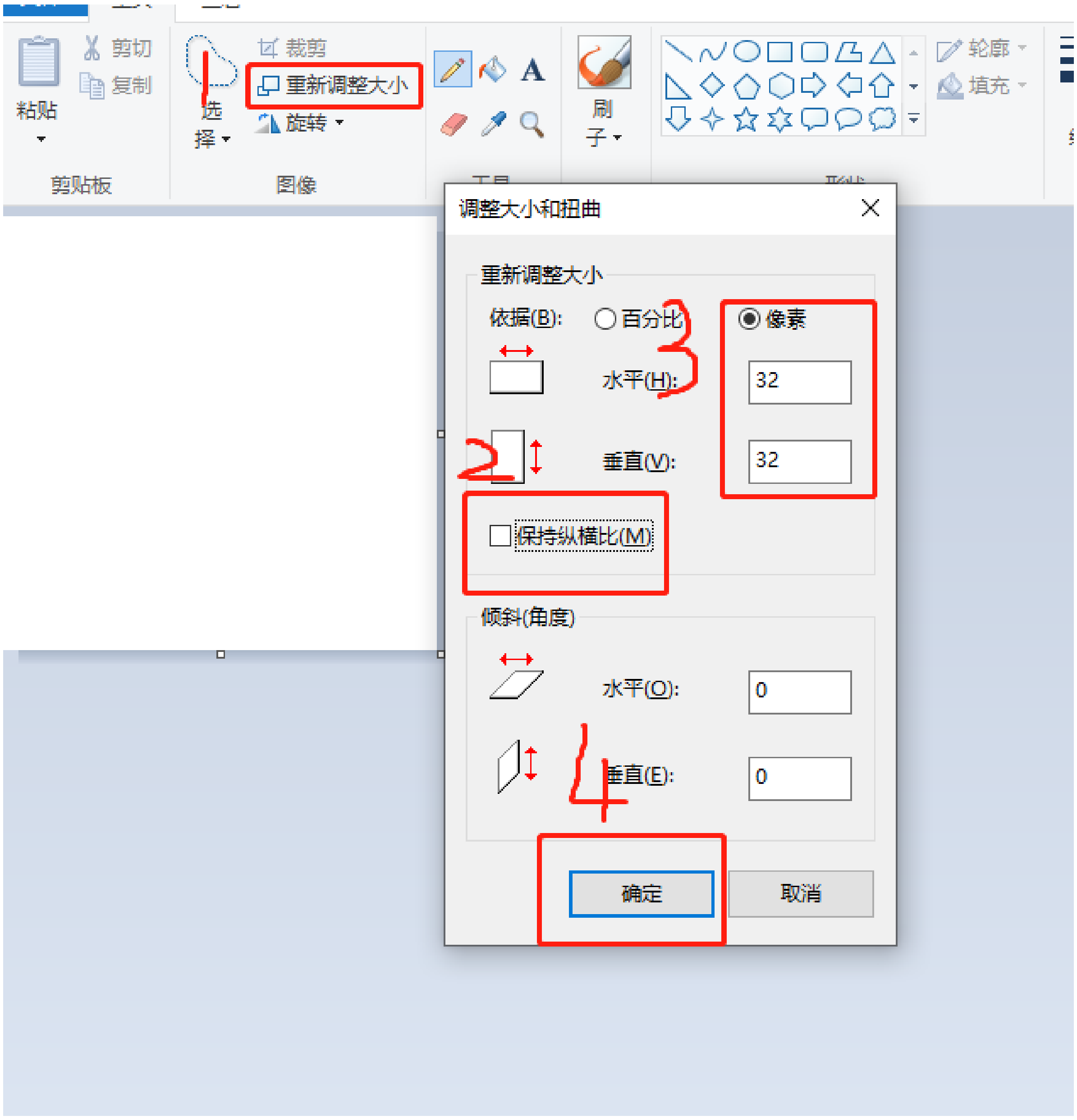


```
handWritingClassTest(3)
```

1.3. Recognize your own handwritten numbers

Steps to create handwritten number images:

- Open the drawing software and adjust the resolution to 32*32. You can also use other resolutions, but it is best to draw the image to fill the entire area, otherwise the error rate will be very high.



