Number Recognition Using K-Nearest-Neighbors Classification

Akshay Nagpal

Computer Science Department,
The Northcap University (Formerly ITM University, Gurgaon)
HUDA Sector 23-A Gurgaon – 122017
Email : akshay12csu022@ncuindia.edu

ABSTRACT

In this paper, I have presented an application of k-nearest-neighbors classification algorithm to single digit recognition. A classifier function was developed which could recognize the number present in the image. The input to the classifier function consists of PNG and JPEG files containing a single digit. The classifier recognized the number by using the dataset of text files given to it. It was observed that the accuracy of the classifier improved with every iteration of the program as the program learnt from its earlier errors. Finally, further improvements and future scope of the model has been discussed.

Keywords: number recognition, k nearest neighbors, classification, supervised machine learning

1. INTRODUCTION

The primary purpose of this paper is to illustrate a method to apply K-nearest-neighbors classification algorithm to digit recognition [1]. Firstly, a PNG/JPG image containing a single digit was given as an input to the model. The image pixels were processed and the image was converted to a text file containing only zeroes and ones. Further, the text file's content was converted to a vector to be used in Python. Finally, this vector was given as an input to the KNN classifier function programmed using Python. The classifier function found the 3 most identical vectors (K nearest neighbors where, K=3) by matching the input to the training dataset and gave the

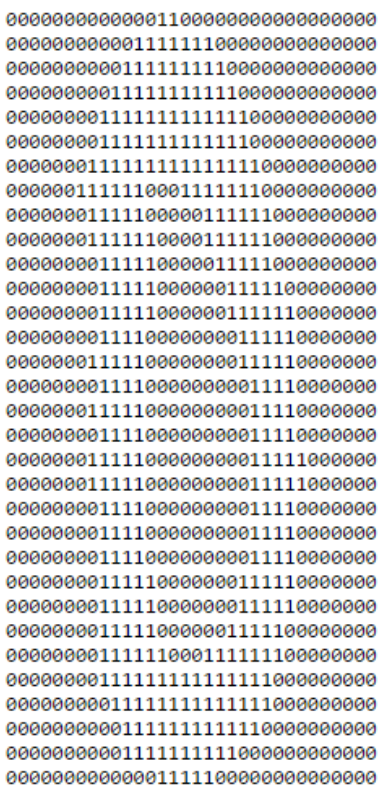

Figure 1(a)

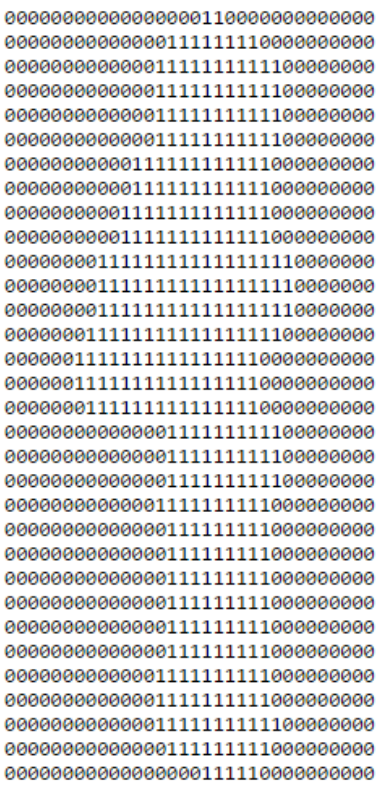

Figure 1(b)

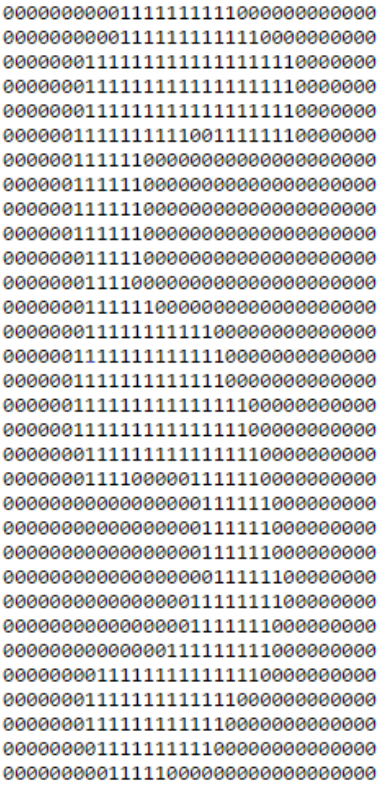

Figure 1(c)

output as the vector most identical to the input among the 3 vectors selected earlier. The label of that vector was the numerical digit that the vector represented, which was given as the final output.

2. DATASETS

The dataset [2] used to train the classifier consists of text files converted from images of single digits. All these images were black and white and were resized to 32x32 pixels in size and then converted to text files as shown in Fig.1.

