CS5487 Programming Assignment 2
Clustering

Antoni Chan
Department of Computer Science
City University of Hong Kong

In this programming assignment you will implement and test several clustering algorithms on both synthetic and real data. Given a set of points $X = \{x_1, \cdots, x_n\}$, where $x_i \in \mathbb{R}^d$, the goal is to assign a cluster label to each point, $y_i \in \{1, \cdots, K\}$, where K is the number of clusters. In this programming assignment, you will consider three clustering algorithms:

1. **K-means algorithm** – The K-means algorithm calculates the cluster centers μ_j using the current assignment to each cluster (K is assumed to be known). In each iteration, K-means performs the following two steps:

$$\text{Cluster Assignment}: \; z_{ij} = \begin{cases} 1, & j = \text{argmin}_{k\in\{1,\cdots,K\}} \|x_i - \mu_k\|^2 \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

$$\text{Estimate Center}: \; \mu_j = \frac{\sum_{i=1}^n z_{ij} x_i}{\sum_{i=1}^n z_{ij}}. \tag{2}$$

   The cluster label for point x_i is the label of the closest cluster center, $y_i = \text{argmax}_j \, z_{ij}$.

2. **EM algorithm for Gaussian mixture models (EM-GMM)** – The EM algorithm calculates a maximum likelihood estimate of a GMM with K components, with parameters $\{\pi_j, \mu_j, \Sigma_j\}_{j=1}^K$. K is also assumed to be known. The E- and M-steps are:

$$\text{E}-\text{step}: \; \hat{z}_{ij} = p(z_i = j|x_i) = \frac{\pi_j \mathcal{N}(x_i|\mu_j, \Sigma_j)}{\sum_{k=1}^K \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}. \tag{3}$$

$$\text{M}-\text{step}: \; \hat{N}_j = \sum_{i=1}^n \hat{z}_{ij}, \quad \hat{\pi}_j = \frac{\hat{N}_j}{n}, \quad \hat{\mu}_j = \frac{1}{\hat{N}_j} \sum_{i=1}^n \hat{z}_{ij} x_i, \tag{4}$$

$$\hat{\Sigma}_j = \frac{1}{\hat{N}_j} \sum_{i=1}^n \hat{z}_{ij} (x_i - \hat{\mu}_j)(x_i - \hat{\mu}_j)^T. \tag{5}$$

   After convergence, the cluster label for x_i is the component with the largest posterior probability, $y_i = \text{argmax}_j \, \hat{z}_{ij}$. The document "gmm-tips", available on the course website, contains some useful tips on implementing EM for GMMs.

3. **Mean-shift algorithm** – The mean-shift algorithm uses gradient ascent with an adaptive step-size to find local peaks in a kernel density estimate of X. We will use a Gaussian kernel with bandwidth h. Given an initial point $\hat{x}^{(0)}$ (where the superscript denotes the iteration number), the mean-shift algorithm update rule is:

$$\hat{x}^{(t+1)} = \frac{\sum_{i=1}^n x_i \mathcal{N}(x_i|\hat{x}^{(t)}, h^2 I)}{\sum_{i=1}^n \mathcal{N}(x_i|\hat{x}^{(t)}, h^2 I)}. \tag{6}$$

