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

Pattern Recognition

Exercises

Practice Sheet 6

Remark

Submit a report containing your findings and source code until the given deadline.
Submission tool: OLAT (eLearning platform)

Important note, valid for all exercise sheets: An extension of the deadline can not be accepted since template solutions will be made available.

Exercise L-6.1 (Face pixel classification)

In this exercise, we will apply pattern classification techniques to 2D face image pixels (see Figure 1. for an example image).

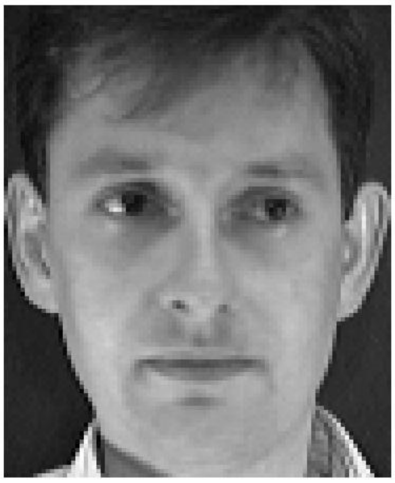

Figure 1. Face image

In order to train and evaluate the classifiers, you are provided 75 training and 100 test images. Each image contains 112 x 92 pixels with gray values between 0 (dark) and 255 (white). The matrix train_images serves as a container for all training images which are stored as follows: Each row of the matrix contains a complete image with 10304 (112*92) gray values. This is achieved by sequentially storing the individual rows of an image into a row vector as illustrated in Figure 2.

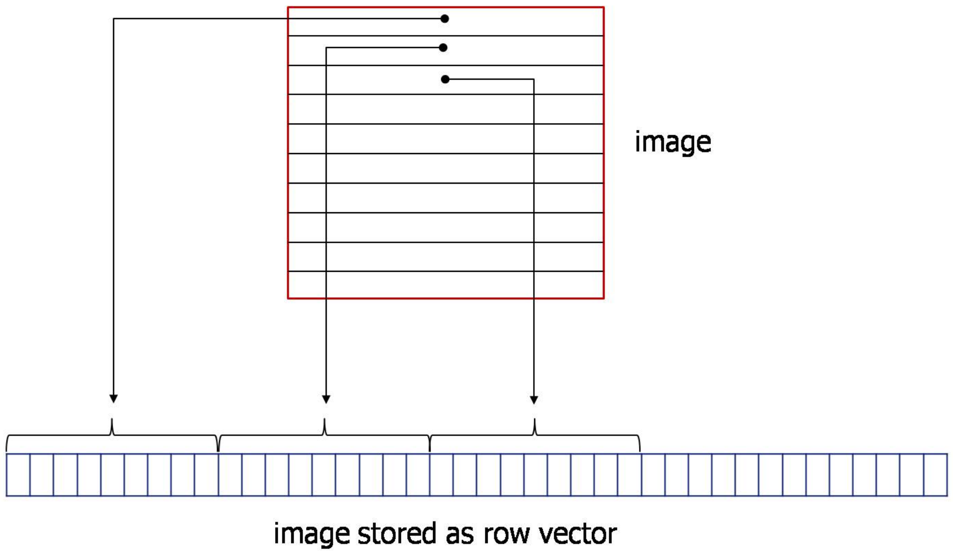

Figure 2. Illustration: Image stored as a row vector.

The following Octave script demonstrates how the training images, which are given in the OLAT Material Folder, are loaded and displayed:

```
load 'train_images.mat'
for i = 1:size(train_images,1)
    I = train_images(i,:);
    I = reshape(I, [112,92]);
    figure(1), imshow(I,[]);
    pause(0.1);
end
```

You may find another file, `test_images.mat`, with the images required for testing in the same folder.

In this exercise we will distinguish 2 classes,

- face pixels and
- non-face pixels,

using two different classifiers: The first is a ***minimum-distance classifier*** and the second a ***Bayes classifier*** using two multivariate Gaussian densities with full covariance matrices (one multivariate Gaussian for each class). See the lecture (especially the application example in Chapter 2.6) for further details.

The ground truth information (i.e. which image pixel is a face pixel and which is not) for the training images is provided in the same format as the image data itself and stored in the file `train_labels.mat`. Face pixels are labeled with ‘1’ and non-face pixels with ‘0’. Note that due to an automatic label generation procedure this ground truth is not at all perfect. Figure 3 shows an (good) example for an image with ground truth label information (white: class face, black: class non-face).

