Master report, IDE1278, December 2012

Quiz corrector

Handwritten digits recognition

Mohsen Masrour

Master thesis

School of Information Science, Computer and Electrical Engineerin

Quiz corrector
Handwritten digits recognition

Master's Thesis in Intelligent systems

December 2012

Author: Mohsen Masrour

Supervisor: Prof. Josef Bigun

Examiner: Prof. Antanas Verikas

School of Information Science, Computer and Electrical Engineering
Halmstad University
PO Box 823, SE-301 18 HALMSTAD, Sweden

Preface

I would like to thank my supervisor and mentor Josef Bigun who helped me in all parts of the project. I mention without his precious help, it would have been impossible for me to do this project alone. His thoughtful ideas helped me many times. I am glad that he gave the chance to work on this project that taught me very new things in machine learning.

Mohsen Masrour,
Halmstad University,
December 2012.

Abstract

In this project, I want to produce a software method, which can correct quiz papers. Quizzes are multiple-choice questions and the participant should write her/his personal number on the quiz form. Since the Swedish personal numbers are unique, it is sufficient to recognize them to establish the identity of the quiz-participant. Recognition of this number is important also for correcting the quiz, since the questions and/or answers can be adapted to each participant. The issue is thus to recognize hand written numerals on quiz-forms. I used image processing to find suitable features for automatic classification relying on; Logistic regression, or Neural network.

Contents

Figures

1 Introduction

Images exist naturally in the World of humans; we sense them, we read what we sense as images, and get different kinds of information from them. However, if machines need to use them in the same way as humans do, the machine should interpret the captured images too. Reading the text in a picture is known as OCR [1], optical character recognition. This process converts an image of a text to a series of machine codes, which the machine can “understand”, i.e. put together so that plain English words can be formed from read characters.

1.1 Goal of the project

This work has been commissioned by BigSafe Technology AB, Stockholm.

In this project, the problem is to recognize

-handwritten numerals which students should write themselves.

There is a collection of different Spirals to be detected. Together they are like an alphabet and can express a word. They are interesting because their detection is robust to noise caused by rotation and zooming. Detection of them is very important to register the quiz-form (as shown in Fig.1), i.e. to make sure that the squares on the quiz forms filled in by different participants are always at the same place. As there exists already a robust algorithm for spiral detection, it was not included in this study. Knowing the positions of two spirals and the size of the paper (scale) it is straight forward to rotate and translate every scanned image (representing quiz answer sheets) such that the centers of all four spirals appear at the same pixel locations for all images (e.g. by a combined use of Matlab functions *cp2transform* and *imtransform*). Recognition and registration was therefore assumed to have been already applied to images in this study.

For answer boxes, I only need to know which squares are filled. This can be done by measuring the average mean value of the pixels in a square. If it is above a threshold then it is filled, else it is empty. As the issue is resolved by a simple method that seems to work fine, I did not study it in the project.

Issues like having more than one box per question containing "crosses", representing answers were not addressed either, since these are not technical but are policy matters of the examination process.

In digit recognition part, I must know the exact personal number, i.e. by who the quiz-form has been filled. Evidently, accuracy in this part is very vital, which is the main goal of my study.

The variation of the appearance of hand written digits is large, and to a certain extent it is unique for each person. Often, it is even difficult to recognize some hand written digits by a human. The variation may be caused by a long list of reasons, including different inks, pen-pressure, trace of erasing, experience of writing in Latin/Arabic numeral symbols (many university students have achieved their pre-university studies using other symbols for numerals), etc.

Contribution

I achieved to trace a road-map for a small digit recognition from gray images up to 80% recognition accuracy by studying implementations and combinations of the following:

-Image preprocessing including normalization

-Feature extraction based on the geometric appearance of numerals

-Recognition engines based on a small training set

I have studied Logistic regression [8, 13] and Artificial Neural Network (ANN) [2, 8, 9, 14, 15] algorithms as recognition engines for digit recognition. Although these techniques have been used before for digit recognition, the published studies concern mainly using them on binary images as features, leaving out how to preprocess image-data that has gray value pixels originally, which additional/replacement features to extract, which features to select[5, 7, 8], and which recognition engine to use. Accordingly the studies do not point on a feasible road-map for how to construct a small system that achieves the entire recognition chain of numeral recognition. On the other hand large general purpose commercial systems of OCR exist but they, apart from being unaffordable for small missions, need an adaption before they can be useful to quiz correction scenario, motivating this study.

Different selection of features from the training set examined to find the best feature subset for classification task.

I am not aware of any work in hand written digit recognition that has followed this thesis approach to construct a road-map for a small digit recognition solution.

Background

Recognition of hand written characters has been studied since the 1980s. Handwritten digit recognition, using a classifier, has major importance and use such as – online handwriting recognition on computer tablets, zip codes recognition on mail for postal mail sorting, cars registering number processing, numeric entries in forms filled up by human (for example - quiz forms) and so on. There are different challenges met while attempting to solve this issue. Handwritten digit recognition can mostly not get help from dictionaries to resolve recognition conflicts as handwritings can do. However, handwritten digit recognition can get some help from checksums if they are available in the application. In this application I did not assume the presence of this help.

Although many classification algorithms have been discussed in the past years, for example logistic regression [8, 13] , artificial neural network [2, 8, 14, 15] and SVM, handwriting digit recognition has always been a challenging task in pattern recognition.

According to [21], they planned a neural network technique for printed and handwritten digits recognition. They used a multilayer and clustered back propagation algorithm. Five independent sub networks were used for classifying the numerals. Minimizing the error between the reached output and the wanted output was the main aim of using multilayer neural network. They trained each subnet with different feature maps. They used another neural classifier, called the improvement network, which works on the forms that are refused by the first network.

Y.Le Cun[20], In 1990, designed a back propagation network for handwritten digits recognition in zip codes. The task here contains data acquisition, binarization, zip code location, initial separation and at the end, the normalization of the digits using linear transformation method. Multi-layer network is used for recognizing, and the network was trained with back propagation network. With their method the training set error was 3.4%.

In[23], they used Freeman chain code Representation and Feed forward Neural network classifier for handwritten Latin characters recognition.

For minimizing the length of the chain code, a randomized algorithm has been generated. Pre-processing, feature extraction and classification are included in this method. The skeleton of a character was achieved by thinning method in the preprocessing stage, in this process, any redundant information is also removed. A randomized algorithm was designed, in feature extraction, for minimizing the length of the chain code. And they used a neural network classifier for classification.

Convolutional Neural Network, CNN, a feed forward network, was used by [22] for handwritten digits recognition in 2009. Pixel images are used directly to recognize samples with minimum need of preprocessing.

In the first hidden layer, features are extracted from the input image, in the final hidden layer, patterns are classified. By use of CNN, information extraction from images needs least preprocessing. It is necessary to remove pixels that do not belong to a numeral from the background when using CNN to obtain good performance.

In their learning process, the white pixels are set to 1 and the background pixels are set to 0. After the learning process, they tested the network with original NIST dataset and obtained 96.74% accuracy.

Previous related works mostly used MNIST[18] data base [2, 17] which contain binary images of hand written digits including a training set of 60,000 examples and a testing set of 10,000 examples, while I used my own data base prepared from quiz paper images. MNIST is a subset of NIST, in this data base digits have been size normalized and centered in a fixed-size image. The images were centered in a 28x28 image.

2 Data preparation

I have numerous samples for each digit from which the machine should learn and later the machine should recognize an example of a digit that it has not seen before.