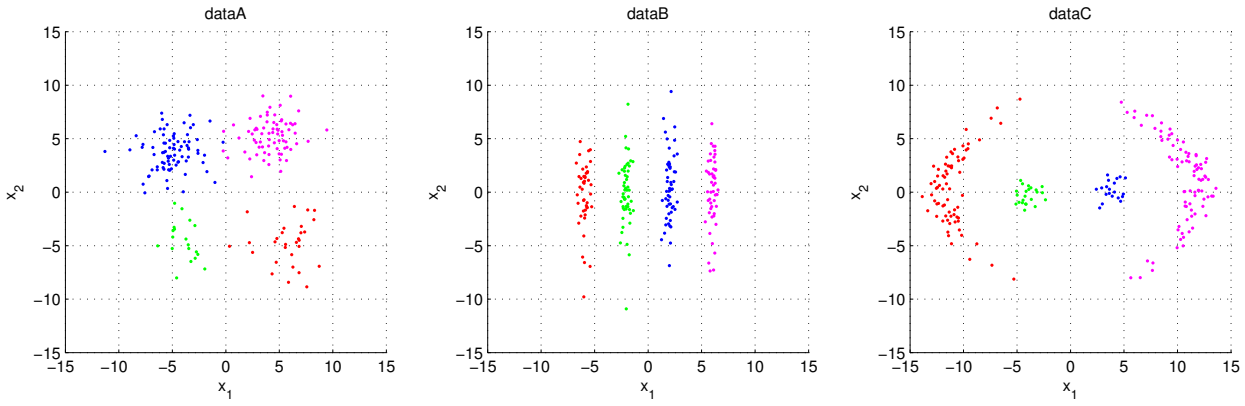
   To obtain the clustering, the mean-shift algorithm is run using each data point x_i as an initial point. After convergence, the points that converged to the same local peak are given the same cluster label. In this case, the number of clusters K is not fixed (and depends on the selected bandwidth h).

Problem 1 Clustering synthetic data

In the first problem, you will test these clustering methods on synthetic data, and examine how each method performs on different configurations of data. The zip file `PA2-cluster-data.zip` contains the synthetic data files. Inside the MATLAB data file `cluster_data.mat` (or `cluster_data_*.txt` if you are not using MATLAB) are three data sets (points and ground-truth labels). The file contains the following variables:

- `dataA_X` - dataset A points. Each column is a data point, $x_i \in \mathbb{R}^2$.
- `dataA_Y` - dataset A true labels, $\{y_i\}$.
- `dataB_X` - dataset B points.
- `dataB_Y` - dataset B true labels.
- `dataC_X` - dataset C points.
- `dataC_Y` - dataset C true labels.

The three datasets are shown in the following figure, where the color represents the ground-truth label.



The goal is to discover the clusters in each dataset using the data points X.

(a) Implement the three clustering algorithms. You will reuse these algorithms in the next problem, so try to make them as general as possible.

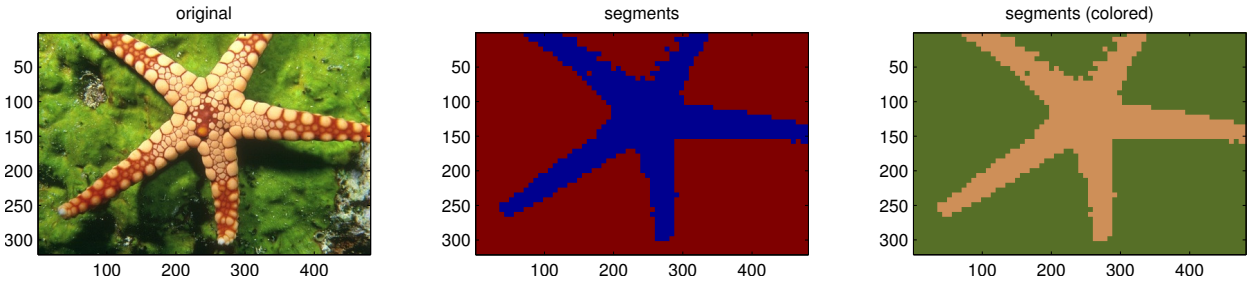
(b) Run the algorithms on the three synthetic datasets. Qualitatively, how does each clustering algorithm perform on each dataset? Comment on the advantages and limitations of each algorithm, in terms of the configuration of the data.

(c) How sensitive is mean-shift to the bandwidth parameter h?

