# 1. KNN recognizes handwritten numbers

## 1.1. KNN (K-nearest neighbor algorithm) recognizes handwritten numbers

### 1.1.1. 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 feature corresponding to 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

### 1.1.2, Take the recognition of handwritten digits as an example to introduce the implementation of KNN algorithm

1), Dataset

- Training set

The training set is the classified data, which can be found in the directory /home/yahboom/YBAMR-COBOT-EDU-00001/src/yahboom_navrobo_other/KNN/knn-digits/trainingDigits

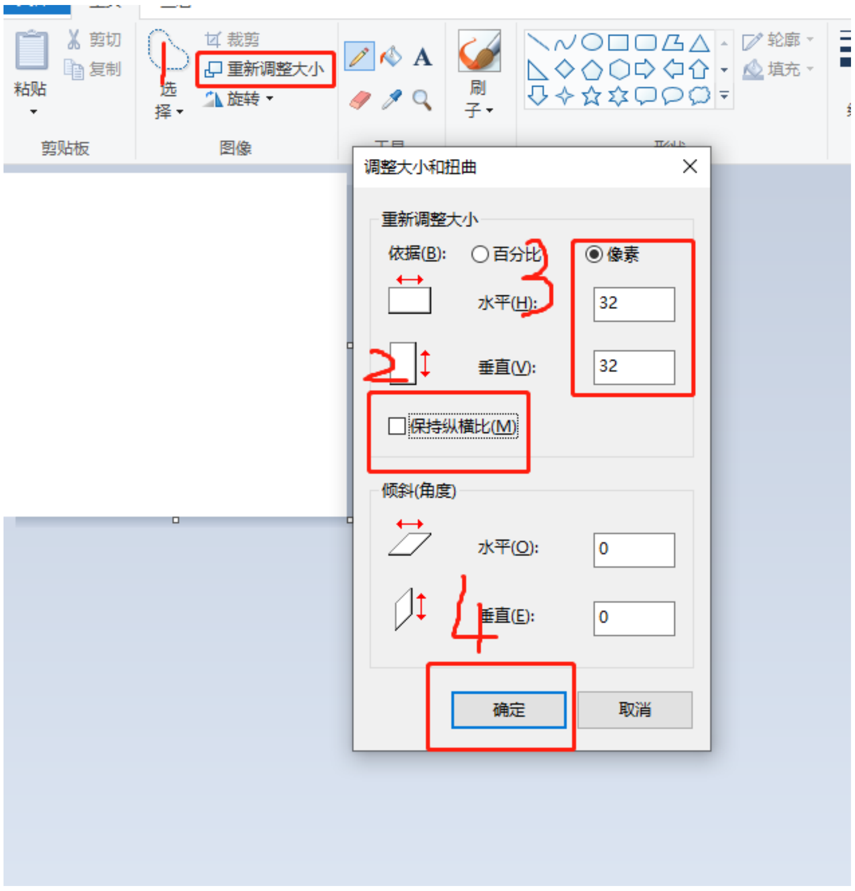
Note: The training set has been trained. If the user wants to train by himself, he needs to back up the original training set first, as the number of training sets is relatively large.

- Test set

The test set is used to test the algorithm. You can refer to the directory /home/yahboom/YBAMR-COBOT-EDU-00001/src/yahboom_navrobo_other/KNN/knn-digits/testDigits

2) Use the drawing function under Windows to make a handwritten digital picture

- Open the drawing software and adjust the resolution to 32*32. You can also use other resolutions, but it is best to fill the entire interval with the drawing, otherwise the error rate is 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)