Humans see many samples of hand written digits in their lives and they learn from them to recognize a new sample. This is similar to machine learning but we have very little knowledge on how the detail of human recognition takes place. Though not reaching the same perfection levels, machines can also do recognition in images representing documents.

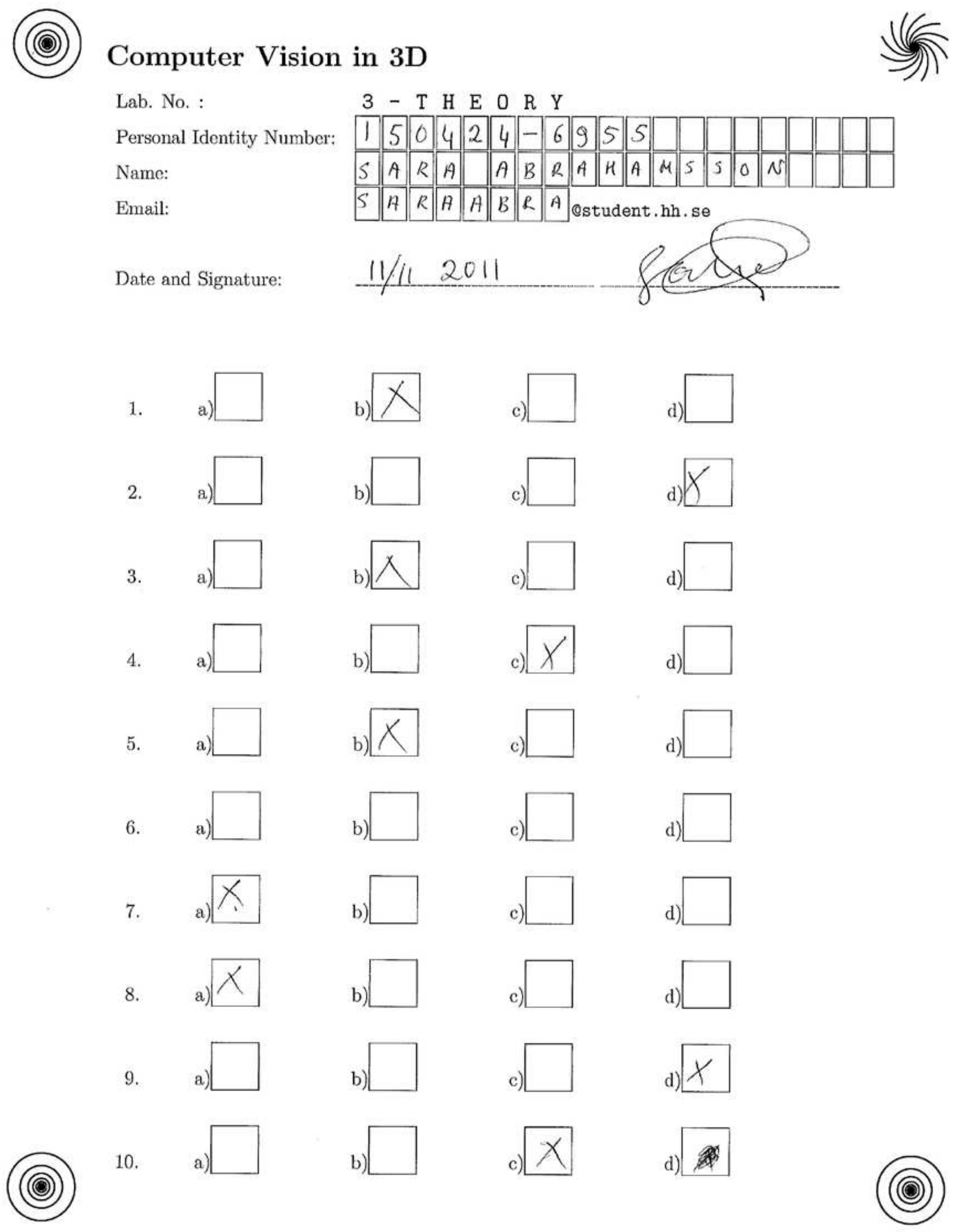
There is much published material about machine learning, in general, numeral recognition in particular.
Optical character recognition is one of the successful applications of pattern recognition [4, 7], and handwritten digit recognition is an active topic in OCR. There are different methods of classification for general purposes such as logistic regression or neural approaches. However, the performance of them on specific tasks may vary, one being not always the most performant.
For my purposes it was not evident which combination of methods would be the most performant, starting from gray-value images up-to recognized symbols. I had to first develop and define the feature set. This was necessary to recognize handwritten digits and improve recognition rate.
Consequently, I had to introduce novel data sets to measure and compare recognition performance of these methods.
Prior to feature extraction the images had to be prepared with the purpose that the extracted features are on the relevant parts of the image containing numerals.
Before describing Data preparation, below I summarize the data of quiz-correction scenario.

2.1 Data set description

By data set, I mean a collection of samples with which I want to teach a machine to learn them. In this project, I work with images containing numerals so I need to prepare a data set that can get a machine to recognize digits.

Here (fig.1) I presented one sample of data, an image in which numerals of "Personal Identity Number" field are to be recognized. All hand entered information is fictive, to protect personal information[1].

1. It is worth noting that I did not have access to quiz questions nor correct quiz answers to protect the privacy of the students represented in the data set further.



Computer Vision in 3D

Lab. No. :	3 - THEORY
Personal Identity Number:	150424-6955
Name:	SARA ABRAHAMSSON
Email:	SARAABRA@student.hh.se
Date and Signature:	11/11 2011 [illegible]

1.	a)	b) ✓	c)	d)
2.	a)	b)	c)	d) ✓
3.	a)	b) ✓	c)	d)
4.	a)	b)	c) ✓	d)
5.	a)	b) ✓	c)	d)
6.	a)	b)	c)	d)
7.	a) ✓	b)	c)	d)
8.	a) ✓	b)	c)	d)
9.	a)	b)	c)	d) ✓
10.	a)	b)	c) ✓	d) ■

Figure – 1: Quiz paper sample

A data set containing 2256 samples of numerals (0 through 9) has been obtained. They are contained in boxes, 10 of them per image, on the top, see Fig. 1. This data set should be separated to two parts, one for training and one for testing. The ratio 10% of dataset is often practiced as the share of a test set. Accordingly, I assigned 2000 examples as training set and 256 examples as testing set. I chose test data at random from the dataset.

I presented in below table that how many of each numeral I had in the training, test and total:

Numeral	Train	Test	Total
0	346	47	393
1	221	26	247
2	162	20	182
3	128	13	141
4	141	19	160
5	199	24	223
6	139	17	156
7	206	26	232
8	325	47	372
9	133	17	150
Total	**2000**	**256**	**2256**

2.2 Registration

Before starting to crop, the image of a quiz answer is registered (here rotation and scaling). This is done first by detection of the spirals, and then applying an appropriate geometric transformation so that the image can be rotated and scaled towards a common answer form. Accordingly, the positions of all rectangles can be assumed unchanged in registered images.

Image registration is thus the process of aligning two or more images so that after alignment one has a common coordinate system. By the way of example, in this case, the first rectangle of the social security number will occupy the same pixel coordinates in all registered images. The methods in this study are thus applied to registered images

2.3 Cropping

To populate the data set that concerns the social security number (Personnummer in Swedish), rectangles which include the social security number should be cropped from images.

