

1.5 Machine learning

Definition:

Machine learning is a multidisciplinary interdisciplinary subject, covering the knowledge of probability theory, statistics, approximate theory and complex algorithms.

Machine learning has the following definitions:

- (1) Machine learning is a science of artificial intelligence. The main research object in this field is artificial intelligence.
- (2) Machine learning is the study of computer algorithms that can be improved automatically through experience.
- (3) Machine learning uses data or past experience to optimize the performance standards of computer programs.

1.5.1 KNN (K Nearest Neighbor Algorithm) recognizes handwritten digits

The idea of KNN is, In the feature space, if most of the k nearest feature samples in a sample belong to a certain category, the sample also belongs to this category.

The working principle of KNN, there is a sample data set (training sample set), and each data in the sample set has a label, that is, we know the corresponding relationship between each data in the sample set and the category to which it belongs. After inputting unlabeled data, compare each feature in the new data with the feature corresponding to the data in the sample set, and extract the classification label of the data with the most similar features in the sample set.

In generally, we only select the top K most similar data in the sample data set. This is the source of K in the K nearest neighbor algorithm, K is an integer smaller than 20. Finally, the classification with the most occurrences of the K most similar data is selected as the classification of the new data.

KNN advantages:

- 1) Flexible usage
- 2) Convenient for small sample prediction
- 3) High accuracy

KNN disadvantages:

- 1) Lack of training stage, unable to cope with multiple samples
- 2) Calculation more complex and space more complex

KNN implementation steps:

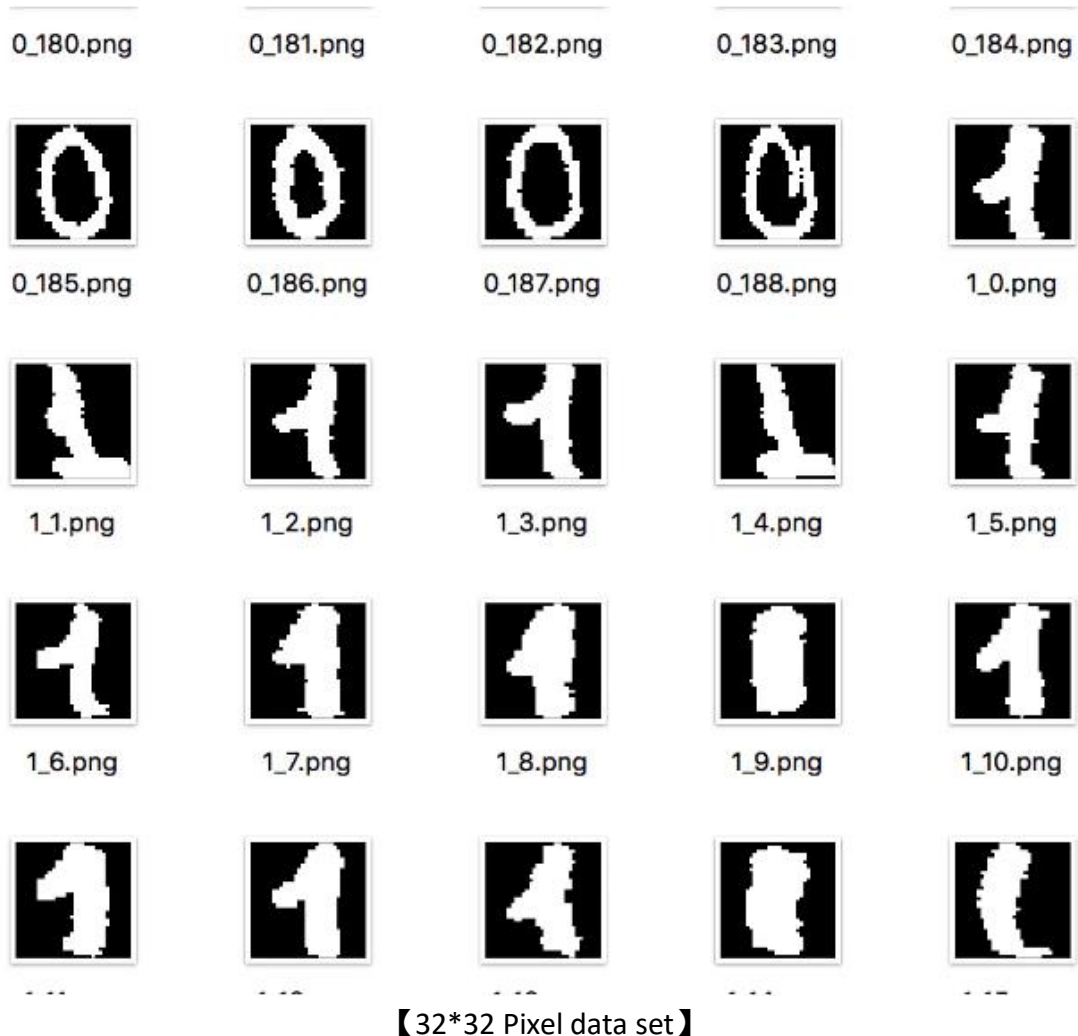
- 1) Calculate distance

The Euclidean distance formula is shown below,

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

- 2) Sort according to the increasing relationship of distance;
 - 3) Select K points with the smallest distance (less than 20 points);
 - 4) Determine the frequency of occurrence of the category of the K points;
- Frequency of occurrence = category/k**
- 5) Return the category with the highest frequency among the K points as the predicted classification of the test data.

Next, we take the recognition of handwritten digits as an example to briefly introduce the implementation of the KNN algorithm.



[illegible]

【Text data】

The data set is divided into training set and testing set, where the training set is already classified data, and the testing set is used to test the algorithm.

1. Recognize handwritten number to determine K value

Code path:

```
/home/pi/Yahboom_Project/Raspbot/1.OpenCV_course/05machine_learning/01KNN
/KNN_1.ipynb
```

Convert data to feature vectors

It can be seen from the above figure that we get a 32*32 matrix, each point is a pixel value, and convert these 1024 (32x32) values into a (1,1024) vector.

```
# Convert image data to (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 Classifier

```
# kNN Classifier

def classifier(inX, dataSet, labels, k):

    # numpy shape [0] return the number of rows in the array, shape [1] returns the
    number of columns

    #MDS

    dataSetSize = dataSet.shape[0]

    # The inverse matrix

    diffMat = numpy.tile(inX, (dataSetSize, 1)) - dataSet

    # subtract a 2-d feature and you multiply it

    sqDiffMat = diffMat ** 2

    # Calculation of distance

    sqDistances = sqDiffMat.sum(axis=1)
```

```
distances = sqDistances ** 0.5

print ("distances:",distances)

#Returns the index of the elements in distance sorted from smallest to largest

sortedDistIndicies = distances.argsort()

print ("sortDistance:",sortDistance)

classCount = {}

for i in range(k):

    #Take the categories of the first k elements

    votelabel = labels[sortedDistIndicies[i]]

    classCount[votelabel] = classCount.get(votelabel, 0) + 1

#reverse

sortedClassCount = sorted(classCount.items(), key=operator.itemgetter(1),
reverse=True)

return sortedClassCount[0][0]
```

Input test set, test algorithm

```
# Test the handwritten number recognition code

def handWritingClassTest(k):

    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)


testFileList = os.listdir('knn-digits/testDigits')

errorCount = 0.0

mTest = len(testFileList)

for i in range(mTest):

    fileNameStr = testFileList[i]

    fileStr = fileNameStr.split('.')[0]

    classNumStr = int(fileStr.split('_')[0])

    vectorTest = img2vector("knn-digits/testDigits/%s" % fileNameStr)

    result = classifier(vectorTest, trainingMat, hwLabels, k)

    print("The classification result is:%d, The real result is:%d" % (result,
classNumStr))

    if result != classNumStr:
```

```

        errorCount += 1.0

    print("Total number of errors:%d" % errorCount)

    print("Error rate:%f" % (errorCount/mTest))

    return errorCount

```

Choose a different k value to view the classification effect

```

# Choose a different k value to view the classification effect

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)

    # Due to the relatively long execution time of the program, here is a voice
    reminder when the program execution is completed (for Mac, Windows users
    remove this line of code)

    os.system("say 'The program is finished'")

    plt.show()