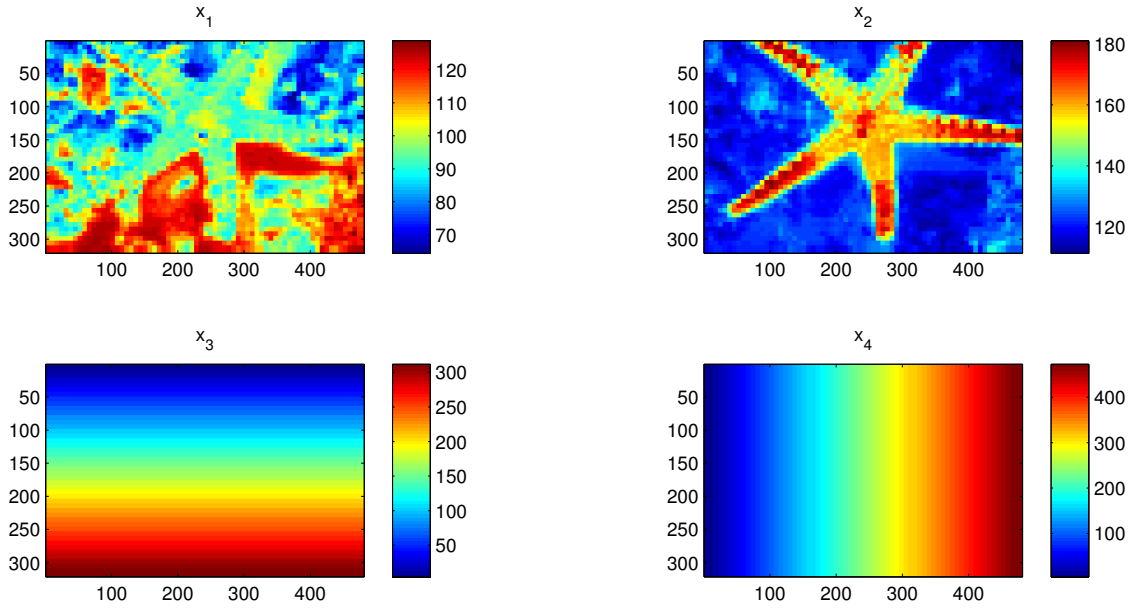
.....

Problem 2 A real world clustering problem – image segmentation

In this problem we will consider a real world clustering problem, *image segmentation*, where the goal is to find homogeneous regions in the image, which hopefully belong to objects or object parts. Below is one of the test images and an example segmentation using K-means (In the right image, each segment is colored based on the average color within the segment).



To segment the image, we first extract feature vectors from each pixel in the image (or a subset of pixels along a regular grid). In particular, we form a window centered around each pixel and extract a four-dimensional feature vector, $x = [u, v, c_y, c_x]^T$, where (u, v) are the average chrominance values (color without brightness) in the window, and (c_x, c_y) is the pixel location (center of the window in x-y-coordinates). The feature values for the example image are shown below (each subplot represents one feature dimension over the entire image).



Next, a clustering algorithm is used to group the feature vectors, and a cluster label is assigned to each corresponding pixel forming the segmentation.

The zip file `PA2-cluster-images.zip` contains the images and MATLAB code for extracting the features, and creating the segmentation image from the cluster labels. The following MATLAB functions are provided:

- `getfeatures.m` – MATLAB function for extracting features from an image.
- `labels2segm.m` – MATLAB function for generating a segmentation image from the cluster labels.

- `colorsegm.m` – MATLAB function to create a nicely colored segmentation image.

As an example, the following code was used to generate the above segmentation images:

```
% read the image and view it
img = imread('images/12003.jpg');
subplot(1,3,1); imagesc(img); axis image;

% extract features (stepsize = 7)
[X, L] = getfeatures(img, 7);
XX = [X(1:2,:) ; X(3:4,:)/10];  % downscale the coordinate features (see part (b))

% run kmeans -- this is the MATLAB version.  You have to write your own!
[Y, C] = kmeans(XX', 2);

% make a segmentation image from the labels
segm = labels2segm(Y, L);
subplot(1,3,2); imagesc(segm); axis image;

% color the segmentation image
csegm = colorsegm(segm, img);
subplot(1,3,3); imagesc(csegm); axis image
```

Documentation for each function can be viewed in MATLAB using "`help getfeatures`", etc.

