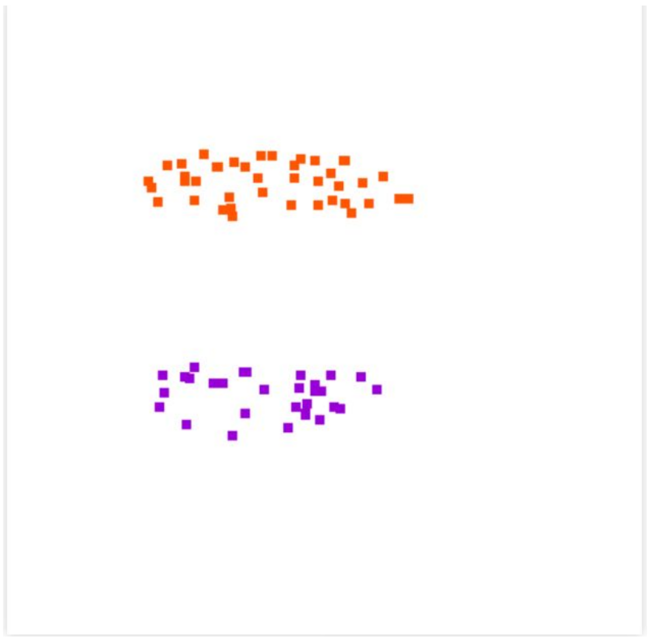
Question 1-a
What happens as beta in soft K-Means approaches infinity? How does this relate to regular K-Means?

Answer:
As beta approaches infinity, gamma goes to one. Then, soft K-means will become a regular K-means setup.

Question 1-b
Draw a data distribution using ml-playground that would be better modeled using a GMM x than with soft K-Means. Explain why this is so. Include the plot you made in ML playground. (Hint: think about the covariance matrix)

Answer:
The most significant difference between GMM and Soft K-means is that each cluster at GMM may have different covariance along each dimension, while soft K-means, however, has circular exponential distributions for each cluster (similar to GMM with spherical covariances, in which the "covariance" along each dimension is the same.)



In GMM, we label a point based on both the distance of the point to a cluster mean, and its likelihood that it belongs to the cluster using Gaussian distribution. In soft-Kmm, the distribution is exponential, instead of Gaussian, and it does not consider the weight of each Distribution in its posterior.

Question 2
Clustering handwritten digits (3 points, total)

In this problem, we will attempt to cluster handwritten digits contained in the MNIST dataset. Because we do not use the labels themselves during training **of KMeans or GMMs** (i.e., we are performing **unsupervised learning**), we can measure our performance directly on the dataset we used to train. For that reason, we will work only with the **test partition** of the MNIST dataset on this portion of the assignment.

Download the test partition of the MNIST dataset from the following links:

images
labels
Use the data loader found in **code/mnist.py** to load the MNIST test images and labels.

Rather than training on the entire test partition, we will use a class-balanced partition of the test set (i.e., we will use **an equal number of examples** from each class). **Find the digit that has the fewest examples in the MNIST test dataset**. Let the number of examples of this digit in the MNIST test dataset be n. Randomly sample n examples from each digit in the MNIST test dataset without replacement. This will be the subset of examples that we will use in our experiments.

Now you will test your clustering algorithms on this class-balanced partition. Each image in MNIST has dimensionality 28x28. **Flatten this representation such that each image is mapped to a 784-dimensional vector (28*28)**, where each element of the vector is the intensity of the corresponding pixel in the image. **Cluster these vectors into 10 clusters** (i.e., n_clusters=10) using the following algorithms:

KMeans
Gaussian Mixture Model
NOTE: IF YOUR IMPLEMENTATION OF GMM/KMEANS IS TOO SLOW FOR THESE EXPERIMENTS (OR YOUR IMPLEMENTATION DOESN'T WORK), YOU MAY USE THE IMPLEMENTATION CONTAINED IN SCIKIT-LEARN TO SOLVE THE FREE RESPONSE:

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

1. Comparing approaches without labels (1 point)

Report the performance of each of these algorithms using the Adjusted Mutual Information Score (implemented in code/metrics.py for you). Which algorithm performed best?

