Heidi Gebauer (geba), Markus Roos (roor)
School of Engineering T

Skin Detection based on Classification Algorithms: Exercise Set

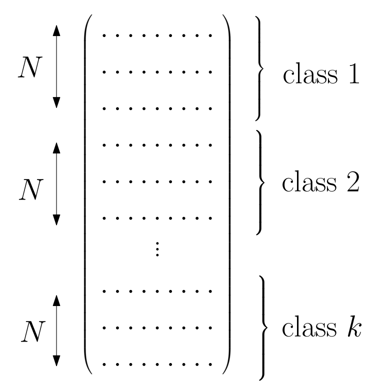
Remark: Please note that the module `load_data.py` provides the required files in form of `Python dict` by a call `load_data()` to and the auxiliary functions s `= distance(`p`,` q`)` and `classify_nearestmean(`A`,`M`,`k`)` (described below) in module `helper.py` are available on moodle.

Task 1 Consider the function s `= distance(`p`,` q`)`, which computes the euclidean distance between two (not necessarily 2-dimensional) points p and q.

Task 2 The goal is to evaluate various strategies for classifying a given point.

(a) Have a look at the function
`[`α`] = classify_nearestmean(`A`,` M`,` k`)` (found in `helper.py`), which takes as input:

- an integer k denoting the number of classes,
- a matrix M each row of which represents a not yet classified point, and
- a matrix A whose rows can be partitioned into k equally sized sets of points belonging to the same class: With N denoting the number of elements per class, the points of class i are contained in the rows of $\mathtt{range}(\mathtt{N}\cdot\mathtt{i},\mathtt{N}\cdot(\mathtt{i}+1))$, for each $i \in \{0,\ldots,k-1\}$ (see the illustration to the right),



and returns a vector α with $\alpha(i)$ denoting the class assigned to the point of the ith row of M according to the nearest mean strategy.

(b) Implement a function $[\alpha]$ = classify_gmm(A,M,k) which is obtained from modifying the function from part (a) in such a way that α is determined using Gaussian mixture models.

Hint: You will need the `Python` function `np.cov( )`, as well as the class `multivariate_normal` found in `scipy.stats`, which provides the method `pdf( )`. See the help pages for details.

Test Cases: For the input (`A1`, `M1`, 3) you should get [2, 2, 1, 1]. Furthermore, also check whether in your internal computation, you get means = [19.7997 60.5564], [81.0068 60.4333], [39.5510 19.7952], and

$$\text{covariance matrices} = \begin{pmatrix} 29.1874 & 0.9561 \\ 0.9561 & 27.9604 \end{pmatrix}, \begin{pmatrix} 33.2761 & -1.6249 \\ -1.6249 & 26.8107 \end{pmatrix}, \begin{pmatrix} 30.5662 & -0.2978 \\ -0.2978 & 25.2951 \end{pmatrix}.$$

For the input (`A2`, `M2`, 2) you should get [0, 1, 1, 1, 1].

Goal: In the subsequent tasks you will analyze various algorithms for skin detection. The below illustration shows a 'perfect' skin detection.

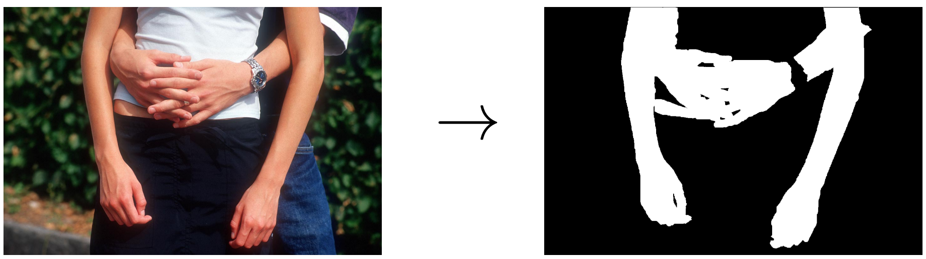


image credit: Prof. Thomas Vetter, University of Basel

Convention: Throughout this exercise sheet, intensity values of pixels are in the range of $[0, 1]$.

Task 3 The goal is to evaluate the algorithms from Task 2 for the datasets `skindata` and `nonskindata` and the image `test`.[1]

(a) First, implement the following auxiliary functions:

- *M*, *dims* = `rgbImage2Matrix(`*img*`)` which takes as input an rgb image *img* and constructs a matrix by concatenating the rows of *img* (as well as returns a tuple `dims` of the matrix dimensions.

  (Example: The image I[2] $= \begin{pmatrix} [0.1\ 0.2\ 0.3] & [0.4\ 0.6\ 0.8] \\ [0.8\ 0.2\ 0.5] & [0.6\ 0.3\ 0.9] \end{pmatrix}$ becomes $M = \begin{pmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.6 & 0.8 \\ 0.8 & 0.2 & 0.5 \\ 0.6 & 0.3 & 0.9 \end{pmatrix}$)

- *img* = `Vector2grayImage(`S`,`r`,`s`)` which takes as input a vector S and constructs a grayscale image I of size $r \times s$ by copying the elements of S column by column.

  Example: $S = [0.1, 0.5, 0.4, 0.6]$, $r = 2$ and $s = 2$ gives $\begin{pmatrix} 0.1 & 0.5 \\ 0.4 & 0.6 \end{pmatrix}$.

[1] These files are from Prof. Thomas Vetter, University of Basel.

[2] provided by `load_data`

(b) Write a script that applies each strategy of Task 2 to the image `test` (using the previously developed auxiliary functions). Represent the output as a binary image (1 = 'skin', 0 = 'non-skin')

- Evaluate the results visually (i.e. by looking at the result).
- For each result, determine the corresponding error rate (the 'true classification' can be found in `maskTest`). [3]

(c) **Optional:** Apply your algorithm based on the Gaussian mixed model to some other images (e.g. from Google) and determine visually whether the result is good.
Remark: This part does not have to be handed in.

Task 4 The goal is to determine a skin/non-skin classification using Support Vector Machines. Write a script (using appropriate Matlab commands) that first trains a classifier for `skindata` and `nonskindata` and then applies the result to `test.png`.

Determine for the error rate, the false positive rate and the false negative rate and compare them with the respective values for the Gaussian mixture model.

Remark: It is recommended to tailor your script to the given situation (i.e., 2 groups of data, $d = 3$ dimensions and $N = 10'000$ samples for each training set). No generalizations are required!

[3] We will refer to this value as *test error*.