Overall Learning Objectives:

After completion of the coding assignment, students should be able to:

1. Recognize which situations are most appropriate for spectral graph clustering. Specifically, when the data may not appear like clusters according to Euclidean distance, due to Euclidean distance usually creating clusters that look like circles.

2. Understand the importance of identifying and selecting the appropriate hyperparameter σ in the RBF and exponential kernels. Students should be able to reason about how modifying hyperparameters in the similarity functions affect the clustering result.

3. Implement the spectral clustering algorithm from scratch (with only access to NumPy functions), and understand how to use the built-in scikit-learn function for graph spectral clustering. Instead of treating spectral clustering and k-means++ as blackboxes, students should be able to understand enough of the algorithm to re-implement it in Python. Along with knowing the pseudocode of graph spectral clustering to the point of being able to re-write it in a programming language, students should be able to search up and read documentation of built-in functions (which are more frequently used in real world applications), and apply those clustering functions to different datasets.

4. Understand the initialization process of k-means++ and its advantages compared to vanilla k-means. Specifically, k-means++ guarantees a O(log k) approximation in expectation for the clustering optimization function, where k is the number of clusters.

5. Understand how spectral clustering fits into the overall exploratory data analysis process, including skills like plotting, visualizing data, data extraction, and data analysis. In a real world scenario, spectral clustering will just be one of many techniques employed during the data analysis lifecycle; students should be able to see the role of clustering within the context of a data problem.

6. Understand the importance of choosing k, the number of clusters, and how observing the eigenvalues of the Laplacian matrix can help with determining which value of k to use.

Assignment Outline:

Part 1: Data Generation

The section involves implementing code that generates data from Gaussian mixtures, as well as visualizing and playing around with this data generation in 1a (Objective 5). This toy dataset serves as an example of a dataset where clustering can be applied (Objective 1), and we use this data for analysis in later parts. Students are asked to interpret a 2D plot of the dataset in 1b and hypothesize the result of clustering as the distribution the data is generated from varies, reinforcing and connecting previous concepts in EE16ML to clustering.

Part 2: The Spectral Clustering Algorithm

This section of the notebook is the bulk of the assignment and asks students to implement k-means++ and the graph spectral clustering algorithm (Objective 3). We provide a rough outline and pseudocode for methods that need to be implemented for Spectral Clustering. These methods include the k-means algorithm and calculating a similarity and Laplacian matrix in questions 2c and 2d. Part of the k-means implementation in 2a includes initializing the centroids via k-means++ (Objective 4), compelling students to understand the difference between vanilla k-means and k-means++, as well as why k-means++ is used in spectral clustering. Students are encouraged to refer to notes/lecture in order to understand this material and properly implement the code. Most importantly, this section allows students to understand how these algorithms are actually implemented, instead of only calling the appropriate sklearn methods. However, that is an important skill as well, and is covered in questions 2b and 2f, since using sklearn will be the bulk of most "machine learning engineering" in the real world, so we allow students to practice this skill as well.

Part 3: Extracting and Visualizing Eigenvectors

Question 3a introduces students to visualizing entries of a matrix as colors, so that they can visualize patterns in the matrix more easily (Objective 5). Question 3b shows the importance of choosing the correct number of clusters k (Objective 6) and by seeing the eigenvalues of the similarity graph Laplacian, they can deduce values of k that work well for the dataset by observing the jumps. Finally, question 3c shows the relationship between the eigenvectors of the similarity graph Laplacian and the actual cluster assignments that graph spectral clustering finds, demonstrating the one of the purposes of graph spectral clustering to students.

Part 4: Classifying Harder Datasets

In this section, we move on to the two moons dataset, and more challenging dataset to cluster than the previous dataset, providing another example of when spectral graph clustering is suitable to use (Objective 1). Similar to the first Gaussian mixture dataset, students need to generate and plot this dataset in 4a. Students will see how this instance of k-means and even k-means++ does not cluster the two moons properly, but after choosing the right hyperparameter σ in the RBF kernel (Objective 2), graph spectral clustering properly clusters the dataset. Additionally, we provide question 4c for students to write their own code to visualize and plot the dataset, with a slider to play around with the hyperparameter σ and interpret its effect on the plot in 4d (Objective 5). From our personal experience, we have found it extremely important to understand how to visualize code, as it provides a quick way to check if our work is heading in the correct direction.