Computer Science Department
Pattern recognition project
KNN Digit Recognition

Overview :

The aim of this project is to classify a handwritten digit from 0 to 9 using K-nearest Neighbour algorithm and test the accuracy of the algorithm on the test set or on a new painted image.

The project was made using C# Windows Form Application.
We used the minst handwritten digits dataset as our test images and the training images.

The project features 3 modes :
1-Single image test mode: The user chooses an image from the test set and classifies that image.

2-Range image test mode: The user chooses a range from the test images then classifies this range , a test result form is shown which contains the confusion matrix and the total accuracy.

3-Drawing mode: The user draws an image then classifies that image.

Reading The datasets :

Class ReadingInput contains 2 functions for reading the mindst training set and test set :
- readIdx3Train()
- readIdx3Test()

which then puts them in 2 lists of DigitImage objects.

Digitimage.cs contains a 2d array that contains the pixels and label for each image.

Classification of images :

KNN.cs uses k-nearest neighbours algorithm to classify a given image or a range of them from the test set.

the class contains function Classify() which is overwritten to support multiple parameters for the range or a single image.

Given a single image it calculates the distance between this image and the training set and chooses the k-nearest images to classify that image.

In case of the of the range the function calculates the confusion matrix and the accuracy for that range.

The distance between 2 images is calculated using euclidean distance between their pixels.

GUI

MainForm.cs contains 3 buttons for the use to choose the mode.

TestForm.cs contains a text box the value of k and the index of the test image and a classify button.

RangeTest.cs contains 2 text boxes for the range and a classify button
this form opens ConfusionMatrix.cs to show the confusion matrix in a data grid view.

DrawForm.cs contains a text box for the k value and a picture box for the user to draw into then a classify button.

Analysis

1-correctness

When run on the 10,000 test set the algorithm's accuracy was 97.05%
but on the painted images it sometimes failed to correctly predict the image due to the fact that the digit was not drawn in the center of the image, or that the digit was not in the same size as the images in the training set because it depends on the distance between images.
this shows a disadvantage of using the k-nearest algorithm when using a simple distance function or a similar training set.

2-Running time

The naive implementation of this algorithm normally tends to be very slow because of the fact that it doesn't have a training step and totally depends on the size of the dataset.

In our case the the naive implementation for classifying a single image took about 28*28*60000 operations where 28*28 is the width and height of the image and 60,000 is the size of the dataset.

There are ways to enhance the running time by using K-D trees or inverted index which could reduce the running time from o(n) over the entire dataset to just o(logn) by grouping similar sample by their features.

3-Memory usage

The algorithm needs to store the entire dataset O(n) during any classification process since there is no training step.

So if the size of your datasets is huge you might want to reconsider using k-nearest algorithm.

This project was made by :

1-Omar Mohamed Abdel-latif

2-Mohamed Ahmed Ismail.

3-Abdallah Hassan Ali

4-Mostafa Zaghloul Abdel-rahman