CS171 - Introduction to Machine Learning and Data Mining Spring 2018 - Assignment 3

Instructor: Vagelis Papalexakis, University of California Riverside

In this assignment you will implement the most popular clustering algorithm (K-Means!) and test its performance on real data.

Question 0: Getting real data *[0%]*
In this assignment you are going to use the Iris dataset from Assignment 1: https://archive.ics.uci.edu/ml/datasets/Iris

Question 1: K-Means Clustering *[50%]*
In this question you are going to implement the K-means clustering algorithm (also known as Lloyd's Algorithm) as we saw it in class. In particular, you have to implement a function:
[cluster_assignments cluster_centroids] = k_means_cs171(x_input, K, init_centroids)
where the **inputs** are:

1. x_input = this is a (data point x feature) matrix. Each row contains a data point to be clustered.
2. K = this is the number of clusters
3. init_centroids = this is a (K x feature) matrix that contains the initializations for the K centroids. In this case, we are going to randomly initialize those centroids by selecting K data points from x_input uniformly at random and setting them as the initial centroids. The reason why we want this initialization to be an argument is because we may want to have a different initialization scheme in the future so this is a more flexible implementation. Before calling your k-means function, set init_centroids as described above and pass it as an input.

The outputs of the function are:

1. cluster_assignments = this is a (data point x 1) vector that contains the cluster number that each data point is assigned to.
2. cluster_centroids = this is a (K x feature) matrix that contains the final centroids that K-means computed.

For all distance calculations, use the L2 distance (aka Euclidean distance).
In order to make sure that your code is running, for K=3 run the algorithm once and report the sum of squares of errors between all data points and their respective centroid.

Question 2: Evaluation *[50%]*
Unlike the classification scenario in Assignment 2, in clustering we can't do cross-validation to determine the number of clusters and the quality of our clustering. Instead, what we are going to do is explore different clusterings and try to understand their quality indirectly. Furthermore, because our dataset fortunately has labels, we can use the labels to judge the homogeneity of the clusters we compute (in question 4, for extra credit). However, in the general case where we would apply clustering, we would not expect to have labeled data, so this step is more of a "bonus" and serves as a sanity check.

1) **Knee Plot**: As we saw in class, a good heuristic for judging the quality of K-means clustering and finding the number of clusters is the “knee plot”. The name of the plot comes from the “knee” in the line that we draw, indicating the point where the error stops improving dramatically, which, in turn, may indicate a good number of clusters. In this question you should generate the knee plot, i.e., the sum of squares of errors of all data points from their assigned cluster on the y-axis (i.e., the function that K-means is minimizing) as a function of the number of clusters K, for K taking values in `[1 2 3 4 5 6 7 8 9 10]` on the x-axis. Make the plot after running the algorithm once. At which K does the “knee” appear? Does it agree with the number of classes in the Iris dataset?
2) **Sensitivity analysis**: We are using K-means with random initialization. This randomness may result in unstable solutions (since we may be unlucky with some initialization and set up our algorithm for failure). In this question we are going to evaluate the sensitivity of our algorithm for the Iris dataset as we vary the number of clusters (as in sub-question 1). More specifically, repeat the knee plot of sub-question 1, but now, for each value of K you are going to run the algorithm for `max_iter` times and record the mean and standard deviation for the sum of squares of errors for a given K. Plot the new knee plot where now, instead of having a single point for each K, you are going to have a single point (the mean) with error-bars (defined by the standard deviation). If the error-bars are tight, this means that our estimates for the mean error have converged and are reliable. If they are not tight, this means that a) we either need more iterations, or b) our algorithm is unstable. Create 3 such knee plots for `max_iter = 2`, `max_iter = 10`, and `max_iter = 100`.

   Note: Make sure every time you run the algorithm with a random initialization, this random initialization is computed from scratch. If the same random initialization is used every time, this will result in deterministic behavior for the algorithm and will not let us study its sensitivity for different random seeds.

<u>Question 3: K-Means++ Initialization *[35%] - EXTRA CREDIT*</u>

As we saw in class, there are ways to improve the reliability of the K-Means algorithm initialization and make a more stable algorithm. One of those methods is the K-Means++ initialization algorithm which we saw in class. In this question you are going to implement K-Means ++ and evaluate its stability:

1. Implement a function init_centroids = kmeanspp(x_input, K) where x_input and K are the same as before, and init_centroids will be the initialization for the centroids as computed by the K-Means++ algorithm. This value for init_centroids will be used instead of the randomly chosen init_centroids in question 2.
2. Now, we are going to compare the sensitivity of initializing K-Means using the ++ algorithm versus the random initialization. Repeat the same plots for Question 2.3 (sensitivity analysis) but now use K-Means++ as the initialization. Do you observe any difference in the sensitivity of our algorithm? K-Means++ also has elements of randomness, so make sure every time you run it you use a different random seed,

<u>Question 4: Top data points per cluster [15%] - EXTRA CREDIT</u>

Write code to identify the top-3 data points for each one of the K clusters, for K = 3, for the version of the algorithm with the random initialization. You are going to determine the top data points by their proximity to the cluster centroid that they are assigned to. For a given cluster, find the class labels of the top-3 data points. How many are they of the same label?

PLEASE READ BELOW - IMPORTANT INFORMATION

Programming language: For this assignment you are going to use *Matlab* or *Python*. Unless noted otherwise, you may ***not*** use library functions that already implement the questions that you have to implement in this assignment. If you are using a non-standard library or function, you must cite it and explain why it is necessary for your implementation.

Deliverables:

1. A report in PDF format describing the approach taken in each question and containing the answers and figures required in each question.
2. Your code, organized by question, and properly commented.

Both deliverables described above should be uploaded as a single archive on iLearn.

Deadline: Please check the class website for the most current deadline.

Late policy: Please check the class website for the late policy.

Grade distribution: Unless noted otherwise, the number of points for each question are equally distributed to each sub-question.

Academic Integrity: Each assignment should be done *individually.* You may discuss general approaches with other students in the class, and ask questions to the TAs, but *you must only submit work that is yours*. If you receive help by any external sources (other than the TA and the instructor), you must properly credit those sources, and if the help is significant, the appropriate grade reduction will be applied. If you fail to do so, the instructor and the TAs are obligated to take the appropriate actions outlined at http://conduct.ucr.edu/policies/academicintegrity.html. Please read carefully the UCR academic integrity policies included in the link.