# IN4085 Pattern Recognition
# Final Project
# "Classification of Handwritten Digits in Different Scenarios"

Tung Phan, ttphan, 4004868 Kevin van Nes, kjmvannes, 4020871

January 14, 2015

## 1 Introduction

This report concludes the final assignment of the TI4085 Pattern Recognition course. For this assignment, two classifiers had to be constructed for the classification of handwritten digits. One classifier was made for the scenario in which a lot of training data is available, whilst the other classifier was made for the scenario in which very few training data is available.
This report contains the details about the different phases in constructing the classification system, as well as the design choices that were made and why these were made. Furthermore, the system has been ran against a set of benchmark data, about which more information can also be found in this report. After successfully having created the system, it was ran against a so-called 'live test', which is a test in which real, self-written data was used to test the system against. Finally, some recommendations will be given to the company that will receive the system.

## 2 Preprocessing

The first phase we went through during this project was the phase in which we preprocessed the data. This was a necessary step, because the data (i.e. the handwritten digits) differed tremendously amongst each other, even within the same class. An example of this would be that a '4' can be written in different ways or that digits would be sheared in various directions. To solve these inequalities, a few steps were taken to normalize the data, as to try to make the data as similar as possible, both inter-class and intra-class.

### 2.1 Boxing and resizing

The first step that was taken to preprocess the data was to normalize all the data objects in terms of their size. In order to do this, the PRTools functions 'im_box' and 'im_resize' were used. We decided to resize the images to 32x32 pixels (total: 1024 pixels), which later turned out to give better results than images of 64x64 pixels and images of 16x16 pixels. This is probably related to the fact that we use the data images' pixels as our features, so that smaller/larger images would have too few or too many features, respectively.

## 2.2 Morphology operations

After having normalized the size of the digits, we looked at the images and saw that some of them contained noise. There were also digits that looked incomplete, by which we mean that these digits contained holes that were not supposed to be there. These holes can be seen as another form of noise. We investigated how to solve both of these problems and it turned out that morphology operations were the key to these problems, particularly the closing and opening operators.

Firstly, we attempted to fill up the small holes in the images using a closing operator combined with a disk-shaped structuring element of size 1. A closing operator is an operator which first performs a dilation and then an erosion operation. The disk shape (which looks like an addition operator) was chosen to deal with the pixel-like structure of the digits.
After the closing operator had run over an image, noise pixels that were not part of the digits themselves were removed by using an opening operator. An opening operator is an operator which first performs an erosion and then a dilation operation, which makes it the opposite of a closing operation. Again, a disk-shaped structuring element of size 1 was used to perform the opening operator.

## 2.3 Shearing for normalization of diagonally orientated digits

The last major step of preprocessing that was taken was the shearing of all diagonally orientated characters. Because of the differences in the handwritings of people that contributed to the MNIST database, some digits of the same kind might look very different from each other, just because they are orientated differently (i.e. skewed). The solution that we implemented in our system consists of two steps: firstly, we use the central moments of the image to determine the orientation angle of the image (REFERENCE STATISTICAL MOMENTS). Secondly we used the orientation angle to skew the digits in the right direction using a simple transformation matrix (REFERENCE PAPER ABOUT SKEW CORRECTION). The orientation angle was determined in the following way:
Firstly, we determined the moments of the image (i.e. the digits). This can be done using the following formula:

$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y),$$

where $I(x, y)$ is the intensity of a pixel (in our case: 0 or 1) at position (x,y).
With this formula, the second-order central moments $\mu_{11}$, $\mu_{20}$ and $\mu_{02}$ were determined by using the following formula:

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})(y - \bar{y}) f(x, y),$$

where $f(x, y)$ is pixel intensity of the image at position (x,y) and where $\bar{x} = \frac{M_{10}}{M_{00}}$ and $\bar{y} = \frac{M_{01}}{M_{00}}$.

Finally, using the three central moments that were calculated from the previous formula, we were able to find the orientation angle of each digit, which were determined by:

$$\Theta = \frac{1}{2} arctan(\frac{2\mu_{11}^{'}}{\mu_{20}^{'} - \mu_{02}^{'}}),$$

where $\mu'_{11} = \frac{\mu 11}{\mu 00}$, $\mu'_{20} = \frac{\mu 20}{\mu 00}$ and $\mu'_{02} = \frac{\mu 02}{\mu 00}$.