3. DIGIT RECOGNITION PROCEDURE

The images to be recognized are first converted to binary text files. Then, the data of the text file is converted to vector for further operation. Finally the vector is compared to the training dataset to recognize the digit in the image. Three procedures were developed using Python, which were used for specific stages of the digit recognition model (Fig. 3).

A. Converting Image File to Text file

This procedure takes the filename and pixel data of the image and creates a text file using the following algorithm. Refer to Appendix for more on RGBA.

PROCEDURE ImageToText

```
/*
filename – Name of the file to be converted to text format
imagedata – Pixel data of the file to be used
pixeldata is the data of every pixel in the form of RGBA (255,255,255,255)
*/
Input: filename, imagedata
Begin PROCESS
1  Open new textfile in write     mode.
2   for each pixeldata in imagedata
3      if pixeldata: = (255,255, 255,255) then write '0' to textfile
4      else write '1' to textfile
5     end if
6  end for
End PROCESS
Output:  textfile
```

B. Converting Text file data to Vector

This procedure extracts the binary data from the text file and converts it into a vector for further operation.

PROCEDURE TextToVector

```
/*
filename – name of the text file to be converted to vector form
returnVect  vector to be returned
32 X 32 is the size of text file (1024 bytes)
*/
Input: filename
Begin PROCESS
1   Initialize returnVect to size [1][1024]
2   open filename.txt
3   for i in range(32)
4                 read line from filename.txt
5           for j in range(32)
6                       returnVect[0, 32*i+j] = int(line [j])
7     end for
8  end for
end PROCESS
Output: returnVect
```

C. Classifying vector using k-nearest-neighbors classification

This procedure uses the k-nearest-neighbors classification algorithm and makes a list of various guesses that the machine makes. The K closest neighbors are then found and the closest among them is selected as the final result.

***PROCEDURE** Classifier*

```
/*
inX – The input vector to classify
dataSet - Training examples
labels - Vector of labels
k - No. of nearest neighbors to use
*/
Input: filename
Begin PROCESS
```

```
1     diffMat = inX - dataset
2     sqDiffMat = diffMat^2
3      For each row in sqDiffMat
4              Append to sqDistances sum of all elements of  that row
5      end for
6     sqDistances = single column matrix containing sum of each row of
sqDiffMat
7      distances = sqDistances^0.5
8       sort the distances matrix in ascending order and select k smallest
values (k=3 in this experiment)
9       match the  minimum distances selected in step 8 to their labels in
the respective text file's name.
10     min_label =  label with the highest frequency from the k matches.
End PROCESS
Output: min_label
```

The complete running code in Python can be obtained from the link [3] given in references.

