Prof. Dr. Thomas Schultz
Shekoufeh Gorgi Zadeh (gorgi@cs.uni-bonn.de)

B-IT
Bonn-Aachen International
Center for Information Technology
May 14, 2018

Summer term 2018

Visual Computing in the Life Sciences

Assignment Sheet 4

If you have questions concerning the exercises, please write to our mailing list:
vl-bioinf@lists.iai.uni-bonn.de.

This exercise can be submitted in **small groups** of 2-3 students. Please submit each solution only once, but clearly indicate who contributed to it and remember that all team members have to be able to explain it.

*Solutions has to be sent until **June 04, 2018, 10:30am** to gorgi@cs.uni-bonn.de. Please bundle the results as PDF and scripts in a single ZIP file. Include your names explicitly in the PDF. Use the following format for naming the files "vclsi-x-lastName.pdf" and "vclsi-x-lastName.ipynb", where x is the ID of assignment sheet. For example, for Alice Smith, the file name must be vclsi-4-smith.pdf.*

Exercise 1 (Spectral Clustering, *22 Points*)

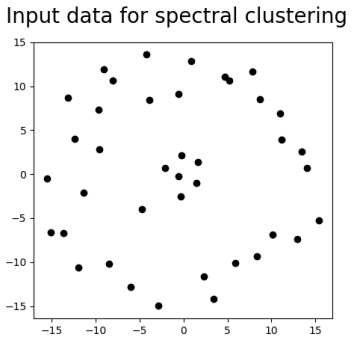

Figure 1: The example input dataset that is to be clustered with Spectral Clustering.

In this exercise you will use spectral clustering for clustering the 2D data shown in Fig. 1 into two separate clusters. Download `FramworkPoints.py` file and use it to implement the following steps:

a) Implement `GetPointEdges`, that has three input parameters, i.e., `Points`, `SigmaDistance`, and `EdgeRadius`. This function must create edges between each pair of points (x_i, y_i) and (x_j, y_j) from `Points` whose distance is below `EdgeRadius`. Edge weights should be proportional to a Gaussian,

$$w_{ij} = \exp\left(-\frac{(x_i - x_j)^2 + (y_i - y_j)^2}{2 \cdot \sigma_D^2}\right),$$

with σ_D given as `SigmaDistance`. (5P)

b) Implement `GetLaplacian()`. This function uses the Graph computed from part a) to create the Laplacian matrix L. (5P)

c) Implement `GetFiedlerVector()`. This function must find and return the eigenvector corresponding to the second smallest eigenvalue of the spectral decomposition of matrix L. (5P)

d) Plot the sorted coefficients of the Fiedler vector. Do they make it easy to define a suitable threshold to obtain a clear clustering? What threshold would you choose and why? (3P)

e) Visualize the data points in a scatter plot, and use the binarized Fiedler vector from part b) to assign different colors to each cluster. (4P)

Exercise 2 (GMMs and EM Algorithm for Image Segmentation, *24 Points*)

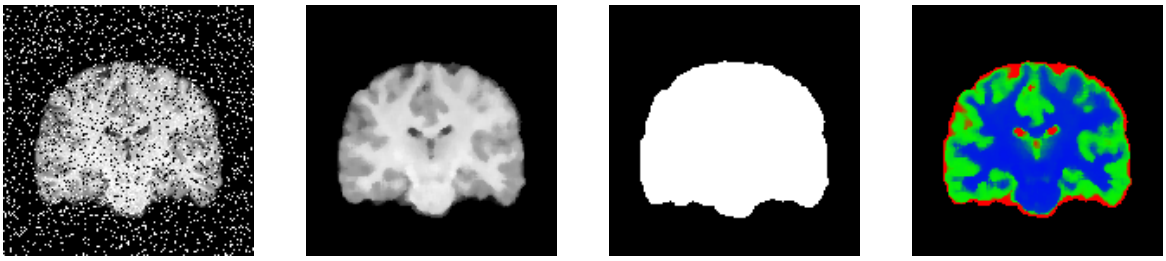

Figure 2: From left to right: The input image, after denoising, the binary mask, segmented brain image.

In this exercise, you will implement a Gaussian Mixture Model (GMM) to produce a probabilistic image segmentation, and to find suitable parameters automatically, by using the EM algorithm.

Hint: Making proper use of Python packages greatly reduces the amounts of code you have to write for this task. Routines for reading and writing images can be found in the Python module `scipy.misc`. Other useful routines for working with images are in `scipy.ndimage`. You may also find routines from `numpy` useful for parts of your analysis. We also recommend using `scikit-image` library for some basic image processing functions.

Proceed in the following steps:

a) Read the grayscale image `brain.png`, which is provided on the lecture homepage. Reduce the salt and pepper noise in the image using a median filter. (3P)

b) Produce a binary mask that marks all pixels with an intensity greater than zero. In all further steps, only treat pixels within that mask. (1P)

c) Plot a log-scaled histogram of the pixels within the mask. It should show how frequently different intensity values occur in the image. What do the peaks in this histogram represent? *Hint:* One way to find out is to create masks that highlight the pixels belonging to each peak. (4P)

d) Now, we will use a three-compartment Gaussian Mixture Model for image segmentation: Based on their gray level, pixels that fall within the mask from c) should be assigned to one of three Gaussians, capturing corticospinal fluid (dark), gray matter (medium), or white matter (bright). To start this process, initialize the parameters of a three-compartment GMM to some reasonable values and use them to compute the responsibilities ρ_{ik} of cluster k for pixel i. (4P)

e) Visualize the responsibilities by mapping the probabilities of belonging to the CSF, gray matter, and white matter clusters to the red, blue, and green color channels, respectively. Please submit the resulting image. (3P)

f) Use the update rules provided in the lecture to re-compute the parameters μ_k, σ_k, and π_k. (4P)

g) Iterate the E and M steps of the algorithm until convergence. Please submit the final parameter values, a visualization of the final responsibilities, and your code. (3P)

h) Create and submit a plot that illustrates the convergence of your algorithm. (3P)

Exercise 3 (Updating σ_k in the EM Algorithm, *4 Points*)

Derive the update rule for σ_k that is performed in the M-step of the EM algorithm:

$$\sigma_k^2 = \frac{\sum_{i=1}^{n} \rho_{ik}(x_i - \mu_k)^2}{N_k}$$

Hint: The derivation is very similar to the one for μ_k, which was shown in the lecture.

Good Luck!