Prof. Dr. Hauke Schramm
Department of Computer Science
Kiel University

Pattern Recognition

Exercises

Practice Sheet 2

Exercise L-2.1 (Analysis of hand written digits)

In this exercise we will analyze images of hand written digits, given in the files `digit[0,1,…,9].mat`. The files may be downloaded from the OLAT material folder:

```
/MaterialFolder/Laboratory/Sheet2/digit[0,…,9].mat
```

After you have loaded a file with the command

```
load("digit0.mat");
```

a matrix `D` of size 980 x 784 (for digit 0) will be available in the Octave workspace.

The matrix serves as a container for all images of a digit, which are stored as follows: Each row of the matrix contains a complete image of size 28 x 28 pixels, with 784 gray values in the range from 0 (black) to 255 (white). This is achieved by sequentially storing the individual rows of an image into a row vector as illustrated in Figure 1.

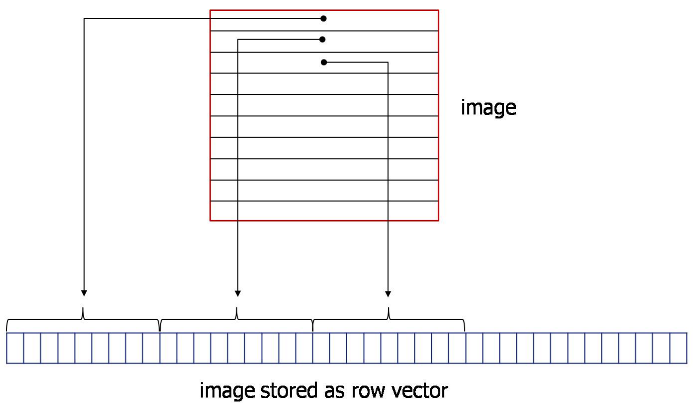


Figure 1. Illustration: Image stored as a row vector.

The following Octave script demonstrates how the images of digit 3 are loaded and displayed:

```
load "digit3.mat"
for i = 1:size(D,1)
      I = D(i,:);
      I = reshape(I, [28,28]); % converts I to size to 28 x 28
      figure(1), imshow(I,[]);
      pause(0.1);
end
```

You will find out that the displayed images are flipped and rotated which can be corrected using the commands `imrotate` and `fliplr` or by using the transpose function A'.

a. Write a function `show_data(im_num)` with one input parameter that uses two `for`-loops to display `im_num` example images for each digit. Use the command `pause` to wait before showing the next digit.

   Example: Calling `show_data (4)` displays 4 example images for each of the 10 digits.

   Hint: To build the filename for a specific digit you may use string concatenation (cf. Octave documentation) and the command `int2str`.

b. Write a script `show_mean_and_variance.m` which plots the mean and variance of each pixel as two new images (see lecture, Chapter 1).

c. Write a script `show_pixel_distr.m` which plots the gray value distribution of pixel number 297 (using the vector representation of the image for indexing) for all given images of digit 3. The pixel is located at the white dot shown in Figure 2.

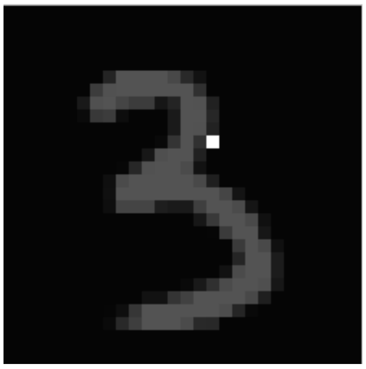
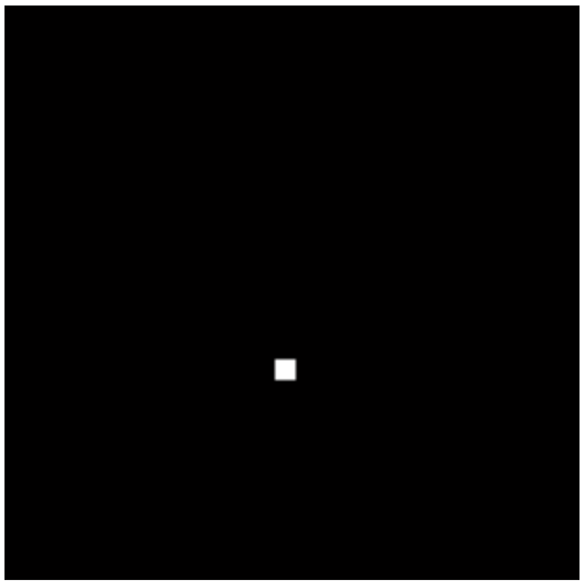


Figure 2. Illustration of the pixel 297.

