Pitfalls

Recommended Test Order

- K-means
- GMM

To check if an element is a member of a numpy array, use the keyword ‘in’:
Suppose A=[0,1,2,3], '1 in A' gives you True and '5 in A' gives you False.

K-means

Fit

Initialize random means (these will represent our clusters)

Update labels: (labels are just what cluster each point belongs to)

- Find Distances from each point to each mean, aka cluster (use euclidean distances, can import from scipy or from knn/collab filter)
- Update labels for each point based on what mean it is closest to (argmin is very useful here!)

Update means: (means are just from what point we are considering the location of each cluster)

- Find indices for the points for each label, aka cluster (can use np.where, logical indexing)
- Update the mean of each cluster based off the mean location of its points

Check if new means == old means to determine if we update again

Store means once done

Tip: Can use helper functions for 1. finding nearest clusters 2. updating means and 3. updating assignments

Predict

Find distances from features to stored means, [use argmin on distances to assign labels (if using KNN distances implementation of nxm array) along axis -1]

Can be done in one line if using helper functions

GMM

e-step

Use _posterior function to fill the prior probabilities for each gaussian. Each cluster represents a column in the matrix.

m-step

Update means, covariances, mixing weights for each cluster according to equations from slide 24 shown below (https://nucs349.github.io/lectures/eecs349_gaussian_mixture_models.pdf)

Update our parameters as follows...

$$\text{new } w_j = \frac{\Gamma_j}{N}$$

$$\text{new } \mu_j = \frac{\sum_{i=1}^{N} \gamma_{j,i} x_i}{\Gamma_j}$$

$$\text{new } \sigma_j^2 = \frac{\sum_{i=1}^{N} \gamma_{j,i} (x_i - \mu_j)^2}{\Gamma_j}$$

Here means = μ_j , mixing_weight = w_j , and the covariance matrix = σ_j^2 .

You will also have to calculate $\gamma_{j,i}$ which is the corresponding column in assignment for each cluster. (use np.newaxis or reshape to convert 1-D to 2-D arrays)

These slides are also great if you want to understand what expectation maximization for GMM is actually doing!

log-likelihood
Calculate logpdf with the indicated function imported from scipy and add it to the log of mixing weights, this should be 1-2 lines

FRQ
If your implementations for the 2 clustering algorithms are too slow to run for the free response, you may import the sklearn implementations and use those instead.

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html

Once you've imported them, you can simply declare a new KMeans or GM object and use the fit() and predict() functions. Specifics on usage are detailed on the page.