Lab 4

Due: Friday 10/06/2023 @ 11:59pm EST

The purpose of labs is to give you some hands on experience programming the things we've talked about in lecture. This lab will focus on learning (i.e. "fitting") using "hard" expectation maximization through the KMeans clustering algorithm. You will segment each point into its closest cluster in the `estep` method and recalculate each cluster in the `mstep` method.

Task 0: Setup

Included with this file is a new file called `requirements.txt`. This file is rather special in Python: it is the convention used to communicate dependencies that your code depends on. For instance, if you open this file, you will see entries for `numpy` and `scikit-learn`: two pythn packages we will need in order to run the code in this lab. You can download and install these python packages through `pip`, which is python's package manager. While there are many ways of invoking `pip`, I like to do the following:

```
python3 -m pip install -r requirements.txt
```

(I let `python3` figure out which `pip` is attached to it rather than the more common usage: `pip install -r requirements.txt`)

Task 1: `class KMeans` (100 points)

In the file `kmeans.py`, you will find a class called `KMeans`. This class has a few methods in it, but I want you to pay attention to two of them in particular: `estep` and `mstep`. These method implement the E-step and M-step of an EM iteration respectively, and it is what you will need to finish in this lab. Your code should work for an arbitrary number of clusters as well as with data of arbitrary dimensionality. The number of clusters is specified at object construction time and is stored in a field called `self.num_clusters`. Please implement the following:

- `estep`. This method takes in a parameter `X`, which is a numpy array storing a column vector. It should have shape of (*num_examples*, d), where *num_examples* is the number of data points in the dataset, and `d` is the dimensionality of each point. Your E-step method should calculate and return the index of the closest cluster to each point inside a column vector. This means that the output should have shape `(num_examples, 1)` and contain integer values ≥ 0. In KMeans we measure the distance between a point $\vec{x_i}$ and a cluster center $\vec{c_j}$ using square euclidean distance:

$$d(\vec{x_i}\vec{c_j}) = \sum_{m=1}^{d} (\vec{x_i}_m - \vec{c_j}_m)^2$$

  where d is the dimensionality of every point (and also the dimensionality of every cluster center). The index of the closest cluster to point $\vec{x_i}$ can be calculated using the following equation:

$$j^* = \arg\min_{1 \leq j \leq k} d(\vec{x_i}, \vec{c_j})$$

  Note: If there are ties for the closest cluster, please choose the cluster with the smallest index. For example, if clusters 3, 8, 13, and 12 all tie for being the closest to point $\vec{x_i}$, please assign $\vec{x_i}$ to cluster 3 since it has the smallest cluster index.

- `mstep`. This method takes two arguments: `X` and `cluster_idx_assignments`. `X` contains the data matrix just like in E-step, while `cluster_idx_assignments` contains the column vector that is the output of the E-step method. In the M-step method you will need to recalculate the cluster centers using the following equation:

$$\vec{c_j}^* = \frac{1}{|c_j|} \sum_{\vec{x} \in c_j} \vec{x}$$

  This equation says that the new cluster center is the mean of the points assigned to that cluster in the E-step.

  Note: If a cluster is not assigned any points to it in the E-step (this can happen!), the cluster does not change (i.e. the cluster center does not change).
  Note: when we take the mean of a bunch of vectors, we handle each component separately. This means that the mean is a new vector where the x component of the mean vector is the mean of the input vectors, the y component of the mean vector is the mean of the y components of the input vectors, etc.

Feel free to run the `kmeans.py` file, there is some basic testing at the bottom of the file that will be run if you run the file. I would recommend adding some of your own testing (what shape is what, etc) to increase your confidence in the correctness of your solution before you submit to the autograder. The autograder will take a few minutes to run!

Task 2: Submit Your Lab

To complete your lab, please **only turn in the `kmeans.py` file** on Gradescope. You shouldn't have to worry about zipping it up or anything, just drag and drop it in.