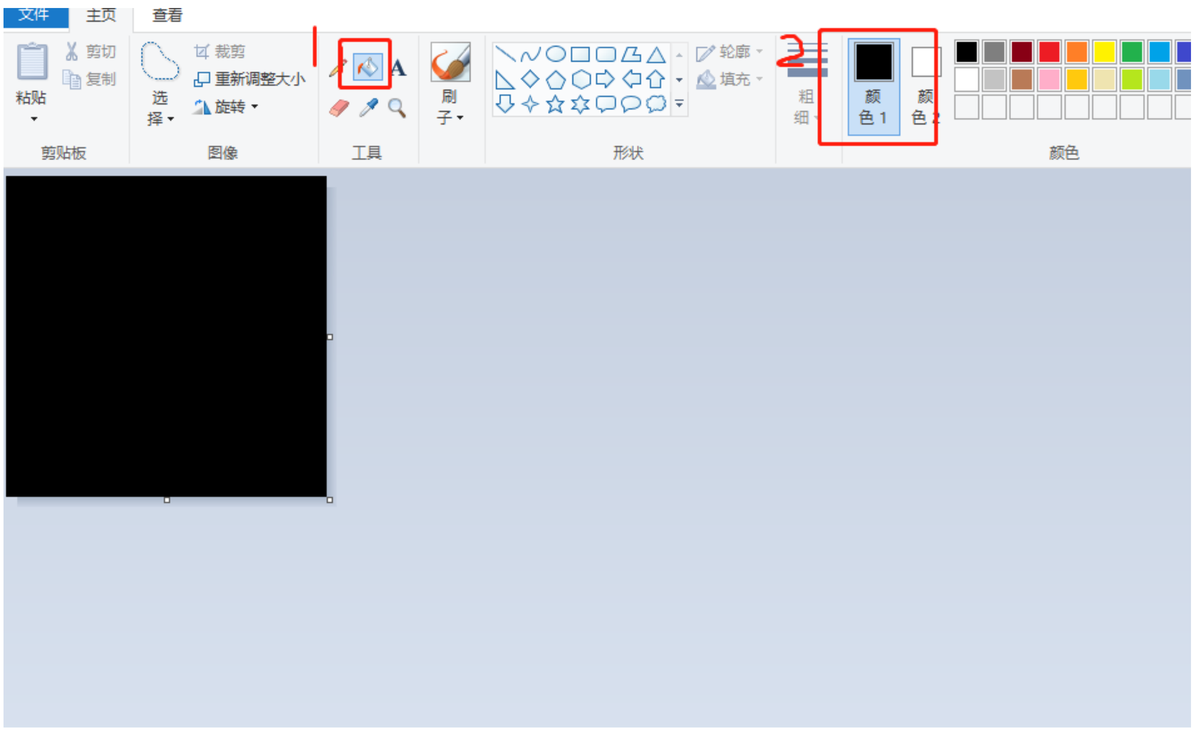
倾斜(角度)

