10907 Pattern Recognition

Lecturers
Prof. Dr. Thomas Vetter ⟨thomas.vetter@unibas.ch⟩

Assistants
Dennis Madsen ⟨dennis.madsen@unibas.ch⟩
Dana Rahbani ⟨dana.rahbani@unibas.ch⟩
Moira Zuber ⟨moira.zuber@unibas.ch⟩
Genia Böing ⟨genia.boeing@unibas.ch⟩

Exercise 2 — Maximum likelihood and Skin Detection

Introduction	07.10
Deadline	**15.10** Upload code to Courses.
	14.10+15.10 Group presentations, U1.001, see schedule online

In this exercise you will write a skin detector, which is able to classify pixels of an image as *skin* or *non-skin*. For the following experiments we provide two data files `skin.mat` and `nonskin.mat` containing RGB colour data (format: 3×#samples). This data represents pixel values from several photographs which were manually labelled as belonging to either *skin* or *non-skin* regions.

Grading: We will ask questions about the exercises listed here. The grading will be 50% coding and 50% theory questions. The questions will be about the required conceptual understanding. The coding part will focus on concrete figures and results. Please appear prepared with the required calculation results readily available (error rates etc.). The total time of discussion has to fit within 15 minutes and each student of the group should contribute.

You can download the data needed for this exercise and a code skeleton from the following repository: `https://bitbucket.org/gravis-unibas/pattern-recognition-2019`.
A number of helper functions are provided - your task is to implement the *TODO* parts in the python (.py) files.

Remember to upload your code to courses in a ZIP file. Do NOT include the data folder. Only the python files you edited should be included + a file containing the group members names. Only 1 person from the team should upload!

1 Maximum Likelihood - Manual implementation

Estimation of dataset distributions for a 2D toy dataset example. The main function for this exercise is found in the file `ex2-ML_1_Toy.py`. You need to fill in the `TODO` sections in the file. Do NOT use build in *cov*, *mean*, *inv*, *det* functions from numpy - you need to implement this functionality!

- Estimate the Multivariate normal distribution (MVND) for each dataset (compute sample mean and sample covariance matrix as we assume the data is gaussian distributed). What is the sample mean and covariance matrix of each of the datasets?
- Implement the probability density function (pdf) of the 2-dimensional MVND. Also implement and make use of the helper functions: matrix-inverse and matrix-determinant.
- Make use of the pdf function to classify a datapoint into class 0 or class 1.

The final step is to sample from the 2 distributions (see function `sample_from_2d_gaussian`). The only random generator you are allowed to use is the uniform distribution $\mathcal{U}(0,1)$ (`np.random.random()`). These samples need to be transformed into samples from a standard normal distribution $\mathcal{N}(0,1)$. This is can be achieved with the *Box-Muller transform*:

$$z_0 = \sqrt{-2\log(u_0)}\cos(2\pi u_1)$$
$$z_1 = \sqrt{-2\log(u_0)}\sin(2\pi u_1)$$

with u_0 and u_1 being samples from a uniform distribution. Finally the 2d vector sampled from a MVND needs to be computed. To do so, we can compute:

$$\boldsymbol{x}_{sample} = L\boldsymbol{z} + \boldsymbol{\mu}, z_i \sim \mathcal{N}(0, 1)$$

where L is the matrix factorization of the covariance matrix:

$$\Sigma = LL^T$$

which can be obtained with the Cholesky decomposition of the covariance matrix Σ.

2 Maximum Likelihood - Skin detection

Estimate a Gaussian distribution for each dataset (skin + non-skin). You should use a full covariance matrix. Use the maximum likelihood estimators known from the lecture. The *main* function for this exercise is found in the file `ex2-ML_2_Skin.py`. In this exercise you are allowed to use all the power of `numpy` to compute the MVND.

2.1 Detection

(a) **Multivariate Normal Distrbution** Implement a MVND class. The general structure of the class is found in the file `myMVND.py`.
Hint: The MVND class takes as input a list of Gaussian distributions. In this exercise, only one list item will be given to the function. In Section 2, when you implement the GMM, you need to provide multiple list items.

(b) **Prior Probabilities** Estimate the prior probabilities for each classification-class (skin & non-skin) in `classifyHelper`. The prior probabilities are estimated from the image `image.png`. The skin label for the image is provided in `mask.png`.

(c) **Likelihood Classifier** Implement a Bayes classifier to distinguish between skin and non-skin. Provide the output as a binary image. In the file `classifyHelper.py` the *likelihood* function needs to be finalized. This function will compute the likelihood for each pixel in an image, given a list of distributions. You also need to compute the false positive/negative and total error for the skin classification problem, both with and without the use of the skin-prior.

3 Gaussian Mixture Models

The task is now to extend the model the skin/non-skin classes with a Gaussian Mixture Model instead of modelling with a single Gaussian distribution. The *main* function for this exercise is found in the file `ex2-ML_3_GMM.py`.

3.1 EM-Algorithm

The function `gmm_em()` needs to be implemented. The EM-Algorithm for Gaussian Mixture Models can be found in the lecture slides *"Density_Estimation_2_GMM"*.

You can visualize the progress of the EM-Algorithm by using the function `gmm_draw` (provided in the `ex2-GMM.py` file). After each iteration plot, the data points are coloured according to their current cluster assignment for a visual debugging.

Develop and test your EM algorithm using the two-dimensional toy data provided in the file `gmmdata.mat`, work with K=3 components (see function `gmmToyExample()`). To initialize, the algorithm uses a random cluster assignment for each data point. Run the algorithm long enough to converge (check at least graphically) and report the obtained cluster means covariance matrices.

Make sure that your `gmm_em()` function works according to specifications:

- K - Number of GMM components, integer (>=1)
- iter - Number of iterations, integer (>=0)

3.2 Skin Detection

Train a GMM with two *K=3* components for each dataset (`skin` and `non-skin`) and use a Bayes classifier for classifying the pixels into skin and non-skin (see function `gmmSkinDetection()`). Report the error rate on the training and the test set. What are the error rates if you use 1, 2, 3 or 4 components?