Images of numerals are obtained by cutting predefined rectangles from the scanned quiz-forms containing them, e.g. Fig. 1, which shows the quiz form. The size of the rectangle in pixels depends on the scanning resolution as well as the physical size of the corresponding rectangle. For example, a 20x20 image will have 400 pixels where each pixel will have a gray-value. Each pixel has the same physical size. In a scanning with 300 pixels per inch resolution, (the case in my experiments) every pixel is a square having a side length of 25.4/300=0.085 millimeter, so that a 20x20 pixels area corresponds to 1.7x1.7 mm after registration (rotation and translation does not change distances).

Evidently, an image of a numeral is likely to be not the same each time it is produced by a human, even if it is the same numeral and produced by the same person, because such numerals are produced at different instants.

I cropped each rectangle separately such that I could omit the black lines representing the rectangle borders, threshold the gray-values such that pixel values became binary (binary images). Furthermore, I reshaped these images such that they were one row (scan-line ordered) and saved all images in a matrix.

As said, all images of documents are registered so the position of each rectangle recurs in the same place in all images after registration. After this, I cropped rectangles as 35x27 images and omitted borders by cropping them again to 35x22 images, thresholded them to binary images, and then reshaped them to one row, 1x770.

2.4 Binarization

Binarization is the step of mapping the value of every pixel to either 0 (background pixel) or 1 (object or foreground pixel). Binarization is needed because working on them is easier and result is better.

The binarization in this study has been done by comparing the gray-value to a fixed threshold, T. The value of T was defined empirically by studying the histogram of an image region containing all boxes with numerals and by guessing.

It can also be determined dynamically and automatically, e.g. by Otsu's method, which is even implemented in Matlab [19].

However, my method with global thresholding with a fixed threshold (same for all numerals) was used for saving computation time (simple system).

2.5 Objects

I need to extract properties of numerals such that they together uniquely can take apart one numeral from another. For this purpose I need to define the concept of "object" in a binary image (pixel values are either black or white).

An object is a set of pixels, which are connected together. Two pixels are connected if they are neighbors. Two pixels are neighbors if they touch another pixel, including at corners, i.e. I assume 8-connected neighbors, Fig. 2. If there are two objects in an image all properties will be computed for each of them separately.

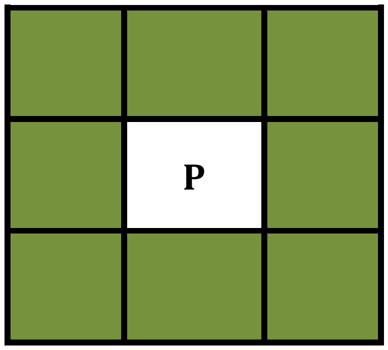


Figure – 2: 8 connected neighbors

After binarization I obtained "objects" with *bwconncomp()* which returns the connected components,

```
    I=bwconncomp(A);
    no_obj(i)=I.NumObjects;
```

Connected components are a set of pixels where every pixel is a neighbor of another, in the sense of 8-connected neighbors (Fig. 2). One of the fields of the variable I, mentioned above, is *NumObjects* which is number of object in input Image so number of object is obtained in all images, these are defined as follows.

2.5.1 Images with one object

An image containing a single numeral can have several objects, some as integral part of it and others produced as a consequence of poor image processing (e.g. inadequate thresholding), writing style, or poor pressure of the pen, etc. Fig. 3 shows images with one object.

Figure – 3: Images with only one object

2.5.2 Images with more than one object

There are some images that have more than one object in, here are some samples:

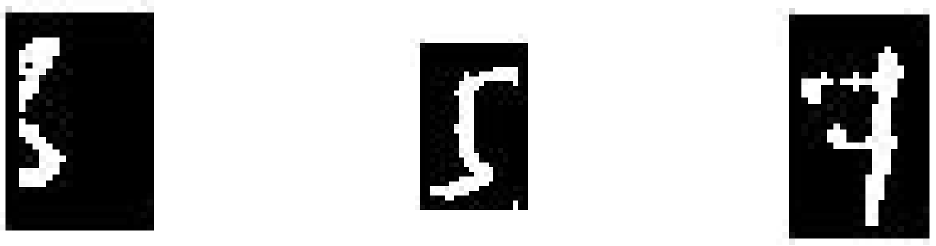


Figure – 4: Images with more than one object

These images can be classified to three groups.

- Images with more than one object that all objects are not a part of usable image:

Figure – 5: First group of images with more than one object

Above images have more than one object, and some of these objects are not a part of the numeral and deleting them will help a machine to learn the numerals better. There are different approaches to achieve this, which will be discussed later.

- Images that all objects are a part of usable image:

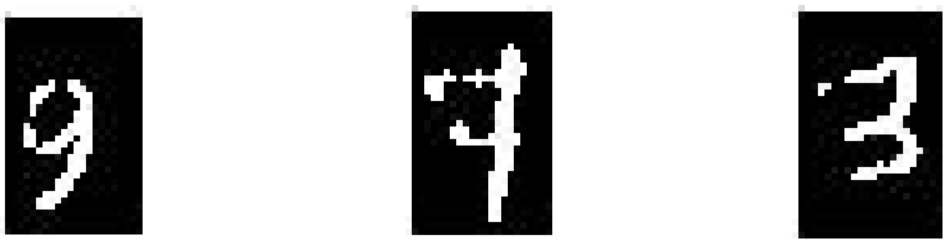


Figure – 6: Second group of images with more than one object

Fig. 6 shows images containing more than one object but all objects are actually integral parts of numerals. Deleting them by filtering "extra" objects will harm machine learning. Accordingly, for these kinds of images some ways should be found to treat them as one object, which will be explained later.

Fig. 7 shows images which contains numerals with missing parts or are unreadable

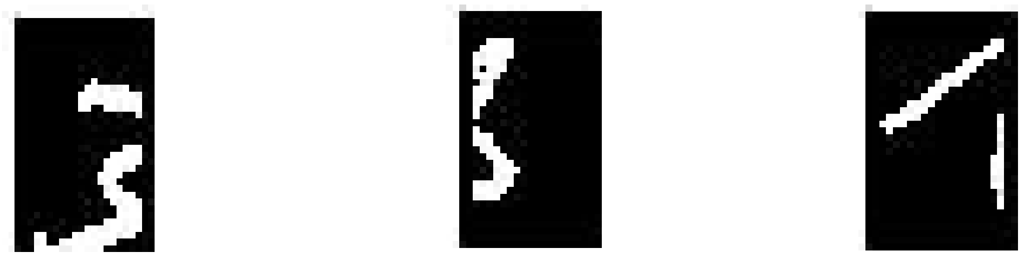


Figure – 7: Third group of images with more than one object

2.6 Filtering by image processing

To achieve better machine learning, I need to filter the binary images such that the resulting novel objects are easier to interpret as numerals by machines.

2.6.1 Mathematical Morphology

Using images with more than one object in training is harmful to machine learning.

There is a theory and technique called Mathematical Morphology (MM) [11, 16], which was originally developed for binary images which can be helpful to merge multiple objects to one, sensibly.

Erosion and dilation are the two most basic operations in MM; these two operators have two inputs; image which should be eroded or dilated and a structuring element.

The way of representing white and black pixels matters in this theory, in a binary image; foreground regions are represented normally by white pixels, while background is denoted by black pixels. Note that in current images this convention is reversed. Accordingly we must reverse the binary values of images before using standard implementations of MM operations.

