Letter recognition using KNN algorithm (machine learning).

Table of content:

1) Introduction

The letter recognition program is written in python and contains two python programs Gendata.py and Trainandtest.py.

When the Gendata.py is run the user enters the letters from the keyboard then the program automatically recognize the letters entered and highlight them in the program window.

Then the user has to run the second program trainandtest.py in order to show the user the letter that are in the image specified in the program.

In the case of my program it is test1.png so it will show the user ABC123.

Python was used in this program in opencv, so cv2 library was included.

This method was used in this project, and it is implemented using openCV.

2) KNN

KNN works by analogy. The idea is that you are what you resemble.

So when we want to classify a point we look at its K-closest (most similar) neighbors and we classify the point as the majority class in those neighbors.

KNN depends on two things: A metric used to compute the distance between two points and the value of "k" the number of neighbors to consider.

When "k" is a very small number KNN can overfit, it will classify just based on the closest neighbors instead of learning a good separating frontier between classes. But if "k" is a very big number KNN will underfit, in the limit if k=n KNN will think every point belongs to the class that has more samples.

3) Explanation of the program

This character recognition program has 2 python files, the first one recognize the characters in the training_chars.png using the keyboard. A red scare is drawn on the letter that has to be selected. OpenCV and numpy are two key libraries used in this program, and KNN is implemented using openCv. After running GenData.py two txt files are created in the program's folder allowing TrainAndTest.py to get the appropriate information about the letters and numbers in the training_chars image. When it is executed the specified image in the python program is recognized then whatever letters and numbers in it would be displayed.

3.1) GenData.py

- Reading the training numbers image

```
imgTrainingNumbers = cv2.imread("training_chars_small_test.png")

if imgTrainingNumbers is None:
    print("error: image not read from file \n\n")
    os.system("pause")
    return
```

- Possible chars needed are digits 0 through 9 put in a list intValidChars

```
intValidChars = [ord('0'), ord('1'), ord('2'), ord('3'), ord('4'), ord('5'), ord('6'), ord('7'), ord('8'), ord('9'),
                 ord('A'), ord('B'), ord('C'), ord('D'), ord('E'), ord('F'), ord('G'), ord('H'), ord('I'), ord('J'),
                 ord('K'), ord('L'), ord('M'), ord('N'), ord('O'), ord('P'), ord('Q'), ord('R'), ord('S'), ord('T'),
                 ord('U'), ord('V'), ord('W'), ord('X'), ord('Y'), ord('Z')]
```

- Draw a rectangle around each countour as the user is asked for

```
cv2.rectangle(imgTrainingNumbers,
              (intX, intY),
              (intX+intW,intY+intH),
              (0, 0, 255),
              2)
```

- Croping the char out of threshold image and resize image this will be more consistent for recognition and storage.

```
imgROI = imgThresh[intY:intY+intH, intX:intX+intW]
imgROIResized = cv2.resize(imgROI, (RESIZED_IMAGE_WIDTH, RESIZED_IMAGE_HEIGHT))
```

- If esc key was pressed the program exits

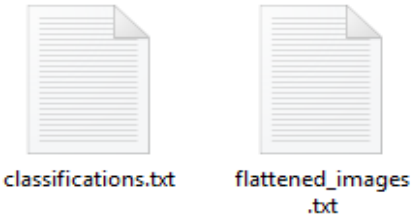
```
if intChar == 27:
    sys.exit()
elif intChar in intValidChars:
```

- Convert classification lists of chars will convert to float later before writing to file

```
try:
    npaClassifications = np.loadtxt("classifications.txt", np.float32)
except:
    print "error, unable to open classifications.txt, exiting program\n"
    os.system("pause")
    return
```

- If the image is not read correctly send an error message

```
if imgTestingNumbers is None:
    print "error: image not read from file \n\n"
    os.system("pause")
    return
```



- After running GenData.py and completing the process of selecting the letters, two txt files are created in the program's folder and are required in the TrainAndTest.py

3.2) TrainAndTest.py

- Reading in training classifications

```
try:
    npaClassifications = np.loadtxt("classifications.txt", np.float32)
except:
    print "error, unable to open classifications.txt, exiting program\n"
    os.system("pause")
    return
```

- Reading in training images

```
try:
    npaFlattenedImages = np.loadtxt("flattened_images.txt", np.float32)
except:
    print "error, unable to open flattened_images.txt, exiting program\n"
    os.system("pause")
    return
```

- Reading in testing numbers image

```
imgTestingNumbers = cv2.imread("test1.png")

if imgTestingNumbers is None:
    print "error: image not read from file \n\n"
    os.system("pause")
    return
```

- Creating a KNN instance

```
kNearest = cv2.ml.KNearest_create()
```

4) The output

- The output: for the program to run perfectly the user must run it several times to get the right results.

```
(base) C:\Users\Admin\Desktop\python_pro>python TrainAndTest.py

ABCT23
```

- The image used test1.png

- The program's first page : the letters must be then selected.


training_numbers.png
0 1 2 3 4 5 6 7 8 9
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
0 1 2 3 4 5 6 7 8 9
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
0 1 2 3 4 5 6 7 8 9
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
0 1 2 3 4 5 6 7 8 9
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
0 1 2 3 4 5 6 7 8 9
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z