水平(O): 0

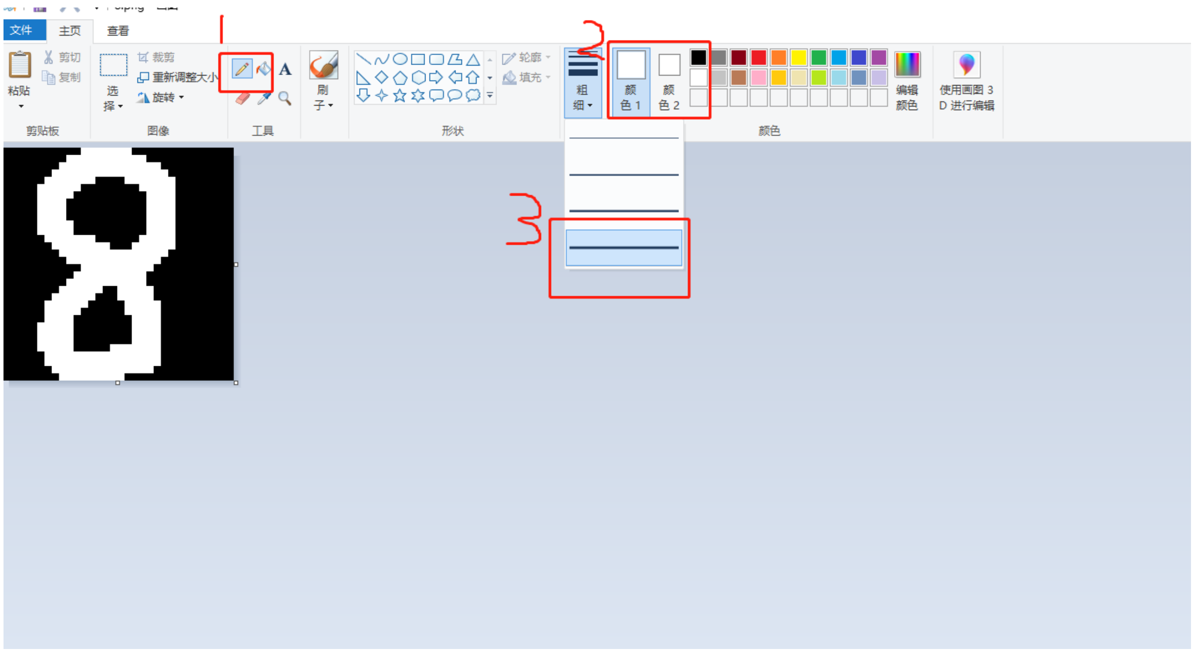
垂直(E): 0

确定 取消

- Press and hold ctrl and slide the mouse wheel up to enlarge the image to the maximum. Select the paint bucket tool and select black to fill the entire screen.



- Select the pencil tool, choose white for the color, and choose the maximum width for the line thickness. 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 through the WinSCP tool

3), convert the image (.img) into text (.txt)

- Code

Code location: /home/yahboom/YBAMR-COBOT-EDU-00001/src/yahboom_navrobo_other/KNN/img2file.py

```python
from PIL import Image

import numpy as np


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='')
img2txt("8.png","./knn-digits/testDigits/8_1.txt")
```