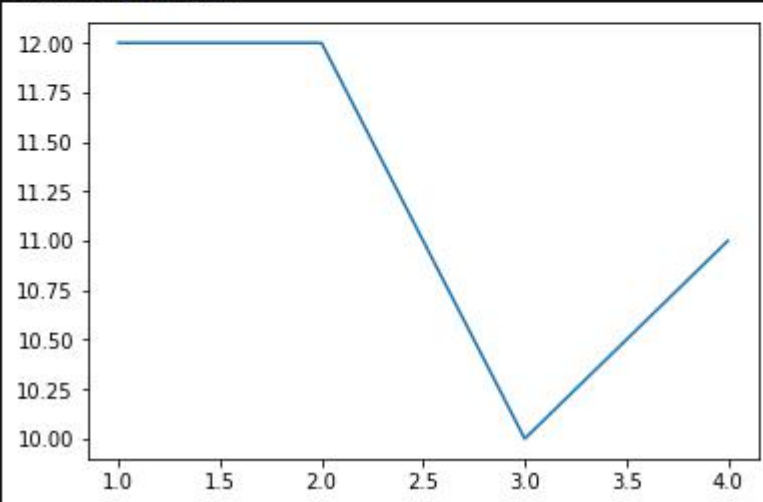
# Start the test, a line chart will be generated

selectK()

```

After running the program, we can see the results shown in the figure below.

```
分类结果是：3, 真实结果是：3  
分类结果是：6, 真实结果是：6  
分类结果是：1, 真实结果是：1  
分类结果是：1, 真实结果是：1  
分类结果是：5, 真实结果是：5  
分类结果是：0, 真实结果是：0  
分类结果是：8, 真实结果是：8  
分类结果是：5, 真实结果是：5  
分类结果是：2, 真实结果是：2  
错误总数：11  
错误率：0.011628
```



The experimental results prove that taking $k = 3$ with a better effect. You can also try other values, but the speed is very slow.

The following is tested with $k = 3$:

```
handWritingClassTest(3)
```

After running the program, we can see the results shown in the figure below.