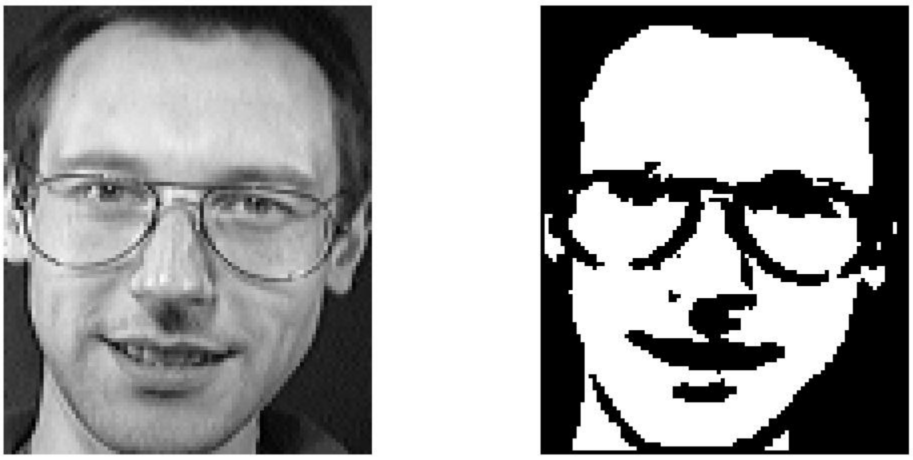

Figure 3. Original and label image

The classification of each image pixel will be based on a simple feature extraction which uses the so-called 9-neighborhood of the pixel to form a 9-dimensional feature vector as illustrated in Figure 4.

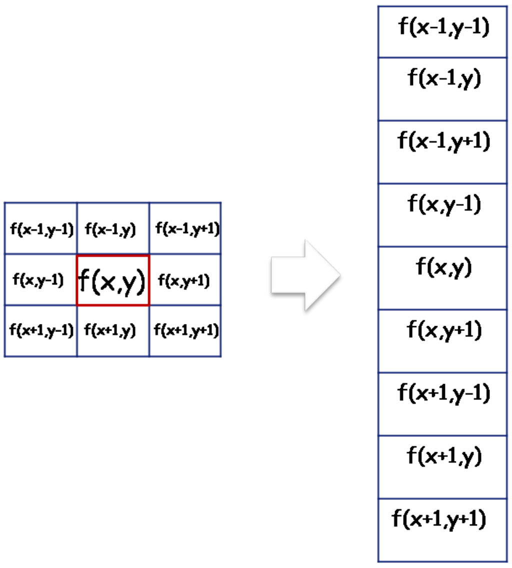

Figure 4. The 9-neighborhood of a pixel f(x,y). These 9 values form the feature vector which will be used for classification.

The script below demonstrates how to process an image with N face pixels and M non-face pixels into

- a 9 x N face feature matrix `faceFeats` and
- a 9 x M non-face feature matrix `nonFaceFeats`.

Each column of these matrices contains a single 9-dimensional feature vector.

Octave script for feature extraction of the image and label matrix:

```
faceFeats = [];
nonFaceFeats = [];

numPix = 0;
for i = 1:size(images,1)

    I = images(i,:);             % get next image from image matrix
    I = reshape(I, [112,92]);    % form a matrix from row vector

    P = labels(i,:);             % get next label matrix
    P = reshape(P, [112,92]);

    % Convert all possible neighborhoods in image to columns
    % B contains as many columns as the number of pixels in I. (This
    % is achieved by adding zeros at the image margin.) im2col is contained
    % in the Octave package image. If it is not available on your
    % system you may write its functionality on your own.
    B = im2col(padarray(I, [1, 1], 0, 'both'), [3, 3], 'sliding');
    faceFeatsNew = B(:,logical(P(:)));
    nonFaceFeatsNew = B(:,~logical(P(:)));

    faceFeats = [faceFeats faceFeatsNew];
    nonFaceFeats = [nonFaceFeats nonFaceFeatsNew];
end
```

After extraction of face and non-face features you must estimate the mean vector and covariance matrix of each class and apply it as described in the lecture for classification of evaluation images (provided in test_images.mat). For orientation, you may consult the face pixel classification tool described in the lecture.