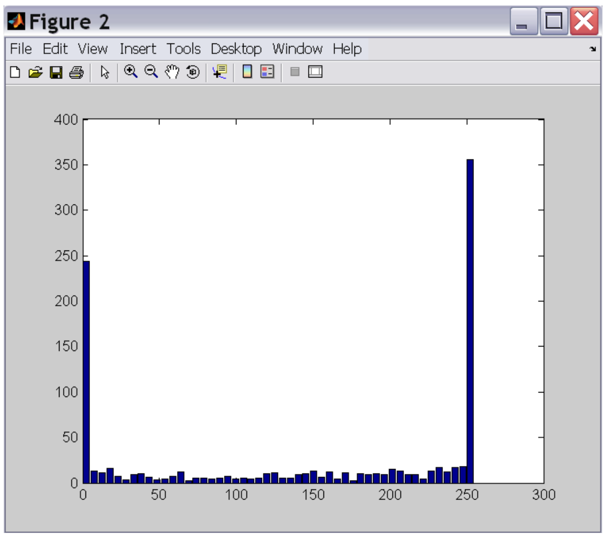
d. Extend the tool in exercise c. to sequentially analyze the images of a given digit at 10 different *interesting* pixel positions. The current pixel must be adequately illustrated.

Example:

Figure 1 illustrates the current pixel.



while Figure 2 displays the corresponding gray value distribution:



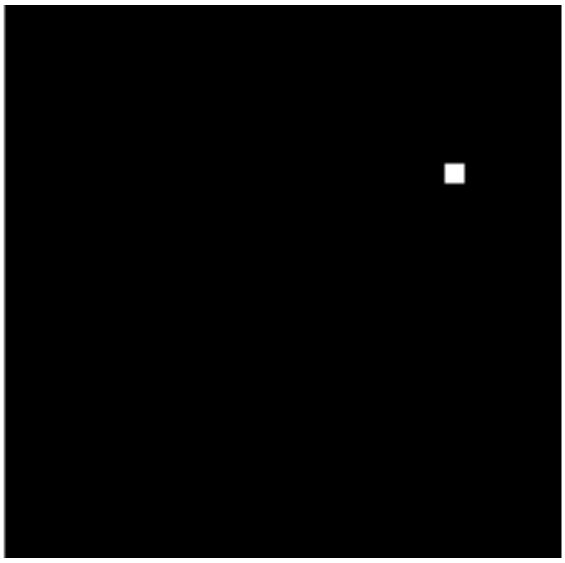
e. Generate covariance images, showing the correlation (function `corr`) between a given *base pixel* and all other pixels in the image. To this end write a function `show_correlation()` which receives three parameters:

- digit to be analyzed
- x coordinate of base pixel
- y coordinate of base pixel

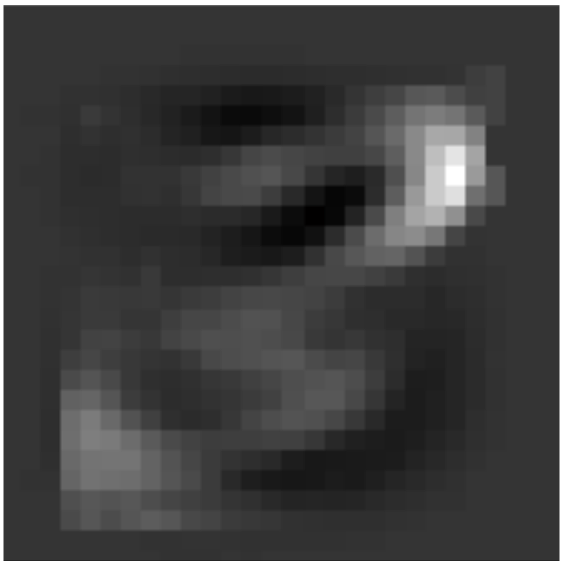
As in exercise d. the function must also illustrate the considered pixel in an extra figure.

Example:

Calling the function `show_correlation (3, 23, 9)` should generate an image showing the analyzed pixel (hint: the origin is at the top left, x runs from left to right and y from top to bottom)



and another one displaying the corresponding correlation image.



f. Write a function `show_joint_probs()`, that considers all images of a specific digit and computes for each pixel the conditional joint probability:

```
P( gray value >= min, gray value  < max | digit, pixel )
```

As with exercise b. the result has to be displayed as a new image in format 28 x 28.

The three input parameters of the function are:

- `min : the parameter ‘min’ in the conditional joint probability`
- `max : the parameter ‘max’ in the conditional joint probability`
- `digit (0 to 9) : the parameter digit in the   -  "  -`

Note that the interval of the minimal and maximal probability `[min(P), max(P)]` should be linearly mapped to the full gray value range from 0 to 255 in order to get an optimal display.

Generate images for the gray value intervals (defined by the parameters `min` and `max`):

- `[0, 1[`
- `[1, 50]`
- `[50, 150]`
- `[250, 255]`

You may use the following Octave functionalities for solving this exercise:

- `A = B > 100;`   % Produces a logical matrix `A` with the same number of rows and
  % columns as `B`, containing 1 for elements where `B` exceeds 100.
- `sum(A);`   % try ‘`help sum`’