```

distances: [17.80449381 15.5241747 13.74772708 ... 11.61895004 16.76305461
16.30950643]
sortDistance: [1370 607 653 ... 455 1390 1537]
The classification result is:5, The real result is:5
distances: [15.09966887 19.84943324 19.33907961 ... 18.22086716 17.60681686
19.67231557]
sortDistance: [ 242 199 415 ... 509 1724 1155]
The classification result is:0, The real result is:0
distances: [15.29705854 17.08800749 15.93737745 ... 16.85229955 14.69693846
16.64331698]
sortDistance: [1332 248 419 ... 1761 393 33]
The classification result is:8, The real result is:8
distances: [16.24807681 15.29705854 13.56465997 ... 13.03840481 14.49137675
18.52025918]
sortDistance: [1156 8 1791 ... 989 917 1683]
The classification result is:5, The real result is:5
distances: [16.30950643 15.68438714 15.03329638 ... 16.24807681 12.
17.29161647]
sortDistance: [ 485 539 1007 ... 393 862 853]
The classification result is:2, The real result is:2
Total number of errors:12
Error rate:0.012672

```

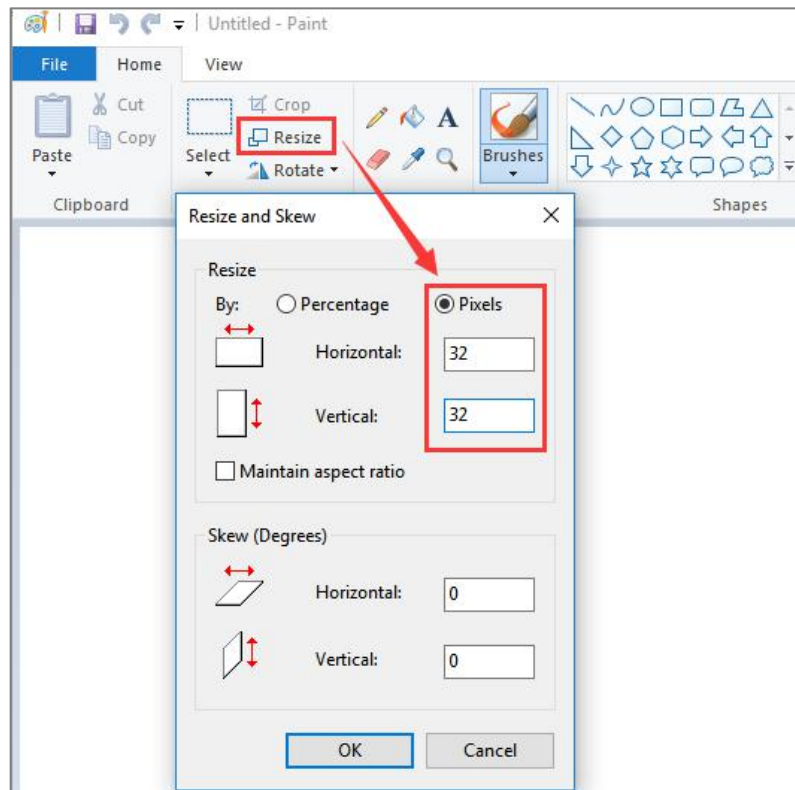
[5]: 12.0

2.Recognizes handwritten number