Dilation

This operator enlarges the boundaries of regions of foreground pixels (*i.e.* white pixels, typically). Thus, areas of foreground pixels grow in size while holes within those regions become smaller. Therefore, by using this operator, useful objects can become one object. Here is an example:

Before dilation After dilation

Figure – 8: Dilation (a)

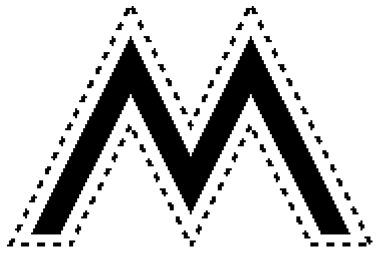
Below is another example with 3 objects in an image and dilation effect on it.

Before dilation (3objects) After dilation (1object)

Figure – 9: Dilation (b)

The dilation function has two inputs. The first is the image, which is to be dilated. The second is a (usually small) set of coordinate points known as a structuring element. This structuring element determines the precise effect of the dilation on the input image. This element is defined by *strel()* function that has different parameters, e.g. square, rectangle, diamond and disk.

Erosion

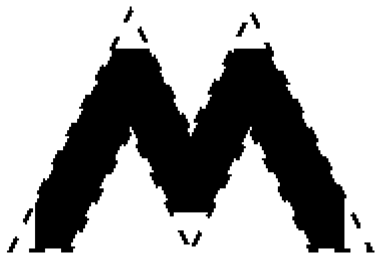


Erosion is another operator in the area of mathematical morphology. It is applied to binary images, but there are versions that work on grayscale images. The effect of this operator on a binary image is to erode away the boundaries of regions of foreground pixels. Thus, areas of foreground pixels shrink in size, and holes within those areas become larger.

Erosion is thereby the dual of dilation, and dilating foreground pixels is equal to eroding the background pixels.

Virtually all other mathematical morphology operators can be defined in terms of combinations of erosion and dilation. Some of the more important are opening, closing.

Opening



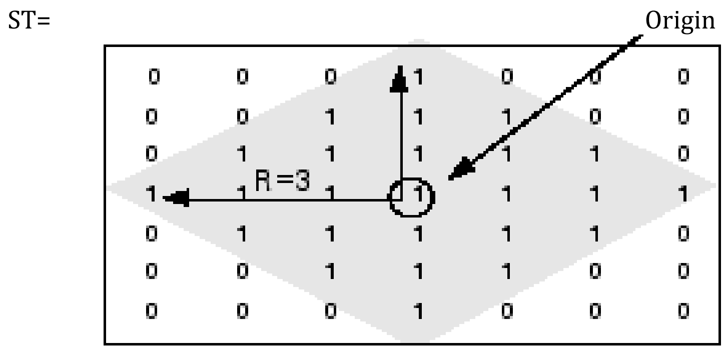
Opening is defined as *dilate (erode (IM))*, an erosion operation followed by a dilation operation, where IM is the binary image. It makes object boundaries more round while it deletes small objects with size smaller than the structuring element.

Closing

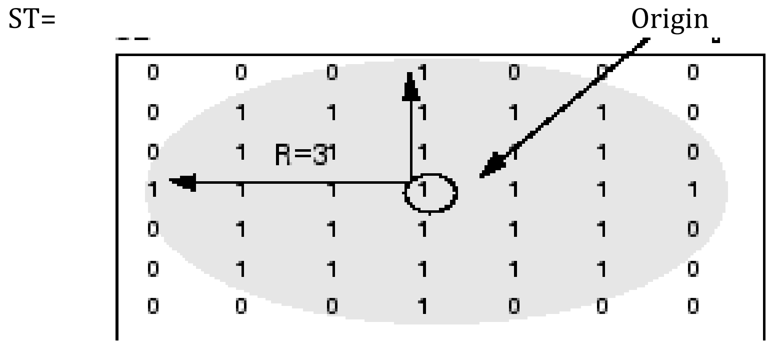
Closing and dilation are similar in some ways; closing tries to enlarge the boundaries of foreground (bright) regions in an image (and shrink background color holes in such regions). Closing is defined as *erode (dilate (IM)),* a dilation operation followed by an erosion operation.

As said before, morphological operators take two pieces of data as input; Image, which may be either binary or grayscale. The other is the *structuring element*. It is this that determines the precise details of the effect of the operator on the image. Function ***strel(shape, parameters)*** creates a structuring element of the type specified by shape. Depending on shape, *strel()* can also take additional parameters. Rectangle, Square, Diamond and Disk has been used in my implementation as parameter *shape*. Depending on these shapes *parameters* can be defined. Here are some examples:

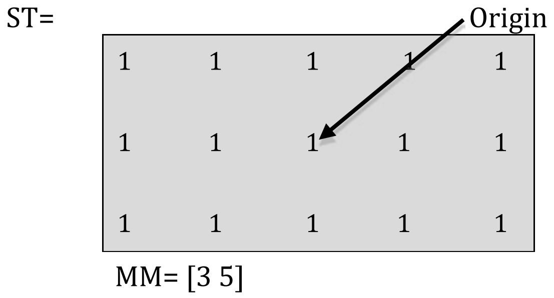
ST= strel('diamond', R) generates a flat, diamond-shaped structuring element, where R specifies the distance from the structuring element origin to the points of the diamond. R must be a nonnegative integer.



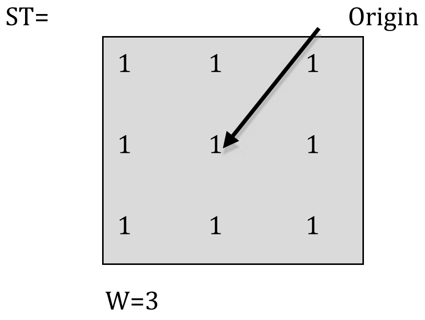
ST = strel('disk', R, N) generates a flat, disk-shaped structuring element,
where R refers to the radius of disk. R must be a nonnegative integer. I used default value of N which is 4.



ST = strel('rectangle', MM) produce a flat, rectangle-shaped structuring element, which MM refers to the size. MM must be a two-element vector of nonnegative integers. MM is the number of rows in the structuring element neighborhood; the second element is the number of columns.



ST = strel('square', W) creates a square structuring element whose width



is W pixels. W must be a nonnegative integer.

In the experiments different values used for different shapes, they have been explained in Result part.

2.6.2 Cleaning objects by Mathematical Morphology

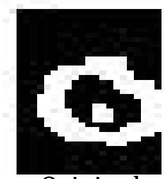
By studying the first group of images with more than one object where all objects is not integral parts of a numeral (usable image); I found out that such objects have not large areas. If one deletes the objects with area smaller than some "small" threshold (area), one can clean images from small objects.

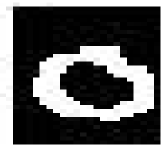
To this effect I used Bwareaopen() in Matlab image processing toolbox [11]. It implements the morphological operator opening. Here are some example images that I obtained when I cleaned images by deleting small objects:

Figure – 10: Images before cleaning

Figure – 11: Images after cleaning

An alternative method is using another morphological filter (implemented by Matlab function bwmorph()). However, it did not yield as good results as bwareaopen(), since it removes isolated pixels, as shown in Fig.12.


Original


Using bwareaopen()

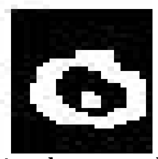

Using bwmorph()

Figure -12: Comparing bwareaopen() and bwmorph()

Accordingly I chose area opening to reduce the learning problems caused by this group of images.

3 Feature extraction

Some useful specifications/measurements should be extracted from each image automatically and assign them with the identities of the respective numerals, the training. If outputs in the data set are discrete, then the task is called classification like the current project. Else, the procedure is called regression.