调整大小和扭曲

重新调整大小

依据(B): ○ 百分比 [✓ ● 像素]

水平(H): 32

垂直(V): 32

☐ 保持纵横比(M)

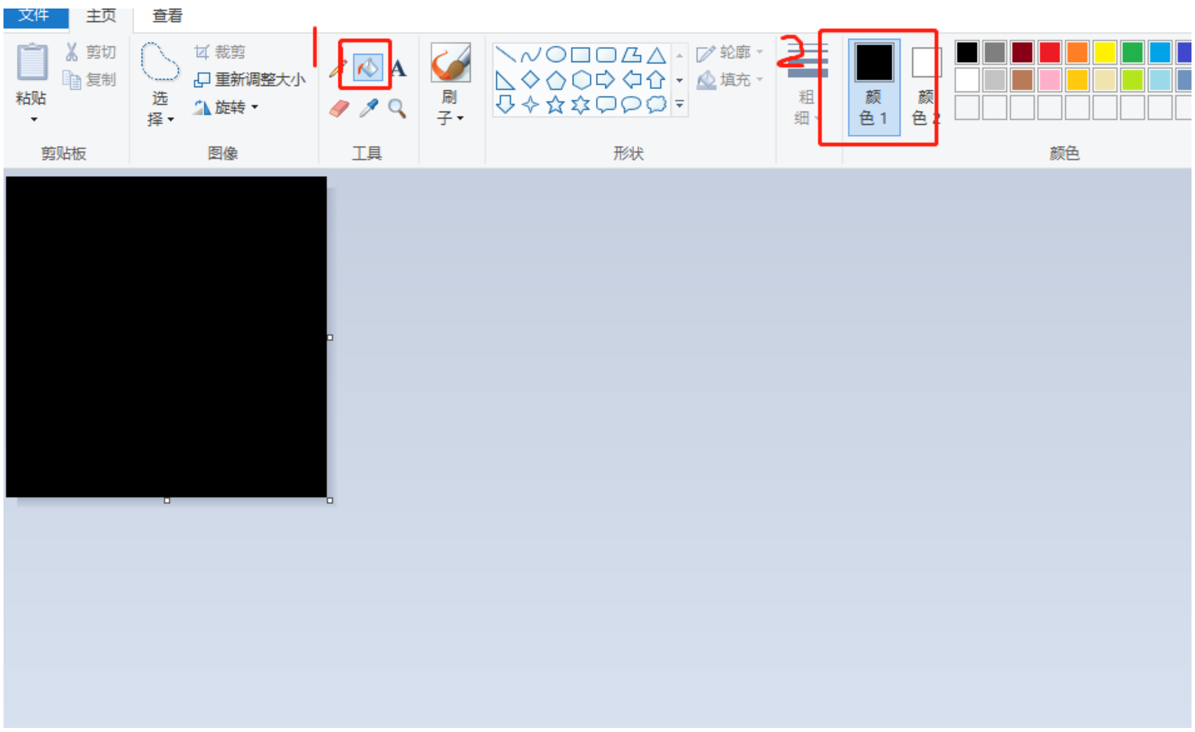
倾斜(角度)