Steps to make a handwritten digital picture:

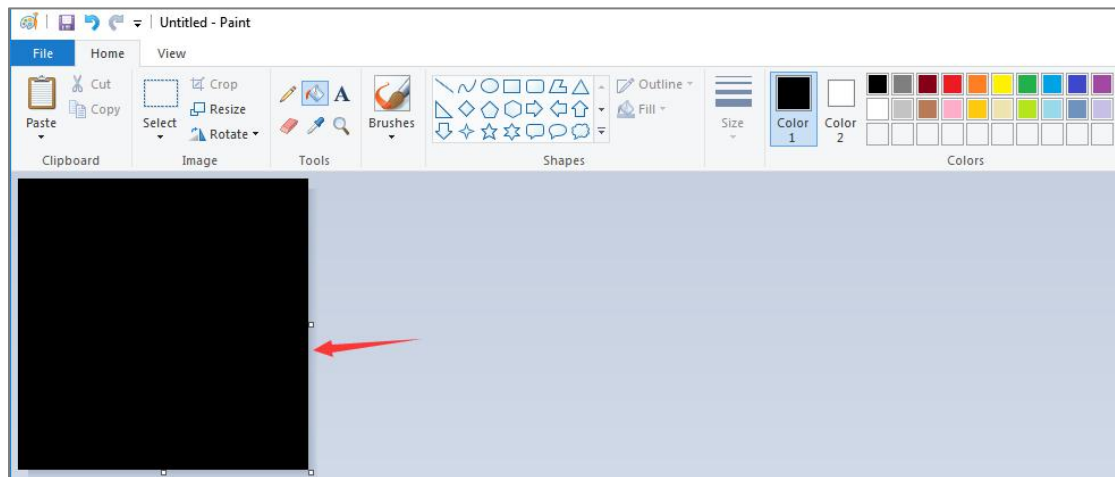
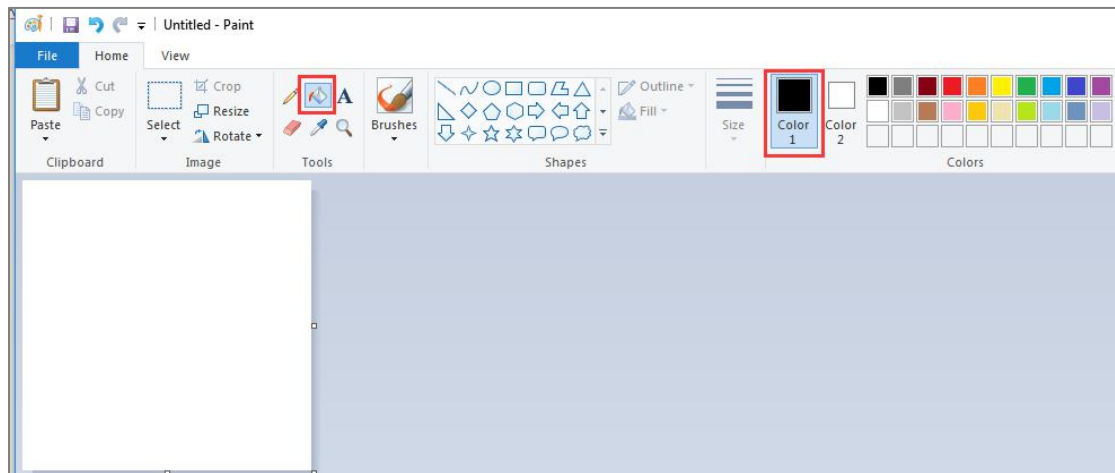
①Open the drawing software, adjust the resolution to 32*32, and other resolutions are also possible, but it is best to the picture can fill the entire interval.

As shown below.

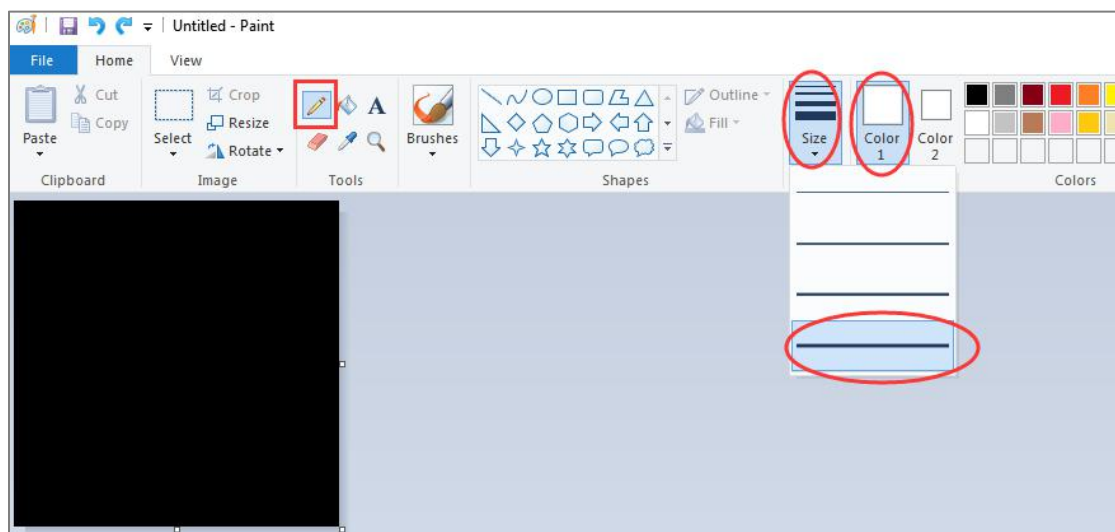


【Adjust the resolution】

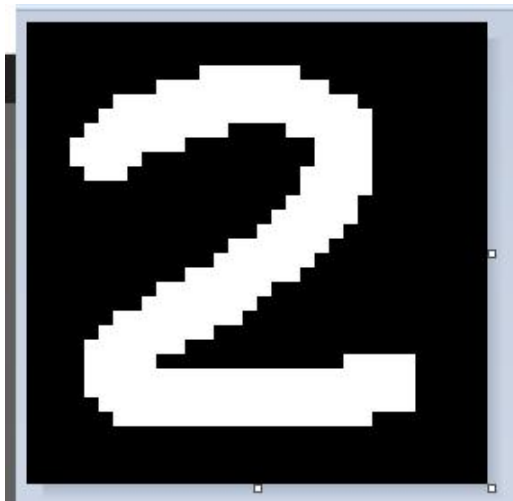
② Press “ctrl” and the mouse wheel slides up to enlarge the picture to the maximum. Select the paint bucket tool and select the black color to fill the entire screen.



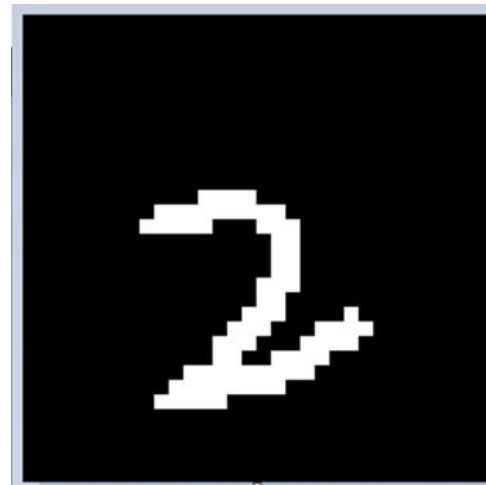
③ Choose the pencil tool, choose white for the color, and choose the maximum width for the line thickness. As shown below.



④ We can write a number, it need fill the entire area.

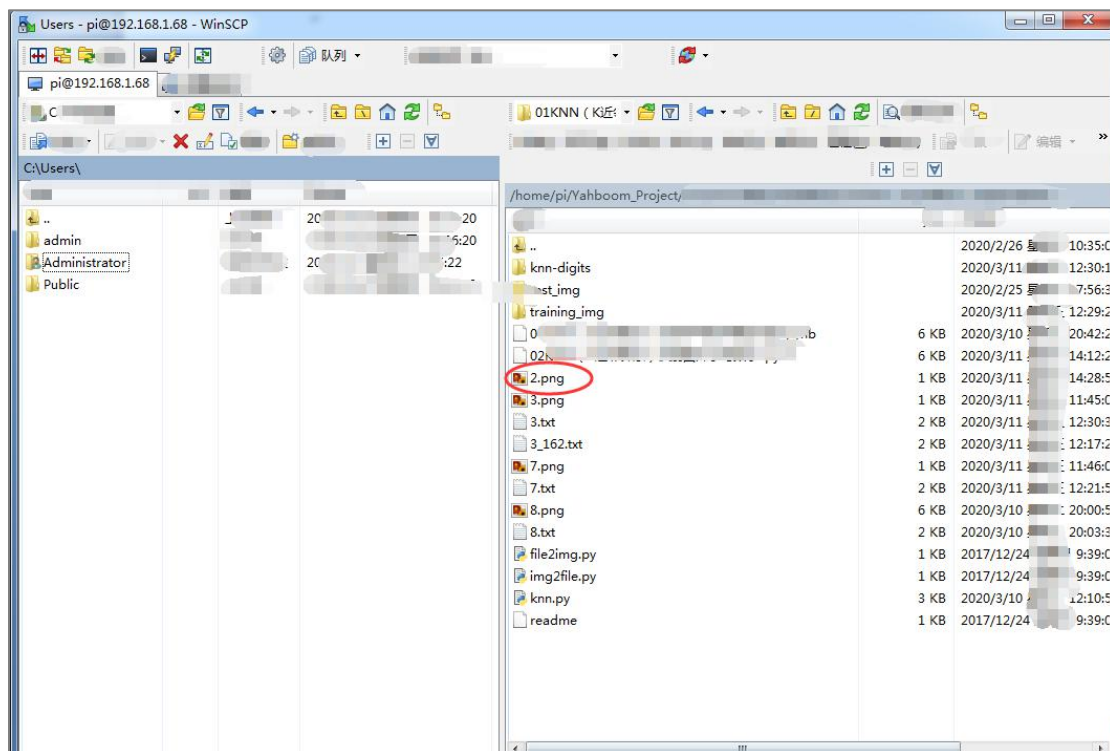


【Correct example】



【Wrong example】

⑤ After drawing, save it as a png image and copy it to the project directory using WinSCP tool.



Code path:

[/home/pi/Yahboom_Project/1.OpenCV_course/05machine_learning/01KNN/KNN_2.ipynb](#)

Code as shown below.

