

Homework Assignment 9: Clustering

Due Friday, April 21, 2023 at 11:59 pm EST

Description

In this assignment we are implementing the k-means clustering algorithm. You **MUST** implement your own clustering algorithm as directed in the question and not use any API clustering function.

What to submit

Create a folder: `ps9_xxxx_LastName_FirstName` and add in your solutions (xxxx = python or MATLAB):

`ps8_xxxx_LastName_FirstName/`

- `input/` - input images, videos or other data supplied with the problem set
- `output/` - directory containing output images and other files your code generates
- `ps9.m` or `ps8.py` - code for completing each part, esp. function calls.
- `*.m` or `*.py` Matlab/python function files, or any utility code.
- `ps9_report.pdf` - a PDF file with all output images and text responses

Zip it as `ps9_xxxx_LastName_FirstName.zip`, and submit on Canvas.

Guidelines

1. Include all the required images in the report to avoid penalty.
2. Include all the textual responses, outputs and data structure values (if asked) in the report.
3. Make sure you submit the correct (and working) version of the code.
4. Include your name and ID on the report.
5. **Comment your code appropriately.**
6. Please avoid late submission. Late submission is not acceptable.
7. Plagiarism is prohibited as outlined in the [Pitt Guidelines on Academic Integrity](#).
 - a. **Please don't share your codes with any of your colleagues.**

Questions

1. K-means clustering and image segmentation

In this problem, you will **implement** the K-means algorithm. You will then use it to perform image clustering (segmentation). **Please note that you are not allowed to use any K-means or segmentation APIs**, instead you will need to implement your own K-means function.

- a. Write a function `[ids, means, ssd] = kmeans_single(X, K, iters)` to implement a basic version of K-means.

Inputs:

- `X` is the $m \times n$ data matrix, where m is the number of samples and n is the dimensionality of each feature vector.
- `K` is the number of clusters to output.
- `iters` is the number of iterations to run for K-means (update centers and assign points)

Outputs:

- `ids` is an $m \times 1$ vector containing the data membership IDs (cluster id) of each sample (denoted by indices ranging from 1 to K, where K is the number of clusters).
- `means` is an $K \times n$ matrix containing the centers/means of each cluster. For example, the first row should be the n-dimensional vector that contains the center of cluster#1.
- `ssd` is a scalar that gives the final SSD of the clustering. Remember, SSD is the sum of the squared distances between points and their assigned means, summed over all clusters.

To randomly initialize the means: Get the range of the feature space, separately for each feature dimension (compute max and min and take the difference) and use this to request random numbers in that range. Check the documentation for `rand`.

Once you have your initial centers, iterate over the following two steps. The first step is to compute the memberships for each data sample. Matlab's function `pdist2` can help you to efficiently compute distances. Then for each sample, find the min distance and the cluster that gives this min distance. The second step is to recompute the cluster means, simply taking the average across samples assigned to that cluster, for each feature dimension. Repeat for `iters` times.

Hint: You can create a small dataset of your own to test your function. Two dimensional data will be easier to visualize and debug.

- b. K-means is sensitive to the random choice of initial centers. To improve your odds of getting a good clustering, implement a wrapper function `[ids, means, ssd] = kmeans_multiple(X, K, iters, R)` to do R random initializations/**restarts**, and return the clustering with the lowest SSD.

Inputs: (same as `kmeans_single`)

- R is a scalar denoting the number of restarts (different random assignments) to perform.

Outputs:

- Same as `kmeans_single`, **but** `ssd` is the lowest SSD across all random restarts.
- c. Now, you will apply clustering to segment and recolor four different images image.
- Download the attached images to your input directory and load them in your workspace using `imread` function. This will return a $H \times W \times 3$ matrix per image, where H and W denote height and width, and the image has three channels (R, G, B).
 - Convert the images to double format (use the function `im2double`). To avoid a long run of your code, downsample the images (reduce their size) e.g. using `im = imresize(im, [100 100]);`
 - To perform segmentation, you need a representation for every image pixel. We will use a three-dimensional feature representation for each pixel, consisting of the R, G and B values of each pixel. The command `X = reshape(im, H*W, 3);` helps to convert the 3D matrix (image) into a 2D matrix (data matrix) with pixels as the rows and channels (features) as the columns. Use the random restarts function (`kmeans_multiple`) you wrote above, to perform clustering over the pixels of the image. Note, you have to perform clustering on each image separately and independently from others.
 - Use your clustering output to recolor each pixel of each image according to their cluster membership. In particular, replace each pixel with the average R, G, B values for the cluster to which the pixel belongs (i.e. recolor using the cluster means). Show the recolored image using `imshow`, but convert it to format `uint8` before displaying.
 - For each image, experiment with different combinations for K , $iters$, and R . Please use the following values:
 - o $K = 3, 5, 7$
 - o $iters = 7, 13, 20$
 - o $R = 5, 15, 30$
 - **Hint1:** to streamline and modularize your implementation, you may need to implement the segmentation using a function:
`[im_out] = Segment_kmeans(im_in, K, iters, R)`
 - **Hint2:** You may want to test your code on small part of the images before letting it run over the entire image. This will help you to verify your implementation without wasting long execution time.
- d. Your report must document your observations, comments, and interpretation of the output. Include only a representative sample of your resultant images from all the combinations in your report. Save all the resultant images in the output folder.