Zurich University
of Applied Sciences

School of
Engineering
ISC Institute of Signal Processing
and Wireless Communications

Digital Image Processing

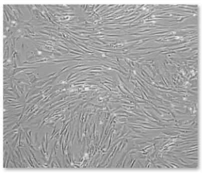
M. Weisenhorn/J. Rosset

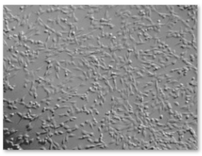
22. November 2022

Laboratory – Bayes Classifier

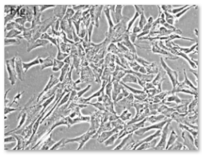
1 introduction

A given microscopic image of muscle cells should be categorized as representing one of the four types of muscle cells shown in the images below. For this you will implement a Bayes classifier. From the microscopic image, four features will be extracted from its co-occurrence matrix to form a feature vector $\boldsymbol{x} \in \mathbb{R}^4$. **Note** that in this lab, the number of features in a feature vector is four and the number of classes is also four by coincidence. In general however, they will be different.


mc1.tif


mc2.tif


mc3.tif


mc4.tif

Bayes Classifier. Simplifying assumptions for the classifier are:

- The likelihoods $p(\boldsymbol{x}|K_j)$ for $j \in \{1, 2, 3, 4\}$ are assumed to be multivariate Gaussian, i.e. expressed completely by mean vectors $\boldsymbol{m}_j$ and covariance matrices $\boldsymbol{C}_j$.
- The loss function $L_{ij} = 1$ if $i \neq j$ and 0 otherwise.
- The prior probabilities $p(K_j)$ are identical for all classes.

Under these assumptions, the Bayes classification rule simplifies to

$$\hat{j} = \arg\max_j \left\{ d_j(\boldsymbol{x}) \right\},$$

with

$$d_j(\boldsymbol{x}) = \ln(P(K_j)) - \frac{1}{2}\ln(|\boldsymbol{C}_j|) - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{m}_j)^T \boldsymbol{C}_j^{-1}(\boldsymbol{x} - \boldsymbol{m}_j),$$

from which we can skip the constant term $ln(P(K_j))$ as $K_j = K_i$ for all i, j, i.e. we can use

$$d_j(\boldsymbol{x}) = -\ln(|\boldsymbol{C}_j|) - (\boldsymbol{x} - \boldsymbol{m}_j)^T \boldsymbol{C}_j^{-1}(\boldsymbol{x} - \boldsymbol{m}_j).$$

Feature Generation. Each image was decomposed into 6x6 non-overlapping sub-images or tiles and a 4-dimensional feature vector was generated for each tile. Specifically, the co-occurrence matrix for the grayscale image with 13 grayvalues was calculated and based on this the *energy*, *contrast*, *entropy*, *homogeneity* measures were calculated. The total of $36 \cdot 4 = 144$ training vectors are available in the file `allFeaturs.csv`. They were divided into a training set and a test set.

Training Set. The training feature-vectors are available for use with matlab and python in the files `featuresForTraining.mat` and `featuresForTesting.pkl`, respectively. The training vectors contain an equal number of representatives of each of the classes 1 to 4. The training vectors for the different object classes are held in different data objects that are accessable via the class index j.

Test Set. The Feature vectors for testing have also be computed from tiles of the images below and made available in `featuresForTesting.mat` and `featuresForTesting.pkl`. The tiles from which the test feature-vectors have been computed are disjoined from the tiles the training feature-vectors have been computed from. The test data contains an equal number of representatives of each of the classes 1 to 4. The test vectors for the different object classes are held in different data objects that are accessable via the class index j. The function `unveil_labels` puts the vectors from all four classes into a single array and provides a further array of the same dimension that contains the class labels.

2 Tasks

a) The reading of the training and test data is prepared. Make yourself known with the code e.g. by inspecting variables in debug mode.

b) Complete the function `train` by determining the class centers $\boldsymbol{m}_j \in \mathbb{R}^4$ and the covariance matrices $\boldsymbol{C}_j \in \mathbb{R}^{4\times 4}$ for all four classes. The mean value $\boldsymbol{m}_j$ for class j is formed by the N training vectors $\boldsymbol{x}_{jn}$ for $n \in 1, 2, \ldots N$:

$$\boldsymbol{m}_j = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_{jn}$$

The covariance is formed by

$$\begin{aligned}\boldsymbol{C}_j &= \frac{1}{N-1}\sum_{n=1}^{N}(\boldsymbol{x}_{jn} - \boldsymbol{m}_j)\cdot(\boldsymbol{x}_{jn} - \boldsymbol{m}_j)^T \\ &= \frac{1}{N-1}\boldsymbol{D}_j\,\boldsymbol{D}_j^T,\end{aligned}$$

where $\boldsymbol{D}_j = [\boldsymbol{d}_{j1},\ \boldsymbol{d}_{j2}, \ldots \boldsymbol{d}_{jN}]$, with $\boldsymbol{d}_{jn} := \boldsymbol{x}_{jn} - \boldsymbol{m}_j$.

c) Implement the `test` functions `train`, `classify`, and optionally `computeConfusionMatrix`.

d) Run the function `main`. You should not observe any misclassifications, as the training and test data sets have been tuned to just achieve perfect classification.

e) Now degrade the classifier by assuming that the covariance matrix is the unity matrix. As a result you should observe misclassifications.