After having obtained the orientation angles of the images, we were able to skew them to a vertical orientation by using a simple shearing transformation matrix, which we made as follows:

$$\begin{bmatrix} 1 & 0 & 0 \\ sin(0.5\pi - 2\Theta) & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The sinus function in this matrix is used because $sin(0.5\pi)$ represents the y-axis, from which $\Theta$ is subtracted to skew the image in the correct angle.

The combination of the morphological operations and the shearing of the digits resulted in a much better representation of the majority of the digits.

# 3 Feature Extraction

After the images had all been preprocessed, we wanted to reduce the amount of features in each image (i.e. the pixels). It was found that Principal Component Analysis (PCA) could be used to reduce the amount of features, while keeping as much of the important data as possible. By doing PCA on our images, the most 'important' pixels would be retained, while less important pixels would be dismissed.

To hold on to as much of the important data as possible, we did PCA on our data's features with a parameter of 0.99, meaning that 99% of the total variance of the features would be preserved. As noted earlier, our preprocessed images consist of 1024 pixels each. Our PCA managed to reduce this amount to 625, which means that 399 pixels made up for only 1% of the total variance of the pixels.

# 4 Classifiers

Now that all of the digit images have been preprocessed and features have been extracted using PCA, the next step was to train the classifiers of our system. Classifiers had to be created for two different scenarios. In the first scenario, Scenario 1, the system is trained once with a large amount of training data, whereas in the second scenario, Scenario 2, the system is trained for each batch of digits, meaning the amount of training data is extremely small.

In the subsections below, the classifiers that we trained for both scenarios will be discussed. Furthermore, the results of what we deemed to be the best classifier for each scenario will be given, based on the classification errors that we produced for each classifier.

### 4.1 Scenario 1

Scenario 1 is the scenario in which the system is trained with a large amount of training data (at least 200 and at most 1000 objects per class). For such a scenario, it is best to use a classifier that performs well when there is low bias and high variance. Classifiers that we found to be fit for this task were: KNN classfiers and SVM classifiers. (REFERENCE HIER)

### 4.2 Scenario 2

Scenario 2 classifiers en percentage dat uit eigen tests kwam. (gemiddelde van 5/10 tests?)

## 5 Benchmarking

Korte intro

### 5.1 Classifier Scenario 1

Benchmark resultaten + vergelijking eigen tests (evt. conclusies trekken uit verschillen/overeenkomsten)

### 5.2 Classifier Scenario 2

Benchmark resultaten + vergelijking eigen tests (evt. conclusies trekken uit verschillen/overeenkomsten)

## 6 Live Test

After the system was tested against benchmark data, we decided to try and test the system against 'live data', which consisted of actual handwritten digits. For each of the digits 0, 1, ..., 9 we wrote six digits each on a piece of paper. This piece of paper was then scanned into a computer. After having scanned in the paper, we were able to segment and preprocess the digits to make them ready for classification.

### 6.1 Segmentation and Preprocessing

Firstly, the 60 handwritten digits were turned into a binary image, after which they were imported into a third-party MATLAB-app, called SegmentTool (REFERENCE HERE). This tool is able to perform segmentation using different kinds of filters and parameters. To segment our digits, we used a Sobel-filter. This was chosen after experimenting manually to retrieve the segmented image that looked best.

The segmentation resulted in different kinds of noise across the total image containing all digits. These kinds of noise were the same as the kinds of noise that appeared in the original MNIST digits. To make sure all images would be cut out correctly in the next step and that no noise would be cut out as if it were a digit, a closing and opening operation were performed on the total image to reduce the noise to a minimum. Next, we cut each digit into its own image by using the MATLAB function *regionprops*. This resulted in each digit being cut out correctly, making them ready for more preprocessing. Since

the noise reduction operations have already been applied to the digits, the only preprocessing steps that still had to be taken were the size and orientation normalizations.

### 6.2 Classification and Results

At this point most of the newly handwritten digits look very similar to the digits contained in the database. This means that it was time to use the digits as test data for our classification system.

## 7 Recommendations

## 8 Conclusion