3.1 features selected from geometric properties

In my first attempt, I used the pixel values as features. I even tested alternative features that I extracted by using image processing. These are described here next.

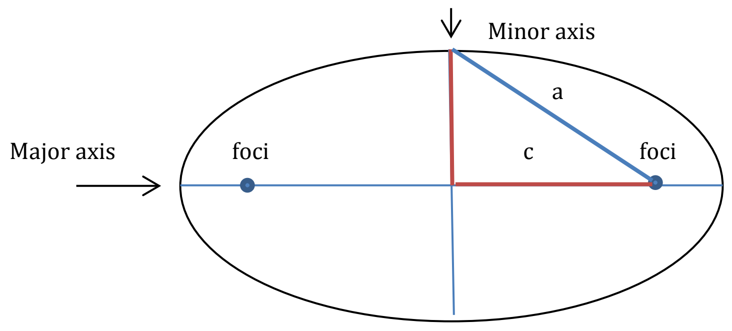
Some unique properties should be used as features instead of or as complement to pixel values. I used below features on their own (without pixel values). These features were extracted from each image, and then stored in a row. However, different combinations of features were selected to find the best result which will be reported in the Result part. Below I summarize the plausible features.

Area

Area is a scalar value that is equal to the number of pixels in the region representing the object, here the area of a numeral in pixels.
There is function implemented in Matlab called *bwarea()* , which estimates the area of the objects in binary image.

Eccentricity

Eccentricity of an object is equal to the eccentricity of the ellipse with the same second-moments as the object. Eccentricity is the ratio of the distance from the center to a focus and the distance from that focus to a vertex. The value is between 0 and 1.



eccentricity=c/a

EquivDiameter

Is a scalar value corresponds to the diameter of a circle with the same area as the region. Computed as sqrt(4*Area/pi).

Euler Number

Is a scalar value that specifies the number of objects in the region minus the number of holes in those objects.

Extent

Scalar value that computed the ratio of pixels in the region to pixels in the total bounding box. Computed like this: Area divided by the area of the bounding box.

Filled Area

Number of on pixels in Filled Image.

MajorAxisLength

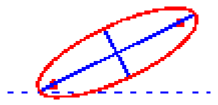
Scalar specifying the length (in pixels) of the major axis of the ellipse that has the same normalized second central moments as the region.

MinorAxisLength

The length (in pixels) of the minor axis of the ellipse that has the same normalized second central moments as the region.

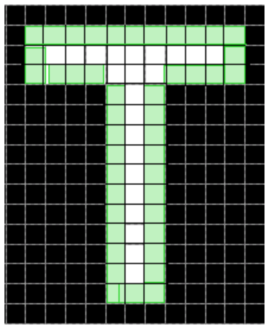
Orientation

Orientation of an object refers to the angle (in degrees) between the x-axis and the major axis of the ellipse that has the same second-moments as the object.



The angle between the horizontal dotted line and the major axis is orientation.

Perimeter
This is the number of pixels at the boundary of an object. When holes are outlined, such as zero, the perimeters of the holes are added to the perimeter of the object.



The above figure shows the pixels included in the perimeter computation for this object.

Solidity

Solidity is area proportion of the object as compared to its convex hull.

Convex area

The Convex hull of a set of object pixels mimics the mathematical definition of the convex hull; the line joining any two boundary point of the hull is completely inside the hull. The number of pixels inside the convex-hull is the convex area.

Now there are some new properties to use as features.

4 Recognition engine

Next step after data preparation is introducing recognition engine, which could learn from this data set and recognize new samples not included in the training data set. As said before this task is a classification. Logistic regression, neural network and Matlab classify function will be discussed as recognition engine.

4.1 Feature normalization

In learning algorithms, it is important that values of the features are in a suitable range for learning, although there is not any specific range that works for all data sets [8]. For example if x_1 is one dimension of a feature vector, it may be between [-1 1]. This is then normalized using the mean value and some sort of range estimation of x_1 as follows.

$$x_{1m} = \frac{x_1 - \mu_1}{s_1}$$

Here x_1 is any value of feature x_1 in the data set, μ_1 is average value of all values of x_1 and s_1 is the range of values of x_1, which could mean (max value – min value) as well as an estimation of standard deviation. I used standard deviation. The quantities μ_1 and s_1 were computed by Matlab functions ***mean()*** and ***std()***[11].

As said before, there is training data and testing data. However, I cannot normalize testing data on its own, as it is an operational situation in which there may not be more than a single sample available. It means that μ and s, which has been computed on training data, should also be used for normalizing testing data.

4.2 Principal Component Analysis (PCA)[1,8,10,12]

With high number of variables, classification becomes more difficult because when the dimensionality increases then much more training data is needed, the amount of data to support a trusty result often grows exponentially with the increase of dimensions. There are different techniques for dimension reduction; PCA is one of these techniques.

PCA is a method to reduce number of features in dataset with high number of features. Assume that $\boldsymbol{x}^{(i)}$ has been mean normalized i.e.

$$\boldsymbol{x}^{(i)} \leftarrow \boldsymbol{x}^{(i)} - \boldsymbol{mu}$$

To reduce from n-dimension to k-dimensions I have proceeded as follows:

First covariance matrix called **Sigma (∑)** should be computed:

$$\sum = 1/m \sum_{i=1}^{n} (\boldsymbol{x}^{(i)})(\boldsymbol{x}^{(i)})^{\mathbf{T}}$$

Here (m) is number of my samples. For data set with same format of mine, (each row is a sample) sigma is computed like this:

$$\mathbf{Sigma} = (1/m) * \mathbf{X}^{\mathbf{T}} * \mathbf{X}$$

Then I use eig() (eigen value decomposition) of the auto-correlation matrix, which is a way to obtain the eigenvectors (in **U**) and eigenvalues (in **S**) as follows:

[U,S] = eig**(Sigma)**

Alternatively I could use Svd decomposition directly on **X**, resulting in the same eigenvectors (but square root of eigenvalues).

U is *nxn* matrix of which I need first *k* column of this matrix. Computing of *k* will be described later. New matrix is called **Upca**

$$\mathbf{Upca} = \mathbf{U}(:, 1:k)$$

Upca is used to compute new data set with reduced features:

$$\mathbf{Xpca} = (\mathbf{Upca}^{\mathbf{T}} * \mathbf{X}^{\mathbf{T}})^{\mathbf{T}}$$

The way of computing of *k*:

For computing (k), the (diagonal) eigenvalue matrix **S** is used which was returned by **eig(),** in below equation:

$$\frac{\sum_{i=1}^{k} s_{ii}}{\sum_{i=1}^{m} s_{ii}} \geq 0.99$$

Smallest value of (k) is picked up when the above equation is true. I mean, I start with (k=*1*) and continue until find the smallest (k) so above equation is true.

It is very important that data have been normalized before using PCA. Suppose that one of the variable (for example $\mathbf{x_1}$) has values in range 0 - 100 while others have between 0.1 – 1, this will distort the axis of highest variance towards $\mathbf{x_1}$'s axis very much. In PCA it is very important to make different attributes more comparable, to achieve this goal, normalization which has been discussed before, has been used.

4.3 Logistic regression[8,13]

In very short, this machine learning algorithm can explained as follows. There are some data samples, which belong to a certain class; every class has a set of samples, which represents the class in the training set. In fact the data set is the collection of all such samples, consisting of feature vectors, whose class-identities (ten numerals) are known.

