Digit Recognition Documentation

Table of Contents

1. Experimental Results....3
   A. K value....3
   B. Efficiency of the kNN algorithm....3
   C. Colors....3
   D. Size of the images....3
   E. Further Improvement....3
2. References....4

GitHub Repository : https://github.com/Leopounet/Digit-Recognition

1. Experimental Results

A. K value

During the creation of this program, the main problem has been to determine what K value should be used. After many attempts at creating an algorithm that could compute the best K, it became obvious that it would not work, the main reason being efficiency (the algorithms designed would have to be run for hours before giving any results). Therefore, K has been chosen after many researched on the Internet (cf. References).

B. Efficiency of the kNN algorithm

Apart from finding the best K value, another experimental result that has been witnessed was the efficiency of the algorithm. The kNN algorithm is not very precise at all, especially the digit “7” is often interpreted as a “1”, the reason being that there the digit “7” is made of a “1” with an extra dash. More surprisingly, “9”s and “4”s also tend to be confused (although it depends on the writing of the user). Finally, “3”s and “8”s are also too close for the algorithm to be precise on those digits.

C. Colors

The MNIST data set is all made of shades of gray, therefore when trying to analyse coloured images the algorithm becomes very inaccurate. To go around this problem, the best way found for this software was to convert the coloured images into black and white images (not gray scaled). This allows the use of non gray scaled images but reduces a little bit the accuracy of the algorithm (negligible, for the algorithm is already not that precise). The conversion is made in two steps, first the main colour (e.g the colour that has the most number number of pixels of its colour) which probably is the background colour is set to black, second the other colours of the image are set to white. This algorithm is accurate when there are only two colours but may struggle if there are more than two colours.

D. Size of the images

The size of the images draw or uploaded are not 28 pixels by 28 pixels, therefore they need to be resized, this has a negligible effect on the accuracy of the algorithm as long as the user does not upload a too large image. The resize method of Java Swing being already sufficient, the size of the images has not been a problem. It is important to note that if the provided image is 28 pixels by 28 pixels, the algorithm has a noticeable greater accuracy.

E. Further Improvement

A way of upgrading the software drastically would be to use another algorithm to determine the digit. Some possibilities might include a neural network or support vectors machines. These would make the software much more accurate but would also either slow the processing down or add an extra step during which the network learns (this phase could be long and that is why it is a drawback).

2. References

Yann.lecun.com. (2019). *MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges*. [online] Available at: http://yann.lecun.com/exdb/mnist/ [Accessed 8 Mar. 2019].

Docs.oracle.com. (2019). *Java Platform SE 7*. [online] Available at: https://docs.oracle.com/javase/7/docs/api/ [Accessed 8 Mar. 2019].

Rajaguru, H. and Prabhakar, S. (n.d.). *KNN Classifier and K-Means Clustering for Robust Classification of Epilepsy from EEG Signals. A Detailed Analysis*. pp.29-34.

Anderson, J. (1995). *An introduction to neural networks*. Cambridge, Mass.: MIT Press.

Campbell, C. and Ying, Y. (2011). *Learning with support vector machines*. [San Rafael, Calif.]: Morgan & Claypool.