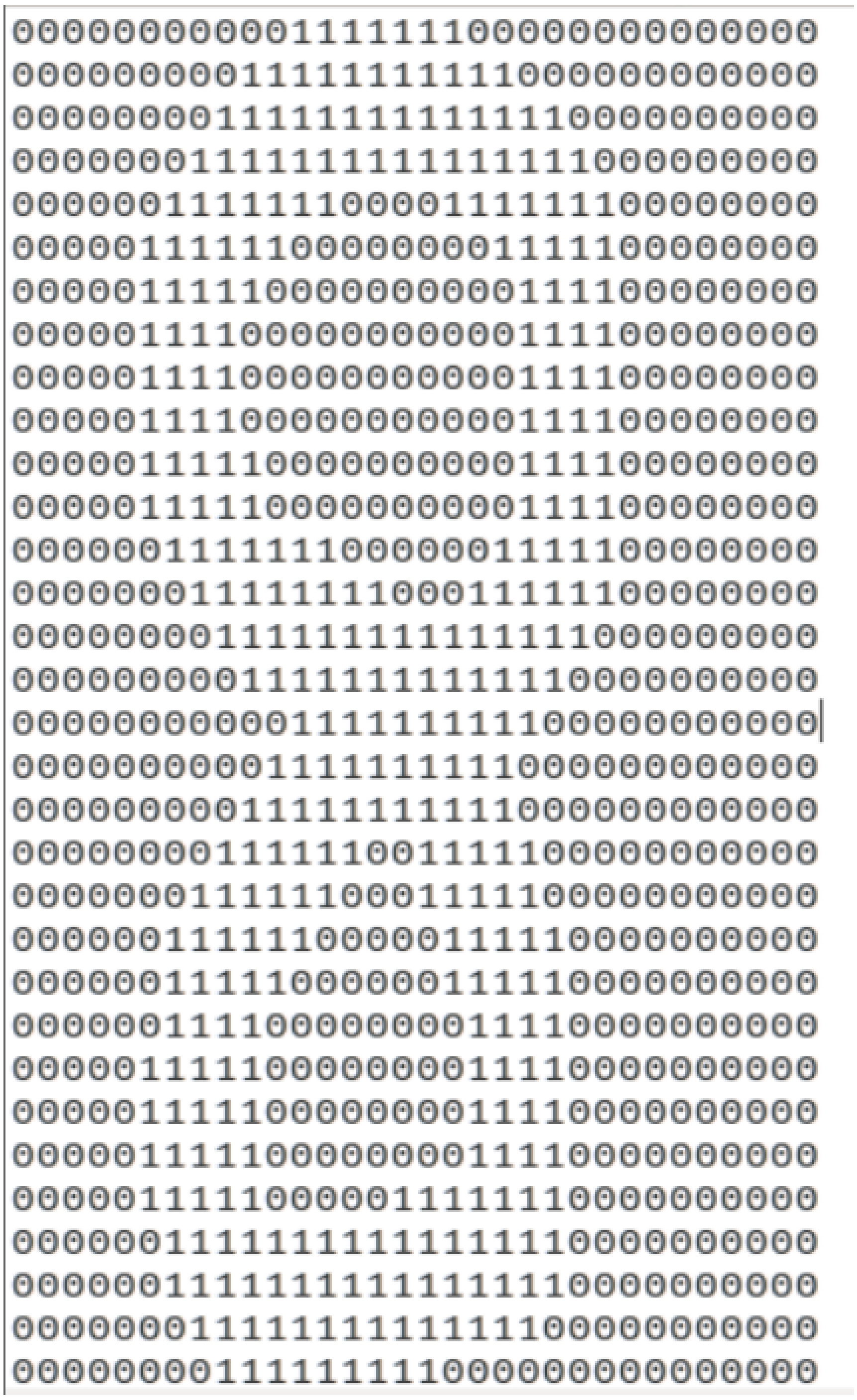
img_path: image file name

txt_name: after conversion, save in the directory /home/yahboom/YBAMR-COBOT-EDU-00001/src/yahboom_navrobo_other/KNN/knn-digits/testDigits, named 8_1.txt

- Run the program

```
cd /home/yahboom/YBAMR-COBOT-EDU-00001/src/yahboom_navrobo_other/KNN
python img2file.py
```

After the program is run, a file named 8_1.txt will be generated in the KNN directory. Double-click it and you can see that it is like this.



```
00000000000111111100000000000000
00000000011111111111000000000000
00000000111111111111110000000000
00000001111111111111111100000000
00000011111110000111111100000000
00000111111000000001111100000000
00000111110000000000111100000000
00000111100000000000111100000000
00000111100000000000111100000000
00000111100000000000111100000000
00000111110000000000111100000000
00000111110000000000111100000000
00000011111110000001111100000000
00000001111111100011111100000000
00000000111111111111111000000000
00000000011111111111110000000000
00000000000111111111100000000000
00000000001111111111000000000000
00000000011111111111000000000000
00000000111111001111100000000000
00000001111110001111100000000000
00000011111100000111110000000000
00000011111000000111110000000000
00000011110000000011110000000000
00000111110000000011110000000000
00000111110000000011110000000000
00000111110000000011110000000000
00000111110000111111110000000000
00000011111111111111110000000000
00000011111111111111110000000000
00000001111111111111100000000000
00000000111111111000000000000000
```

You can see that the part with the number 8 is roughly enclosed as a number 8.

4) Run the KNN recognition algorithm program

```
In the /home/yahboom/YBAMR-COBOT-EDU-00001/src/yahboom_navrobo_other/KNN
directory, open the terminal and run
python knn.py
```

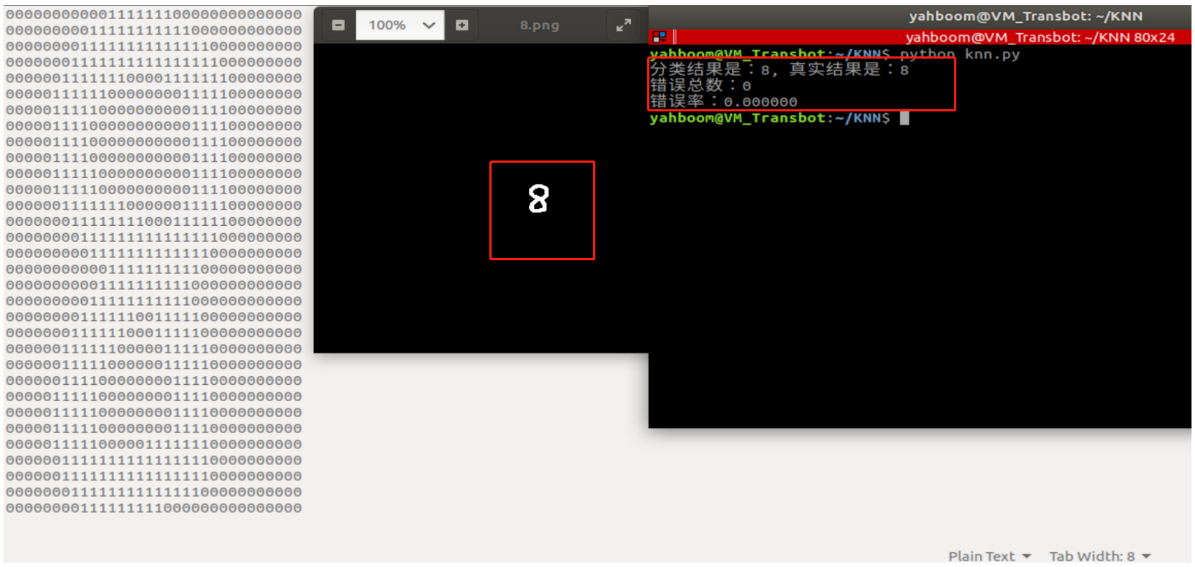
- Code

Code location: /home/yahboom/YBAMR-COBOT-EDU-00001/src/yahboom_navrobo_other/KNN/knn.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Apr 16 14:53:46 2017
```