For Python users, the file `PA2-cluster-python.zip` contains Python versions of the above demo code and helper functions for extracting features. This Python code requires the `numpy`, `scipy`, `matplotlib`, and `Image` modules. Alternatively, the file `PA2-cluster-imfeatures-txt.zip` contains a text files of the extracted image features for a step size of 7.

(a) Use the three clustering algorithms to segment a few of the provided images. Qualitatively, which algorithm gives better results? How do the results change with different K and h? Which is less sensitive to changes in the parameters? Comment on any interesting properties or limitations observed about the clustering algorithms.

The feature vector x contains two types of features that are on different scales; the chrominance values (u, v) range from 0-255, while the pixel locations (c_x, c_y) range from 0-512. The EM-GMM clustering algorithm can adapt to the different scales of each feature by changing the covariance matrix. On the other hand, K-means and mean-shift assume a fixed isotropic covariance matrix. We can modify these two algorithms to allow different scaling of the features:

- For K-means, the distance to a cluster center x' is modified to apply a weighting between the feature types,

$$d(x, x') = \left\| \begin{bmatrix} u \\ v \end{bmatrix} - \begin{bmatrix} u' \\ v' \end{bmatrix} \right\|^2 + \lambda \left\| \begin{bmatrix} c_x \\ c_y \end{bmatrix} - \begin{bmatrix} c'_x \\ c'_y \end{bmatrix} \right\|^2, \tag{7}$$

  where $\lambda \geq 0$ controls the weighting.

- For mean-shift, the kernel is modified to use separate bandwidths on each feature type,

$$k(x, x') = \frac{1}{(2\pi)^2 h_p^2 h_c^2} \exp\left\{ -\frac{1}{2h_c^2} \left\| \begin{bmatrix} u \\ v \end{bmatrix} - \begin{bmatrix} u' \\ v' \end{bmatrix} \right\|^2 - \frac{1}{2h_p^2} \left\| \begin{bmatrix} c_x \\ c_y \end{bmatrix} - \begin{bmatrix} c'_x \\ c'_y \end{bmatrix} \right\|^2 \right\}, \tag{8}$$

where h_c is the bandwidth for the color features, and h_p is the bandwidth for the pixel locations.

(b) Modify your K-means and mean-shift implementations to allow different feature scaling. **Hint**: changing the distance in (7) or kernel in (8) is equivalent to scaling each dimension in the feature vector x by an appropriate amount. Rerun the segmentation experiment. Do the segmentation results improve?

.....

Problem 3 Quantitative Evaluation of Clustering (Optional)

In this optional problem, you will quantitatively evaluate your results from Problems 1 and 2. Consider a set $S = \{s_1, \cdots, s_n\}$ with n elements, and two partitions of S into subsets, $\mathcal{Y} = \{Y_1, \cdots, Y_R\}$ and $\mathcal{Z} = \{Z_1, \cdots, Z_C\}$, where Y_i are the subsets of $\mathcal{Y}$, and similarly for Z_i and $\mathcal{Z}$.

The *Rand index* can be used to quantitatively measure the amount of agreement between the two clusterings of the set S. Intuitively, the Rand index corresponds to the probability of pair-wise agreement between the $\mathcal{Y}$ and $\mathcal{Z}$, i.e. the probability that the assignment of any two items will be correct with respect to each other (in the same cluster, or in different clusters).