水平(O): 0

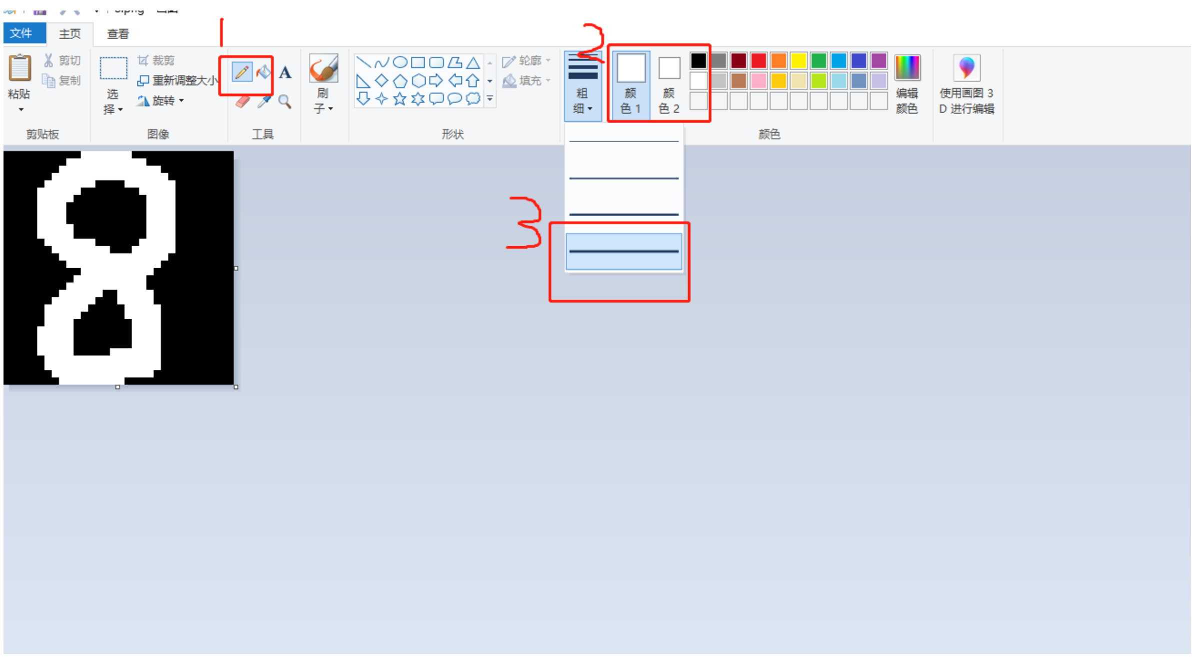
垂直(E): 0

确定

- Hold down ctrl and scroll up to zoom in the image. Select the paint bucket tool and fill the entire image with black.



- Select the pencil tool, choose white for color, and choose the maximum line width. As shown below:



After drawing, save it as a png image (8.png is used as an example in this example), and copy it to the project directory using the WinSCP tool

Source code path:

/home/pi/project_demo/06.Open_source_cv_fundamentals_course/E.Machine_Learning/01_KNN_Digit_Recog/02_KNN_Personal_Handwriting.ipynb

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import numpy as np
import operator
import os
```

```python
# 将图像数据转换为（1，1024）向量 Convert the image data to a (1, 1024) vector
def img2vector(filename):
    returnVect = np.zeros((1, 1024))
    file = open(filename)
    for i in range(32):
        lineStr = file.readline()
        for j in range(32):
            returnVect[0, 32 * i + j] = int(lineStr[j])
    return returnVect
```

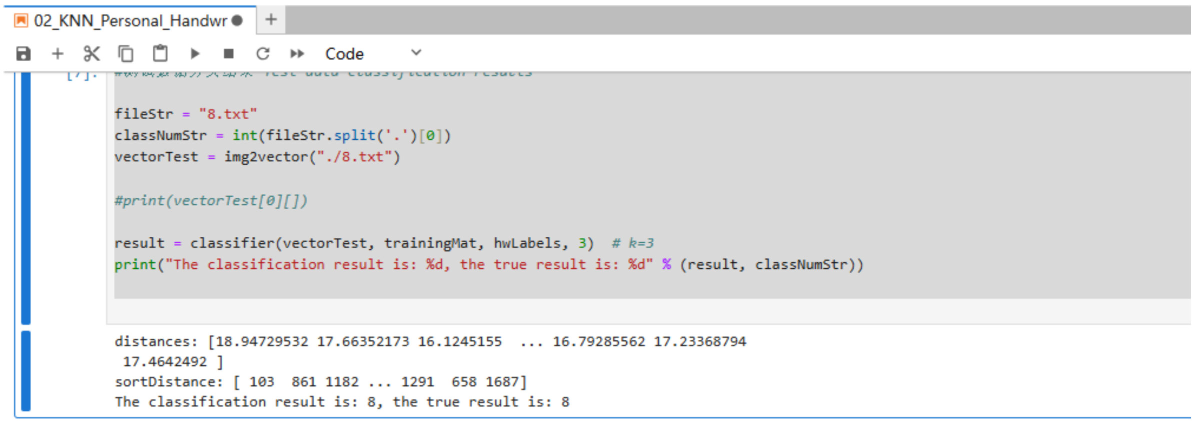
```python
# KNN分类器 KNN Classifier
def classifier(inX, dataSet, labels, k):

