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

Zürcher Hochschule für Angewandte Wissenschaften

Clustering – Approaches and Applications: Exercise Set

To be handed in: Your solution to tasks 3, 4 and 5.

Preparation: To load the required data sets, call the function `load_data()` (available on moodle) that returns a `Python dict` with the sets (including the image of the moon), addressable with key strings like `'data1'`, `'moon'`, etc.

Task 1 Have a look at the following three functions available on moodle or within `sklearn.metrics`:

(i) `plot_clusters(`X`,` α`)`, which takes as input

- a matrix X each row of which represents the coordinates of a point, and
- a vector α containing cluster indices of each point;

and plots the clusters with different colors.

Remark: We assume that the number of clusters specified by α is at least 2 and at most 5.

(ii) val = `silhouette_score(`X`,` α`)` from `sklearn.metrics`, which averages the silhouette values of the single clusters for data set X and 'cluster-indices-vector' α. There is a related function `silhouette_sample(`X`,` α`)` that provides the silhouette values of each sample instead of their mean.

(iii) `cluster_kmeans(data)`, which applies the k-means-algorithm to the given dataset for each $k \in \{2, 3, 4, 5\}$, then selects the 'best' of these clusterings (with respect to the silhouette value), and finally plots the result.

Task 2 First, familiarize yourself with the `KMeans` class in `sklearn.cluster`, especially the methods `fit()` and `predict()` that are primarily used here.

Further down, you find 10 sets of points. They correspond to the datasets in the given folder. Run the script `task2_test_kmeans` (available on moodle) to apply the function from Task 1.(iii) for each of these sets. Identify those datasets, where the clustering determined by `cluster_kmeans` deviates from the 'natural clustering' and try to formulate the reason for the actual outcome of the k-means algorithm.

Note: Since the k-means algorithm is probabilistic, its results may vary. So, if for some dataset the output is different from what you expect then re-run `cluster_kmeans` (for this particular input) to make sure that this is not an atypical outcome.

Task 3 Adapt the code in `cluster_kmeans(data)` in such a way that the clustering is done according to the second approach presented in the lecture.
Hint: Study the help pages for `BayesianGaussianMixture` of `sklearn.mixture`. To proceed efficiently, just investigate the mandatory parameters and settings. Spoiler: you only have to modify a few lines!

Task 4 For all those datasets, where the algorithm for Task 2 gave an 'unnatural' result: Determine the clustering obtained by the algorithm for Task 3 and comment on the differences. Add your findings as comments into the script for task 3.

Task 5 The goal of this task is to **segment an image** using k-means. We use the convention that each pixel is represented by an intensity value in the interval $[0, 1]$.

(a) First, implement the following auxiliary functions:

- a function V, *dims* = `grayscaleImage2Vector(`I`)` which takes as input a grayscale image I and constructs a vector by concatenating the rows of I (as well as returns the dimensions *dims* of the image).

  (Example: The image $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ becomes $\begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{pmatrix}$)

- a function I = `vector2GrayscalImage(`V`,(`r`,`s`))` which constructs an $r \times s$ matrix I by copying the elements of V row by row.

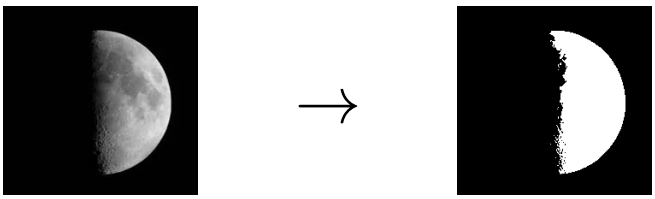
  (Example: For $V = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{pmatrix}$ and $r = 2$, $s = 3$ the result is $I = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$)

  Hint: Have a look at the `numpy` methods `reshape()` for arrays.

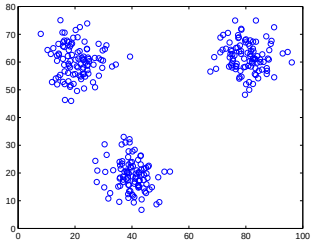
(b) Write a script which takes the image `moon.jpg`, converts it into a grayscale image with values in $[0, 1]$ and then proceeds as follows:

- Transform the image into a vector (using the auxiliary function from part (a)).
- Divide the intensity values into 2 clusters using the k-means algorithm. We refer to the centroid with the smaller value as c_1, and to the centroid with the larger value as c_2.

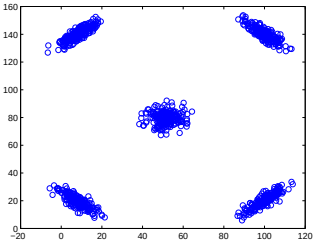
- To all pixels in the cluster of c_1: assign color black,
  to all pixels in the cluster of c_2: assign color white.
- Transform the vector back to an image of the original size (using the auxiliary function from part (a)) and display the result.



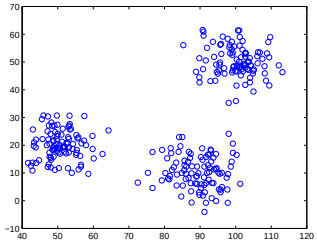
Remark: This is only one of many approaches you can use for image segmentation! (A couple of other techniques will be presented in the next lecture.)



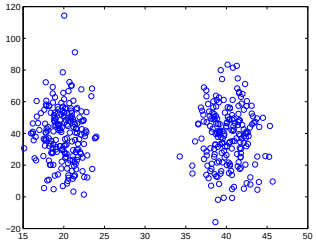
data1



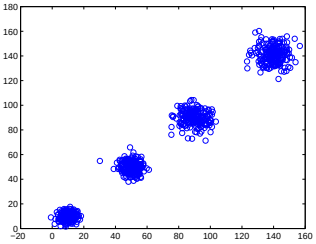
data2



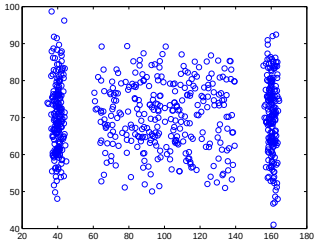
data3



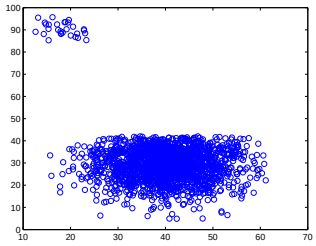
data4



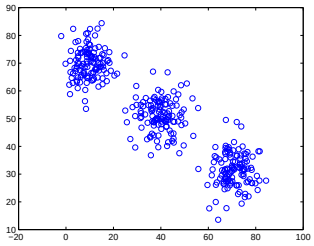
data5



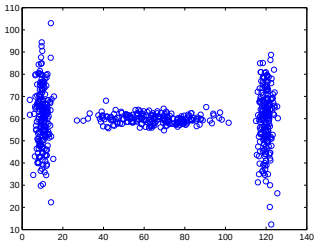
data6



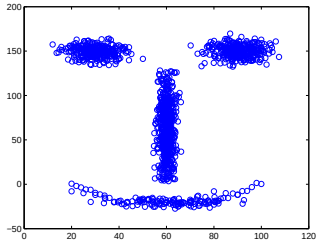
data7



data8



data9



data10