Answer:
Diagonal GMM: 0.345477880770952
Spherical GMM: 0.5255048007369354
Kmeans: 0.5087613459187998

Therefore Spherical GMM is the best

2.

Comparing approaches with labels (1 point)
Since we actually _do_ know the labels of the handwritten digits, we can also consider the accuracy of these unsupervised approaches.
For each cluster, find the most common label of all examples in that cluster. Call that the label of the cluster.
Find the proportion of images whose label matches their cluster label. That's the accuracy.
Report the performance of each of these algorithms using this measure. Which algorithm performed best? Is this the same one that did best with Adjusted Mutual Information Score?

Answer:
(Diagonal GMM score: ', 0.47197309417040356)
('spherical GMM score: ', 0.6076233183856502)
('Kmeans GMM score: ', 0.5854260089686099)
Spherical GMM is still the best.

3. Visualization (1 point)

Visualizing examples within each cluster can help us understand what our clustering algorithms have learned. Here are two ways that a cluster can be visualized:

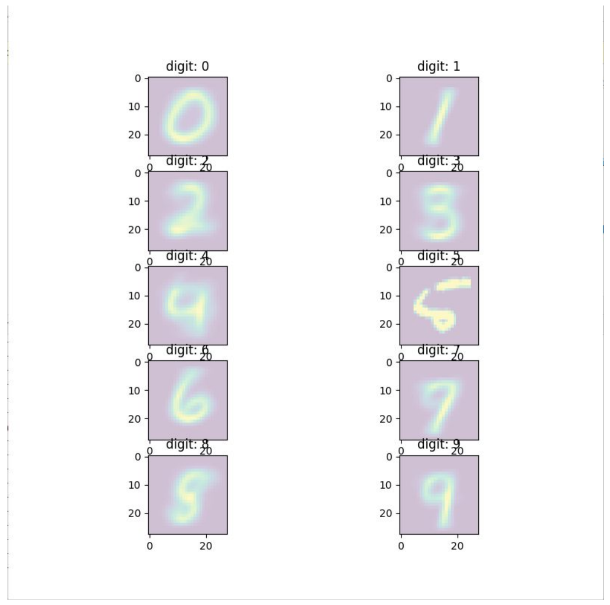
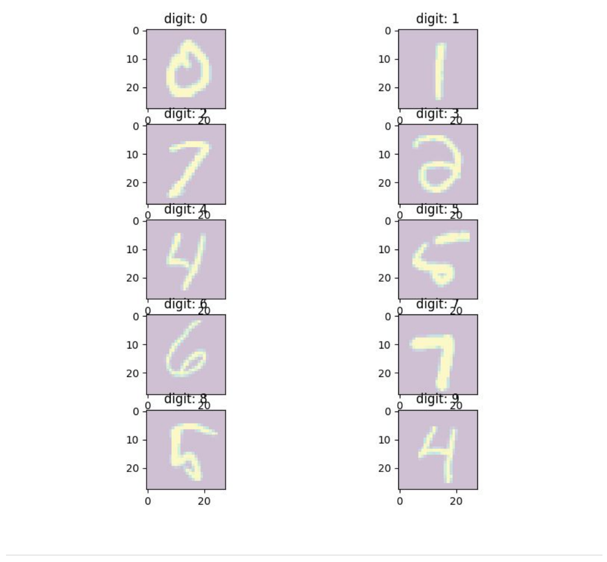
Find the mean of all examples belonging to a cluster.
Find the mean of all examples belonging to a cluster, then find the nearest example to the mean (i.e., the nearest neighbor).
For the **best performing** algorithm according to Adjusted Mutual Information Score, perform both of these visualization techniques on all 10 clusters. Show us the results of the visualization.

Note that you will have to reshape your features to 28x28 images to see the results. Use euclidean distance to determine the nearest neighbor.

What **visual differences** do you notice between the images generated from each of the visualization techniques? What **artifacts** do you notice in first technique? What do you think is the causing these artifacts?

Answer:

The best performing algorithm is spherical GMM.
Below are 1. Visualization of cluster means and 2. Visualization of the closest point to the mean by euclidean distances.

Note: before calculating the cluster means, I initialized each mean with a randomly selected image to prevent the situation where the mode of two clusters are the same.