Formally, the Rand index is calculated as follows. Define the following counts:

- a – the number of pairs in S that are in the same set in $\mathcal{Y}$ and in the same set in $\mathcal{Z}$,
- b – the number of pairs in S that are in different sets in $\mathcal{Y}$ and in different sets in $\mathcal{Z}$,
- c – the number of pairs in S that are in the same set in $\mathcal{Y}$ and in different sets in $\mathcal{Z}$,
- d – the number of pairs in S that are in different sets in $\mathcal{Y}$ and in the same set in $\mathcal{Z}$,

then the Rand index is the fraction of pairwise agreements

$$Rand = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}}. \tag{9}$$

The Rand index can be efficiently calculated from a contingency table:

		Partition $\mathcal{Z}$				
	Class	Z_1	Z_2	$\dots$	Z_C	Sums
Partition $\mathcal{Y}$	Y_1	n_{11}	n_{12}	$\dots$	n_{1C}	r_1
	Y_2	n_{21}	n_{22}	$\dots$	n_{2C}	r_2
	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
	Y_R	n_{R1}	n_{R2}	$\dots$	n_{RC}	r_R
	Sums	c_1	c_2	$\dots$	c_C	n

Each entry n_{ij} is the number of items in S that are common to Y_i and Z_j, and c_j and r_i are the sums over the jth column and ith row, respectively. Using the contingency table, the number of agreements (the numerator in (9)) can be calculated as

$$a + b = \binom{n}{2} + \sum_{i=1}^{R}\sum_{j=1}^{C} n_{ij}^2 - \frac{1}{2}\left(\sum_{i=1}^{R} r_i^2 + \sum_{j=1}^{C} c_j^2\right). \tag{10}$$

(a) Problem 1: Use the Rand index to evaluate the your clusterings with the ground-truth clusters. Which algorithm performs best overall, and for each individual dataset? How do the results change with the hyperparameter value (K or h)? (e.g., make a plot)

(b) Problem 2: The ground-truth segmentations for the images are available in the "gtruth" directory of the zip file. The file names are "`<imgname>-<userid>.png`", where `<imgname>` is the image filename and `<userid>` is the ID of the human who marked the segmentation. In the segmentation image, each grayvalue corresponds to one segment. Note that the segment labels (gray-values) are evenly spread out between 0-255 so that the segmentations can be viewed (you may want to remap it back to 1-K). Use the Rand index to evaluate your segmentations. Overall, which algorithm performs best at segmentation? How do the results change with the hyperparameters (K or h)? (e.g., make a plot).

References for the Rand index are available here:

- W. M. Rand. "Objective criteria for the evaluation of clustering methods". *Journal of the American Statistical Association*, 66 (336): 846-850, 1971.
- Lawrence Hubert and Phipps Arabie. "Comparing partitions". *Journal of Classification*, 2 (1): 193-218, 1985.

.....

- **What to hand in** – You need to turn in the following things:

  1. Answers, plots, analysis, discussion, etc. for the above problems.
  2. Source code files.

  You must submit your programming assignment using the BlackBoard website. Go to "Assignments" ⇒ "Assignment submissions" ⇒ and create a journal entry.

- **Plagiarism** – You must implement each clustering algorithm using your own code. Do not use someone else's implementation of the clustering algorithms (including MATLAB's implementation). It is okay to use common library components and functions, e.g. standard matrix operations (e.g., `inv`, `eig`). *If you use any special toolboxes or libraries, make sure to note them in your report.*

- **Grading** – The marks for this assignment will be distributed as follows:

  – 20% – Implementation of the K-means, EM, and mean-shift algorithm (1a).
  – 20% – Results on toy datasets (1b).
  – 10% – Sensitivity to parameters (1c).
  – 20% – Results on image segmentation and sensitivity to parameters (2a).
  – 10% – Results using feature scaling (2b).
  – 20% – Quality of the written report. More points for insightful observations and analysis.

  Note: if you really cannot implement the algorithms correctly, it is okay to use 3rd party software. You will not get marks for implementation, but you can still get marks for presenting the results. *If you use a 3rd party implementation, you must acknowledge the source in your report.*

- **Optional Problems** – The optional problem 3 will not be graded.