CMPT 459 Spring 2024
Data Mining
Instructor: Martin Ester
TAs: Arash Khoeini and
Ali Alimohammadi

[Total Marks:100]

The aim of this assignment is to implement K-Means, which is a representation-based clustering algorithm, using the Python programming language, and to test it on a single-cell RNA Sequencing (scRNA-seq) dataset.

scRNA-seq Analysis

scRNA-seq datasets are a result of recent new technologies for sequencing individual cells providing a higher resolution of cellular differences and a better understanding of the function of an individual cell. Each scRNA-seq dataset/experiment produces a matrix, where rows are cells and columns are genes. The elements $X_{i,j}$ of the matrix represent the expression of gene j in cell i, i.e. the number of copies of gene j observed in cell i. This matrix is sparse, meaning we do not know the value for most of the elements.

One important step in scRNA-seq analysis is clustering the cells based on their gene expression profiles, such that cells with similar gene expression are grouped together in the same cluster. These clusters correspond to cell-types, groups of cells performing the same biological function. This is a challenging clustering task due to the sparsity and high dimensionality of scRNA-seq datasets, but also paves the way for further analysis, such as diagnosis (i.e. whether the patient has cancer) and prognosis (i.e. whether the cancer will develop aggressively).

Dataset [download from here]

In this assignment you are provided with a human pancreas tissue dataset. This dataset contains 8469 pancreas cells with expression for 20,125 genes. Since this is a high dimensional dataset, you first need to reduce its dimensionality using the PCA algorithm. You are provided with a PCA implementation in *main.py.*

KMeans++

One weakness of the K-means algorithm is that it is sensitive to the initialization of the centroids. So, if a centroid is initialized to be a “far-off” point, it might just end up with no points associated with it, and at the same time, more than one cluster might end up being linked with a single centroid. Similarly, more than one centroid might be initialized into the same cluster resulting in a poor clustering.

To overcome the above-mentioned drawback, the K-means++ algorithm ensures a smarter initialization of the centroids and improves the quality of the clustering. Apart from the initialization, the rest of the algorithm is the same as the standard K-means algorithm. That is K-means++ is the standard K-means algorithm coupled with the following method of initialization of the centroids:

1. Randomly select the first centroid from the data points.
2. For each data point, compute its distance from the nearest, previously chosen centroid.
3. Select the next centroid from the data points such that the probability of choosing a point as a centroid is directly proportional to its distance from the nearest, previously chosen centroid. (i.e. the point having maximum distance from the nearest centroid is most likely to be selected next as a centroid, but other points may be selected with smaller probability).
4. Repeat steps 2 and 3 until k centroids have been sampled.

You are provided with a code template, which you will need to complete with your own implementation.

Make sure you have the latest versions of the following libraries installed:

- Numpy (i.e. pip install numpy)
- Sklearn (i.e. pip install scikit-learn)
- Anndata (i.e. pip install anndata)
- ScanPy (i.e. pip install scanpy)
- Matplotlib (i.e. pip install matplotlib)

Please note that you are allowed to use Sklearn **ONLY** for PCA computations.

The provided template includes two files:

- main.py : includes functions for reading the data, preprocessing, and dimensionality reduction. This file accepts two arguments: –n-clusters and –data
- kmeans.py: This file contains the KMeans class, which you need to implement for this assignment.

Tasks

1) Implement KMeans based on the provided code template. You need to complete the following methods in addition to any other of your own methods:
    a) *fit*: gets X as input, finds n_clusters clusters and returns cluster labels in a Numpy array [5 marks]
    b) *initialize_centroids*: Initializes cluster centroids using either random initialization or KMeans++ initialization. [10 marks]
    c) *update_centroids*: Updates centroids based on the current clustering [5 marks]
    d) *euclidean_distance*: Computes the euclidean distance between rows of two matrices. [5 marks]
    e) *silhouette*: Computes silhouette coefficient for the clustering. [5 marks]

(hint: This dataset is quite sizable. To minimize runtime, please leverage numpy vectorization extensively. Note that runtime will not impact your grade.)

2) Apply your implementation of KMeans to cluster the human pancreas tissue dataset. Use the provided PCA implementation to reduce the dimensionality to 100 first. Use random initialization to produce clusterings for k from 2 to 10 and plot the silhouette coefficient for all clusterings in your report file. What is the best k? [25 marks]
3) Produce clusterings for k from 2 to 10 using KMeans++ initialization. Report the silhouette coefficients again. What is the best k? What is your conclusion? [25 marks]
4) Use a scatter plot to visualize the clusters with the best k from task 2 and 3. You should reduce the dimensionality further to 2 in order to be able to visualize the clusters in a 2D plot (Note that this is only for visualization and you still need to perform KMeans using 100 features). Color-code the cells based on their clusters. Include both plots in your report file. [20 marks]

[IMPORTANT] You should submit a *[student-id].zip* file. The zip file should include the following:

- A report file: *report.pdf*
- Two Python file: *main.py and kmeans.py*

Running the main.py file using python>=3.8 with appropriate arguments should perform task 4 and reproduce the plot you reported for that task.

Deadline: 23:59 pm PST on March 12th.
You will lose 10% of the marks for submissions after this deadline, as long as it's not more than 24 hours late. You will lose all the marks for submissions after that.

Libraries: You can use: *matplotlib*, *anndata*, *numpy*, *pandas*, *sklearn.decomposition.PCA*.

You MUST provide YOUR OWN code for the KMeans algorithm and for all the tasks specified in this assignment. These MUST be implemented from scratch i.e. not using scikit-learn or other libraries. You will be marked on the correctness of your implementation.