In logistic regression, I work with the functions:

$$\mathbf{h_\theta(x) = g(\theta^T x)}$$

$$\boldsymbol{g}(z) = \frac{1}{1 + e^{-z}}$$

The vector **x** is the feature vector representing a rectangle of 20x20 containing a numeral. The **θ** is a parameter vector, which the recognition engine (logistic regression) should compute, and ***g(z)*** is sigmoid function that approaches to 1 for large positive values of ***z*** and for large negative values, it approaches to 0.

Each time that machine learns a numeral from training data; a **θ** vector specific to that numeral is computed. I have saved such vectors, and used them on test data. As a decision, I chose the numeral whose **θ** vector gave the highest value of $\mathbf{g(\theta^T x)}$**.** Test accuracy is the portion of correct decisions using test samples.

Train accuracy is computed by using the training samples to calculate the portion of correct decisions.

4.3.1 Experiments and results

The feature vector used is the binarized image. The binary images of numerals were vectorized such that each feature vector is obtained as a traversal of the image (binary) matrix, from top-to-down, and then left-to-right. As first trial, the engine designed by logistic regression approach is used, the details of experiments are:

Data set: 2256x770

Train data: 2000x770 (rows 1:1500 and 1757:2256)

Test data: 256x770 (rows 1501:1756)

Train accuracy: 86%

Test accuracy: 39%

As described before the ratio 10% of dataset is often practiced as the share of a test set. The engine is learned by training data, using computed parameters to predict output of test data.

Training speed is slow because of the dimension of feature vectors, which is 770; this high dimension is not a good option. For efficient regression, there should be ideally at least 10 times more data than the number of unknown parameters. Our data has the dimension 770, so that we should have at least 7700 different images per numeral and 77000 numerals in total in the training set. There is another point also for describing this high dimension as disadvantage. By looking at hand written numerals I found out many of the pixels which I use as features are not usable for training because these were stray pixels that can be called as noise. Accordingly, additional processing should be applied to reduce the dimension and also extract features which are reasonable for the engine to learn.

I tried resizing images to smaller images, but if images were resized to smaller squares since they are strict rectangles, original images of numeral rectangles would be rescaled differently in horizontal and vertical direction. Therefore, I added some white pixels to original image rectangles, and then rescaled them to a square of 20x20 image rectangles. The results were as follows.

Dataset: 2256x400

Train data: 2000x400 (rows 1:1500 and 1757:2256)

Test data: 256x400 (rows 1501:1756)

Train accuracy: 64%

Test accuracy: 47%

The result was improved but 400 variables appeared still too large of a dimension. Images which I used as features were binary. Using the original gray scale images did not change the results significantly.

Feature normalization applied on same data and result is:

Train accuracy: 72%

Test accuracy: 43%

Applying PCA on same data:

Number of features after reduction: 166

Train accuracy: 65%

Test accuracy: 46%

These results suggest that feature extraction and/or classification need to be improved. One may attempt to improve on both ends.

4.3.1.1 Some alternative features

These features were extracted from images with only one object and a data set was specified with them:

Dataset: 1685x9
Train data: 1486x9 (rows 1:600 800:end)
Test data: 199x9 (rows 601:799)

Features:

Area

Extent

MajorAxisLength

MinorAxisLength

Orientation

Perimeter

Solidity

Eccentricity

Filledarea

I used logistic regression engine and the result was:

Train accuracy: 62%

Test accuracy: 62%

4.3.1.2 Mathematical Morphology

Mathematical Morphology operators were used on all images to achieve images with only one object. I used 11 features comprising the 9 features above (Area,..., Filledarea) and EquivDiameter and EulerNumber.

Structuring element:

```
    se = strel('rectangle', [1 5]);
    A=imdilate(A,se);
```

Dataset: 2010x11
Train data: 1811x11 (rows 1:600 800:end)
Test data: 199x11 (rows 601:799)

Result:

Train: 58%

Test: 59%

Structuring element:

```
    se = strel('square', 4);
    A=imdilate(A,se);
```

Dataset: 2103x11
Train data: 1904x11 (rows 1:600 800:end)
Test data: 199x11 (rows 601:799)

Result:

Train: 56%

Test: 59%

4.4 Neural network[2,8,9,14,15]

Artificial Neural Network is an advanced algorithm for classification tasks with high training time and large data sets.

The neural network is configured by combination of neuron which can be presented by a logic unit with activation function "F". Suppose that there are n+1 input as i_0 to i_n and n+1 weights as w_0 to w_n where i_0 = 1 and w_0 = "bias" as shown in fig.13

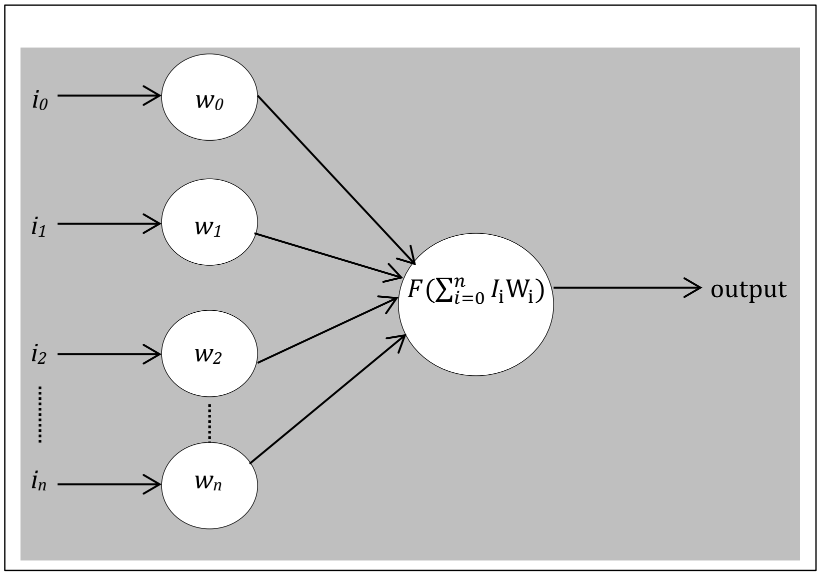


Figure 13: Artificial neuron

Function "F" converts the sum of multiplication of inputs by weights ($i_0w_0 + i_1w_1 + \ldots + i_nw_n$) to neuron output.

The first layer is input layer, last layer is output layer and layers between them called hidden layer. There are different kinds of artificial neural network (ANN) architectures. Feed forward is the one which I used in this project and in this architecture there is not any backward connection to previous layer. A simple feed forward network is shown in fig.14 with 3 nodes in input layer, one hidden layer with two nodes and one node in output layer.

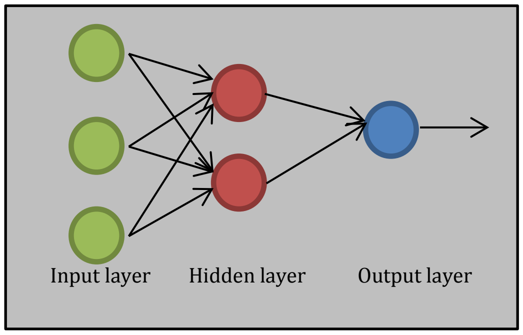


Figure 14: Feed forward network

A point should be mentioned about using Neural Network. In this classifier my output configuration is more complex. I have a 10 class classification (labeled as 0..9), so that my output has a column with 10 nodes (output layer). The values of the nodes can be either 0 or 1 but 1 should only occur at the correct node. For example if the data of the numeral presented at the input layer is the one for "2" then the output should look like this:

[0 1 0 0 0 0 0 0 0 0]

The training data set has been specified accordingly when training the neural network. I have then obtained a vector of 10 elements as output each time I introduced the numeral data at the input.