```
@author: jiangkang
"""

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

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

def classifier(inX, dataSet, labels, k):
    dataSetSize = dataSet.shape[0]
    diffMat = numpy.tile(inX, (dataSetSize, 1)) - dataSet
    sqDiffMat = diffMat ** 2
    sqDistances = sqDiffMat.sum(axis=1)
    distances = sqDistances ** 0.5
    sortedDistIndicies = distances.argsort()
    classCount = {}
    for i in range(k):
        voteIlabel = labels[sortedDistIndicies[i]]
        classCount[voteIlabel] = classCount.get(voteIlabel, 0) + 1
    sortedClassCount = sorted(classCount.items(), key=operator.itemgetter(1),
reverse=True)
    return sortedClassCount[0][0]
# Train first, then test recognition, k represents the k value in the algorithm -
select the first K most similar data in the sample data set, the k value is
generally an integer not greater than 20
def handwritingClassTest(k):
    hwLabels = []
    trainingFileList = os.listdir('knn-digits/trainingDigits')  #Read training
set data
   m = len(trainingFileList)
    trainingMat = numpy.zeros((m, 1024))
    for i in range(m):
        fileNameStr = trainingFileList[i]
        fileStr = fileNameStr.split('.')[0]
        classNumStr = int(fileStr.split('_')[0])
        hwLabels.append(classNumStr)
        trainingMat[i, :] = img2vector("knn-digits/trainingDigits/%s" %
fileNameStr)
    testFileList = os.listdir('knn-digits/testDigits') #Read test set data
    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("分类结果是：%d, 真实结果是：%d" % (result, classNumStr))
        if result != classNumStr:
            errorCount += 1.0
    '''fileStr = "2.txt"
    classNumStr = int(fileStr.split('.')[0])
    vectorTest = img2vector("./2.txt")
    result = classifier(vectorTest, trainingMat, hwLabels, 3)'''
    print("错误总数：%d" % errorCount)
    print("错误率：%f" % (errorCount / mTest))
    return errorCount

handWritingClassTest(1)
```

- Run screenshot



```
00000000000111111100000000000000
00000000011111111111000000000000
00000000111111111111110000000000
00000001111111111111111000000000
00000011111110000111111100000000
00000111111000000001111100000000
00000111110000000000111100000000
00000111100000000000111100000000
00000111100000000000111100000000
00000111100000000000111100000000
00000111110000000000111100000000
00000111110000000000111100000000
00000011111110000001111100000000
00000001111111110001111100000000
00000000111111111111111000000000
00000000011111111111110000000000
00000000000111111111100000000000
00000000001111111111000000000000
00000000011111111111000000000000
00000000111111001111100000000000
00000001111111000111100000000000
00000011111100000111110000000000
00000011111000000111110000000000
00000011110000000011110000000000
00000111110000000011110000000000
00000111110000000011110000000000
00000111110000000011110000000000
00000111110000011111110000000000
00000011111111111111110000000000
00000011111111111111110000000000
00000001111111111111100000000000
00000000111111111000000000000000
```

```
yahboom@VM_Transbot:~/KNN$ python knn.py
分类结果是：8, 真实结果是：8
错误总数：0
错误率：0.000000
yahboom@VM_Transbot:~/KNN$
```

As shown in the figure, the recognized number is 8, which is correct. If the recognition result is different from the actual result, please copy the converted txt document to knn-digits/trainingDigits/ and name it as follows: 8_801.txt, then retrain and recognize it normally.

- Program description

Naming the text file converted from the image, take 8_1 as an example, knn.py will parse this file name in the program, with underscores as the boundary, the front 8 represents the real number, and the back part can be customized, see the code,

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

Training first, then recognition.