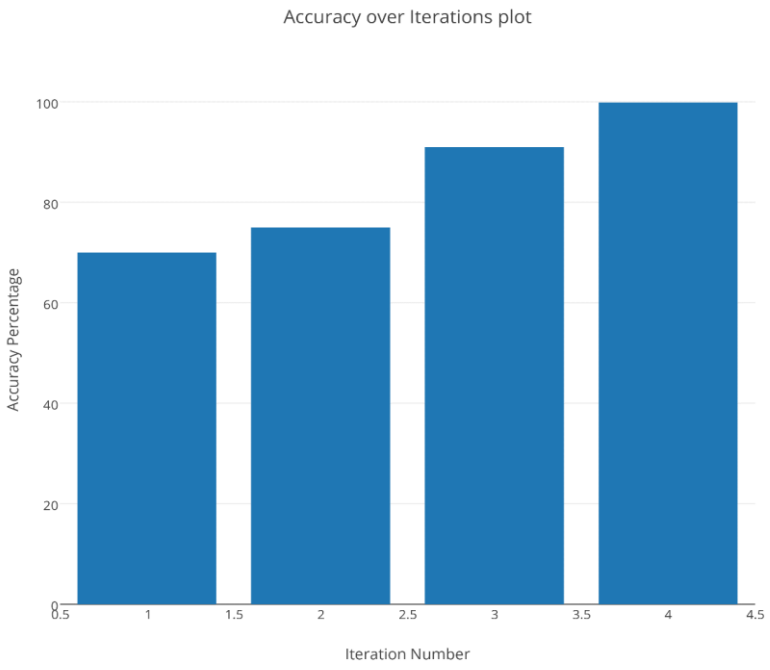

Figure 2. Accuracy / Iteration Plot

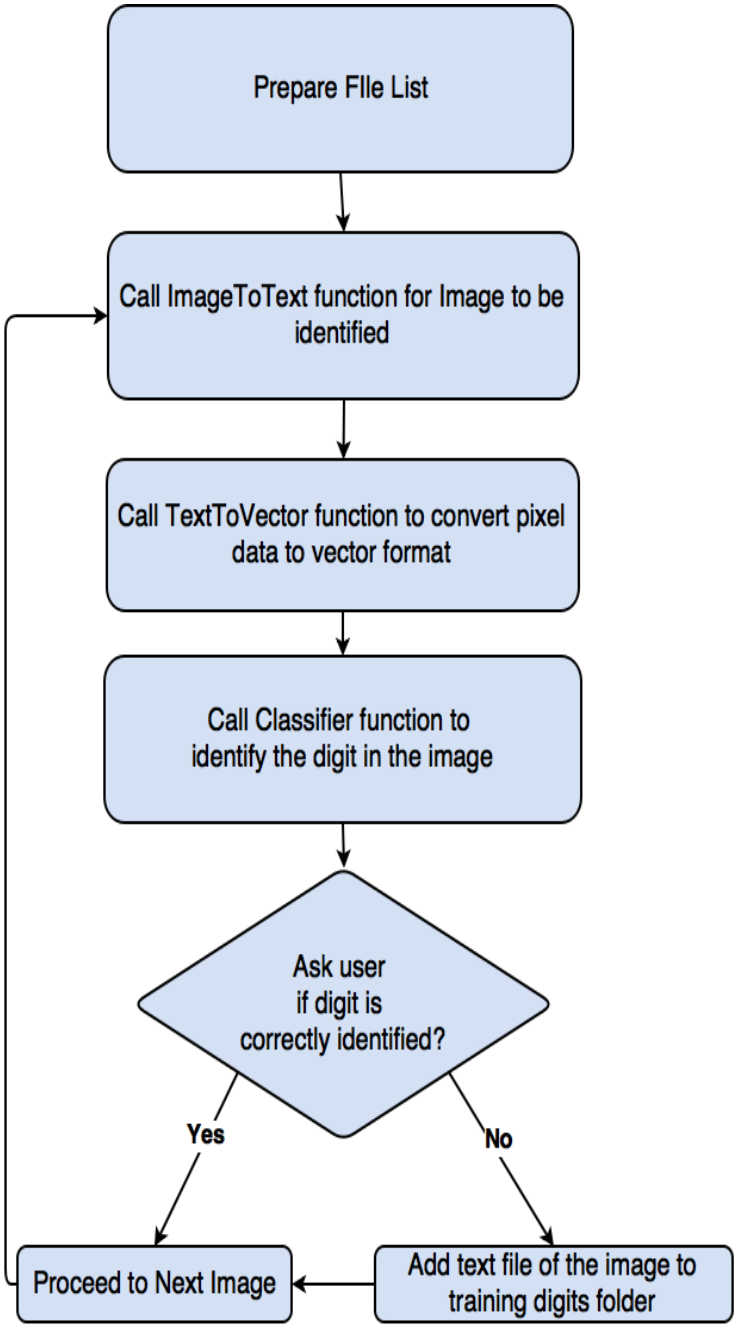

Figure 3. Stages and Program Flow

4. ACCURACY CALCULATION AND IMPROVEMENTS

Accuracy percentage was calculated by using the formula

$$\left(\frac{Number\ of\ images\ correctly\ recognized}{Total\ number\ of\ images\ given\ as\ input}\right) \times 100$$

After recognizing the digit in each input file, the program prompts the user to verify if the number was correctly recognized or not. If the program correctly identifies the digit user inputs 'Y' and the program proceeds to the

next input image. If the program's guess is incorrect then the user enters 'N' and the binary text file created from the incorrectly recognized image is added to the training digits folder.

During the next iteration, the previously added file is also included in the matching process. With the following iterations, the frequency of the added file increases hence, the accuracy percentage increases.

5. RESULTS

After 4 training iterations, the accuracy percentage on testing set (15 images having single digits from 0 to 9) was 99.9 % (Fig. 2).

6. CONCLUSION

K-nearest-neighbors classification was successfully applied to identify numbers in images having single digits. This model can be further extended to recognize numbers in images having multiple digits. The results showed that such an implementation can be used in various areas of artificial intelligence like number plate recognition, handwriting recognition, and educational robotics

7. REFERENCES

[1] Höppner, Frank. *Fuzzy cluster analysis: methods for classification, data analysis and image recognition*. John Wiley & Sons, 1999.

[2] E. Alpaydin, C. Kaynak. (2010). *Optical Recognition of Handwritten Digits Data Set* [Data file]. Retrieved from https://github.com/pbharrin/machinelearninginaction/blob/master/Ch02/digits.zip

[3] Akshay Nagpal, *Number recognition using Machine Learning*, (2015), Python code on GitHub, https://github.com/akshaynagpal/number_recognition/blob/master/NUMBER_RECOGNIZER.py

8. APPENDIX

8.1 RGBA Color Model

RGBA is the abbreviation for Red Green Blue Alpha. It is a use of RGB color model with Alpha indicating opacity. The RGB model is color model in which red, green, and blue light are mixed together in various ways to generate a broad array of colors. The name of the model comes from the initials of the three additive primary colors, red, green, and blue.