Selecting an optimal number of hidden layers and nodes is normally done by experimenting with the training data and it influences the classification error. I changed them empirically until classification error did not show a significant decrease.

4.4.1 Experiments and results

I created a neural network with one hidden layer on the dataset:

Features: pixels values

Dataset: 2256x400

Train data: 2000x400 (rows 1:1500 and 1757:2256)

Test data: 256x400 (rows 1501:1756)

Test accuracy: 35% with 3 hidden layers

4.4.1.1 Alternative features

These features were extracted from image with only one object (without using Mathematical Morphology operators);

Area

Extent

MajorAxisLength

MinorAxisLength

Orientation

Perimeter

Dataset: 1685x6

Train data: 1486x6 (rows 1:600 800:end)

Test data: 199x6 (rows 601:799)

Hidden layer: 45

Test accuracy with Neural network 58%

These two properties added as new features:

Solidity

Eccentricity

Dataset: 1685x8

Train data: 1486x8 (rows 1:600 800:end)

Test data: 199x8 (rows 601:799)

Hidden layer: 55

Result: 60%

This result is improved, so these features are good selections.

Another property which is added is:

Filledarea

Hidden layer: 45

Result: 67%

I also used the following 3 properties additionally:

EquivDiameter

ConvexArea

EulerNumber

Dataset: 1685x12

Train data: 1486x12 (rows 1:600 800:end)

Test data: 199x12 (rows 601:799)

Hidden layer: 65

Result: 70%

12 features works well for Neural network so I can keep it for this classifier.

Now I checked NN with omitting *ConvexArea* and adding feature ***area,*** result became better:

Hidden layer: 65

Test accuracy:74%

This is the best result so far.

Now NN with property *ConvexArea* as feature:

Test accuracy:70% with 65 nodes

So adding *ConvexArea* had not good effect.

4.4.1.2 Mathematical Morphology

I used Mathematical Morphology operators to change images with more than one object to images with one object. Below is the best result I obtained:

11 features (Area, Extent, MajorAxisLength, MinorAxisLength, Orientation, Perimeter, Solidity, Eccentricity, Filledarea , EquivDiameter, ConvexArea):

Dataset: 2010x11

Train data: 1811x11   (rows 1:600 800:end)

Test data: 199x11   (rows 601:799)

With this change A=*bwareaopen*(A,15);

Erosion

Structuring element:

se = strel('rectangle',[1 5]);

A=imerode(A,se);

Hidden layer: 30

Result is: 54%

se = strel('square', 4);

Hidden layer: 35

Result: 54%

4.4.1.2.1 MM on All images

Till now I used filters on only images with more than one object in them now I use operators on all images. Below I report only the best results I obtained when changing the structuring element and the number of hidden layers.

Dilation

With 12 features (Area, Extent, MajorAxisLength, MinorAxisLength, Orientation, Perimeter, Solidity, Eccentricity, Filledarea , EquivDiameter, ConvexArea, EulerNumber)

Dataset: 2021x12

Train data: 1822x12 (rows 1:600 800: end)

Test data: 199x12 (rows 601:799)

se = strel('square',2);

Result is: 76%   with 55 hidden layers

I added feature bwarea

Result: 75% with 45 hidden layers

Using bwmorph, clean

Result: 77%, 45 hidden layers with 12 features

With 13 features (Area, Extent, MajorAxisLength, MinorAxisLength, Orientation, Perimeter, Solidity, Eccentricity, Filledarea , EquivDiameter, ConvexArea, EulerNumber, bwarea)

Dataset: 2021x13

Train data: 1822x13 (rows 1:600 800: end)

Test data: 199x13 (rows 601:799)

se = strel('square',2);

Result: 79%, with 45 hidden layers.

This is the best result so far.

Closing

se = strel('disk', 3);

Result: 71% with 55 hidden layers

4.5 Classify function[11]

There is also a function called ***Classify()***

output=classify(TestInput,TrainInput,TrainOutput,'Type');

This function classifies each row of the data in 'TestInput' into one of the 'TrainOutput' by help of 'TrainInput'. Consequently 'TestInput' and 'TrainInput' must be matrixes with the same number of columns.

I used Matlab which already have this function to check its result on my data, it is possible to specify *type* in this function which *type* is one of the: Linear, Diaglinear, Quadratic, Diagquadratic, Mahalanobis.

The result of using them will be shown in result part.

4.5.1 Experiments and results

I used ***classify()*** on current data set with *diaglinear* classification type and the result is:

Features: pixels values

Dataset: 2256x400

Train data: 2000x400 (rows 1:1500 and 1757:2256)

Test data: 256x400 (rows 1501:1756)

Test accuracy: 39%

4.5.1.1 Alternative features

Extracting new properties as features has been discussed before so now I need to prepare a new data set with these new features.

As first trial, all images with only one object in them are found and these properties are extracted as features:

Area

Extent

MajorAxisLength

MinorAxisLength

Orientation

Perimeter

After extracting features I separated dataset to train part and test part:

Dataset: 1685x6

Train data: 1486x6 (rows 1:600 800:end)

Test data: 199x6 (rows 601:799)

By using *classify()* function with different type, results are:

Linear: 46%

Diaglinear: 44%

Quadratic: 54%

Digquadratic: 43%

Mahalanobis: 51%

Some new features will be added to see if they improve the result.

These two properties added as new features:

Solidity

Eccentricity

Run again with function classify:

Linear:53%

Diaglinear: 46%

Quadratic: 57%

Digquadratic: 43%

Mahalanobis: 62%

Comparing with last result, improvement can be seen.

Another property which is added is:

Filledarea

Run classify:

Linear:57%

Diaglinear: 46%

Quadratic: 58%

Digquadratic: 46%

Mahalanobis: 66%

I also used 3 more properties:

EquivDiameter

ConvexArea

EulerNumber

In total there were 12 features, (Area, Extent, MajorAxisLength, MinorAxisLength, Orientation, Perimeter, Solidity, Eccentricity, Filledarea , EquivDiameter, ConvexArea, EulerNumber)

Dataset: 1685x12

Train data: 1486x12   (rows 1:600 800:end)

Test data: 199x12   (rows 601:799)

Run *classify*():

Linear:55%

Diaglinear: 48%

Quadratic: 57%

Digquadratic: 48%

Mahalanobis: 64%

This time seems that it does not like changes, I omitted *ConexArea* and result became better so it seems that this property is not suitable as features.

There is another property which is unused yet, ***area*** which is returned by function ***bwarea()***, I add it as a feature to dataset and run *classify()* function.

As I can conclude from the result till now type ***mahalanobis*** works fine for this function so I used this only from now. Result by these 12 features by this function is:

Test accuracy: 68%

Result is better and by looking on all result this accuracy is highest till now.

Best result till now:

68% by classify function

4.5.1.2 Mathematical Morphology

Images with only one object in them has been worked with, now images with more than one object, different groups of these images and different kind of filters has been discussed before, below is different trials with short description of them:

4.5.1.2.1 MM on Images with more than one object

Use Mathematical Morphology operators to change images with more than one object to images with one object:

Dilation

Structuring element:

se = strel('rectangle',[1 5]);

A=imdilate(A,se);

Then 11 features except euler were extracted;

Dataset: 2010x11

Train data: 1811x11 (rows 1:600 800:end)

Test data: 199x11 (rows 601:799)

Test accuracy = 65%.