```
#!/usr/bin/env python3
```

```
# -*- coding: utf-8 -*-
```

```
import numpy as np
```

```
import operator
```

```
import os
```

```
# Convert image data to (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
```

```
# KNN Classifier
```

```
def classifier(inX, dataSet, labels, k):
```

```
    # numpy shape [0] return the number of rows in the array, shape [1] returns the  
    number of columns
```

```
    #MDS
```

```
    dataSetSize = dataSet.shape[0]
```

```
    #The inverse matrix
```

```

diffMat = np.tile(inX, (dataSetSize, 1)) - dataSet

#Two-dimensional features are subtracted first and then power

sqDiffMat = diffMat ** 2

#Calculate distance

sqDistances = sqDiffMat.sum(axis=1)

distances = sqDistances ** 0.5

print ("distances:",distances)

#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):

    #Take out the first k element categories

    votelabel = labels[sortedDistIndicies[i]]

    classCount[votelabel] = classCount.get(votelabel, 0) + 1

#reverse

sortedClassCount = sorted(classCount.items(), key=operator.itemgetter(1),
reverse=True)

return sortedClassCount[0][0]

```

```

#Convert image data to txt file

from PIL import Image

```

```
"""
```

```
    Convert image data to txt file
```

```
    :param img_path: path of image
```

```
    :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="")
```

```
#Convert the image to a 32*32 array
```

```
img2txt("3.png", "3.txt")
```

```
#training part
```

```
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)

# Recognize handwritten classification results

fileStr = "2.txt"

classNumStr = int(fileStr.split('.')[0])

vectorTest = img2vector("./2.txt")

result = classifier(vectorTest, trainingMat, hwLabels, 3) # k=3

print("The classification result is:%d, The real result is:%d" % (result, classNumStr))

```

After running the program, we can see the results shown in the figure below.

```

distances: [18.24828759 18.78829423 17.74823935 ... 19.46792233 15.84297952
18.22086716]
sortDistance: [1434 751 593 ... 853 1618 393]
The classification result is:2, The real result is:2

```



! Note:

If the recognition result is different from the real result. Please copy the converted txt file to knn-digits/trainingDigits/, and name it as follows: 2_201.txt, and then re-train, it will can work normally.