    #numpy中shape[0]返回数组的行数，shape[1]返回列数 In numpy, shape[0] returns the number of rows in the array, and shape[1] returns the number of columns
    #MDS降维操作 MDS Dimensionality Reduction Operation
    dataSetSize = dataSet.shape[0]
    #去逆矩阵 De-invert matrix
    diffMat = np.tile(inX, (dataSetSize, 1)) - dataSet
    #二维特征相减后乘方 Subtract two-dimensional features and then square them
    sqDiffMat = diffMat ** 2
    #计算距离 Calculating distance
    sqDistances = sqDiffMat.sum(axis=1)
    distances = sqDistances ** 0.5
    print ("distances:",distances)
    #返回distance中元素从小到大排序后的索引 Returns the index of the elements in 
distance sorted from small to large
    sortedDistIndicies = distances.argsort()
    print ("sortDistance:",sortedDistIndicies)
    classCount = {}
    for i in range(k):
        #取出前k个元素的类别 Take out the categories of the first k elements
        voteIlabel = labels[sortedDistIndicies[i]]
        classCount[voteIlabel] = classCount.get(voteIlabel, 0) + 1
    #reverse降序排序字典 reverse sort dictionary in descending order
    sortedClassCount = sorted(classCount.items(), key=operator.itemgetter(1), 
reverse=True)
    return sortedClassCount[0][0]
```

```
#图像数据转换为txt文件 Convert image data to txt file

from PIL import Image
"""
    将图像数据转换为txt文件 Convert image data to txt file
    :param img_path: 图像文件路径 param img_path: image file path
    :type txt_name: 输出txt文件路径 type txt_name: output txt file path
"""
def img2txt(img_path, txt_name):

    im = Image.open(img_path).convert('1').resize((32, 32))  # type:Image.Image
    data = np.asarray(im)
    np.savetxt(txt_name, data, fmt='%d', delimiter='')
```

```
#转换图片转化成32*32数组 Convert the image into a 32*32 array

img2txt("8.png", "8.txt")
```

```
#训练部分 Training

hwLabels = []
trainingFileList = os.listdir('knn-digits/trainingDigits')
m = len(trainingFileList)
trainingMat = np.zeros((m, 1024))
for i in range(m):
    fileNameStr = trainingFileList[i]
    fileStr = fileNameStr.split('.')[0]
    try:
#         if(fileStr.split('_')[0] == ''):
#             continue
#         else:
        classNumStr = int(fileStr.split('_')[0])
    except Exception as e:
        print('Error:', e)
    hwLabels.append(classNumStr)
    trainingMat[i, :] = img2vector("knn-digits/trainingDigits/%s" % fileNameStr)
```

```
#测试数据分类结果 Test data classification results

fileStr = "8.txt"
classNumStr = int(fileStr.split('.')[0])
vectorTest = img2vector("./8.txt")

#print(vectorTest[0][])

result = classifier(vectorTest, trainingMat, hwLabels, 3)  # k=3
print("The classification result is: %d, the true result is: %d" % (result,
classNumStr))
```



```
fileStr = "8.txt"
classNumStr = int(fileStr.split('.')[0])
vectorTest = img2vector("./8.txt")

#print(vectorTest[0][])

result = classifier(vectorTest, trainingMat, hwLabels, 3)  # k=3
print("The classification result is: %d, the true result is: %d" % (result, classNumStr))
```

```
distances: [18.94729532 17.66352173 16.1245155  ... 16.79285562 17.23368794
 17.4642492 ]
sortDistance: [ 103  861 1182 ... 1291  658 1687]
The classification result is: 8, the true result is: 8
```