With this change A=*bwareaopen*(A,15);

Test :65% and 2010 image with one object

Erosion

Structuring element:

se = strel('diamond',3);

A=imerode(A,se);

Test: 63%, images: 1942

Opening

se = strel('diamond',3);

Test: 63% images:1942

Closing

Structuring element:

se = strel('diamond',3);

    A=imclose(A,se);

Test:64% images:2107

4.5.1.2.2 MM on All images

Till now I used filters on only images with more than one object in them now I use operators on all images,

Dilation

Dataset: 2021x12

Train data: 1822x12 (rows 1:600 800:end)

Test data: 199x12 (rows 601:799)

se = strel('square',2);

test:67% with 12 feature

Below I summarized all of the results:

Classifier	**Feature**	**Mathematical Morphology**	**Test accuracy**
Logistic regression	Pixel value	Without use of MM	47%
	Geometric property	Without use of MM	62%
		By use of MM (dilation,square,4)	59%
Neural Network	Pixel value	Without use of MM	35%
	Geometric property	Without use of MM	74%
		By use of MM (dilation,square,2)	**79%**
Classify function (type Mahalanobis)	Pixel value	Without use of MM	39%
	Geometric property	Without use of MM	68%
		By use of MM (dilation,square,2)	67%

5 Conclusion and Future work

When I want to start a machine-learning task, it is more important to focus on preparing data than selecting the method of learning, because even if I have an advanced engine but the train data is not prepared carefully, that engine cannot do its job correctly.

Among different features extracted for classification problem, the geometric properties of images was found to be the best feature subset extracted from data for classification, looking directly on the images in data set is useful to find a reasonably good set of features. This allows one to see whether or not there is sufficient discriminative information after having applied image processing with the purpose of classification improvement. I was helped by this strategy during this project several times to find a reasonably good set of features. Feature selection should be done very carefully. A high number of features is perhaps more descriptive (i.e. it can reproduce the numeral well) but it does not guarantee a good discrimination. I observed this many times when I had 770 pixel values as features. The recognition result was lower than with12 image properties as features in addition to that the speed of training was much longer (than 12 properties).

A data set with 2256 image samples has been obtained. I separated this to two parts, one for training and one for testing. The ratio 10% of dataset is often practiced as the share of a test set. This is approximately what I did when I chose test data at random from the dataset.

My best result was close to 80%. The configuration is given below for convenience again. I think it is difficult to improve it when we have so few data to train on (approximately 150 samples per numeral) and with the features I used. Further improvements cannot be ruled out if other more sophisticated shape features are used using more image processing, e.g. Fourier descriptors.

Also, if there is only one object per test image, we obtain systematically better results. This is not surprising because the recognition engine does not have to learn noise all the while and testing image does not offer difficulties in terms of noise.

To guarantee that automatically, one can choose the object that has largest area deleting others, if the image processing results in more than one object. I did not try this automatic strategy, but the results would be comparable to those I reported. This is because I manually assured that there was one object both in the training and in the test. In other words, what I did manually for quick implementation can be automatized.

Conclusion

It is very important that each digit has been written completely inside the rectangle, so when image has cropped I do not miss any information about image. When attempting to remove noise, some useful data may also be lost. Test data should be prepared exactly in the same way as train data.

Since the number which should be recognized is Swedish personal number, so its unique format can be used for recognition improvement, for example, in first field of month, 0 and 1 are only expected, and also by having the list of enrolled students one can add some complementary process to improve the recognition.

Best result achieved by this configuration:

Dataset: 2021x13

Train data: 1822x13 (rows 1:600 800:end)

Test data: 199x13 (rows 601:799)

Classifier: Neural Network

Number of nodes: 45

Features: 13 feature

'area','filledarea','eulernumber','convexarea','majoraxislength','minoraxisle

ngth','equivDiameter','extent','orientation','perimeter','solidity',

‘eccentricity’,’bwarea’.

Mathematical Morphology operator: Dilation

Structuring element: Square with parameter 2

Use *bwareaopen(A,15)* and *bwmorph(A,clean)*

Test accuracy: 79%

Train accuracy: 77%

6 References

[1] Bigun, J., (2006). *Vision with Direction,* Springer-Verlag Berlin Heidelberg

[2] Russell, S. and Norvig, P; (2003). *Artificial Intelligence: A Modern Approach 2nd edition*, Prentice Hall

[3] T. Hastie, R. Tibshirani, J. H. Friedman; (2009). *The Elements of Statistical Learning:Data Mining, Inference, and Prediction (Springer Series in Statistics), 2ndEdition*, Springer-Verlag, New York.

[4] R. O. Duda, P. E. Hart, D. G. Stork; (2001). *Pattern Classification, 2nd Edition*, John Wiley & Sons, New York.

[5] O. D. Trier, A. K. Jain, T. Taxt; (1996). *Feature Extraction Methods for Character Recognition- A Survey,Pattern Recognition, Vol. 29, No.4, pp.641-662*.

[6] C.-L. Liu, K. Nakashima, H. Sako, H. Fujisawa; (2002). *Handwritten Digit Recognition Using State of the art Techniques,* Proceedings of 8th International Workshop on Frontiers in Handwriting Recognition (IWFHR), Canada.

[7] Bishop, Christopher (2006). *Pattern Recognition and Machine Learning*. Berlin: Springer.

[8] Machine learning online course by Professor Andrew NG, Stanford University, https://www.coursera.org/course/ml

[9] Egmont-Petersen, M., de Ridder, D., Handels, H. (2002). *Image processing with neural networks - a review.* Pattern Recognition Society. Published by Elsevier Science Ltd.

[10] Abdi. H., & Williams, L.J. (2010). *Principal component analysis*. Wiley Interdisciplinary Reviews.

[11] Matlab documentation

[12] Jolliffe, I.T. (2002). *Principal Component Analysis,* second edition (Springer).

[13] Hilbe, Joseph M. (2009). *Logistic Regression Models.* Chapman & Hall/CRC Press.

[14] Bishop, C.M. (1995) *Neural Networks for Pattern Recognition*, Oxford University Press.

[15] Ripley, Brian D. (1996) *Pattern Recognition and Neural Networks*, Cambridge University Press

[16] Gerald J.F. Banon, Junior Barrera, Ulisses M. Braga-Neto (2007). *Mathematical Morphology and its Applications to Signal and Image Processing*. proceedings of the 8th international symposium on mathematical morphology.

[17] Mike O'Neill, *Neural Network for Recognition of Handwritten Digits*, (2006). Code project.

[18] Database at http://yann.lecun.com/exdb/mnist/

[19] Documentation center at http://www.mathworks.se/help/images/ref/graythresh.html

[20]) Y.Le Cuu, B.Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard and L.D. Jackel,(1990)*" Handwritten Digit Recognition with a Back-Propagation Network,"* AT&T Bell Laboratories, Holmdel.

[21] Daniel Cruces Alvarez, Fernando Martin Rodriguez, Xulio Fernandez Hermida,(1998)*"Printed and Handwritten Digits Recognition Using Neural Networks"*, E.T.S.I.T. Ciudad Universitaria S/N.36200 Vigo.SPAIN.

[22] Calin Enachescu, Cristian-Dumitru Miron,(2009)*"Handwritten Digits Recognition Using Neural Computing"*, Scientific Bulletin of the Petru Maior University of Tirgu Mures, Vol.6(XXIII), ISSN 1841-9267.

[23] Dewi Nasien, Siti S. Yuhaniz, Habibollah Haron,(2010)*" Recognition of Isolated Handwritten Latin Characters Using One Continuous Route of Freeman Chain Code Representation and Feedforward Neural Network Classifier"*, World Academy of Science, Engineering and Technology 67