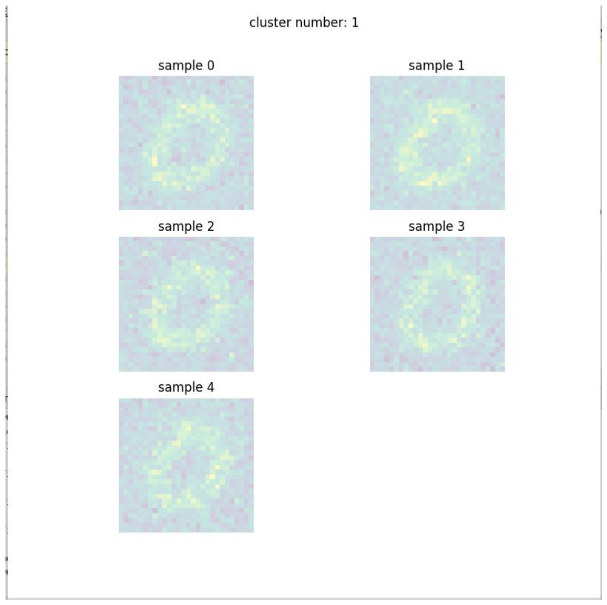
The major visual differences include:

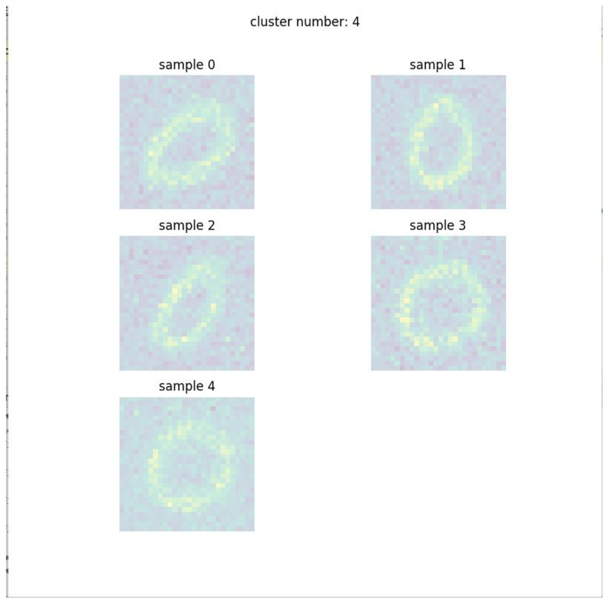
1. The image generated by the cluster means are always blurred, and there are very blurred edges that are not supposed to appear.
2. The image generated by the closest neighbor has false images in them. The reason for that is the mean of the image might be very close to an image that is actually not the digit.

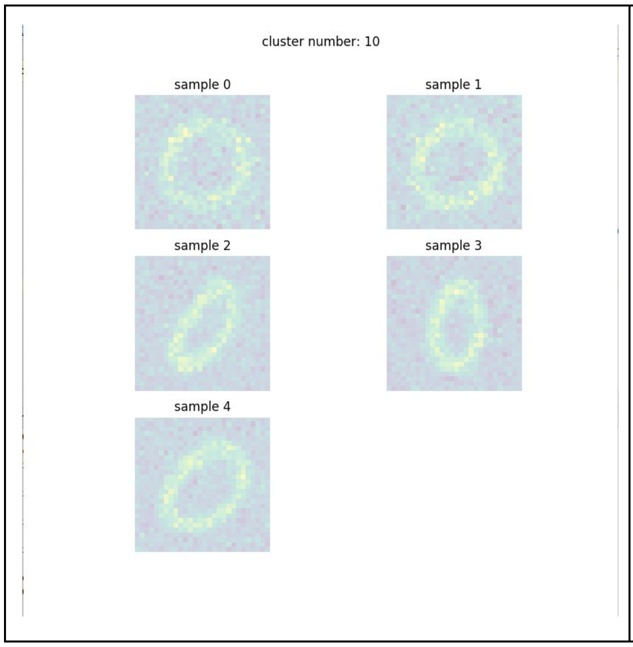
The major artifact I noticed in the cluster means images is that the edges are blurred, and that is due to the mixing of other misclassified images.

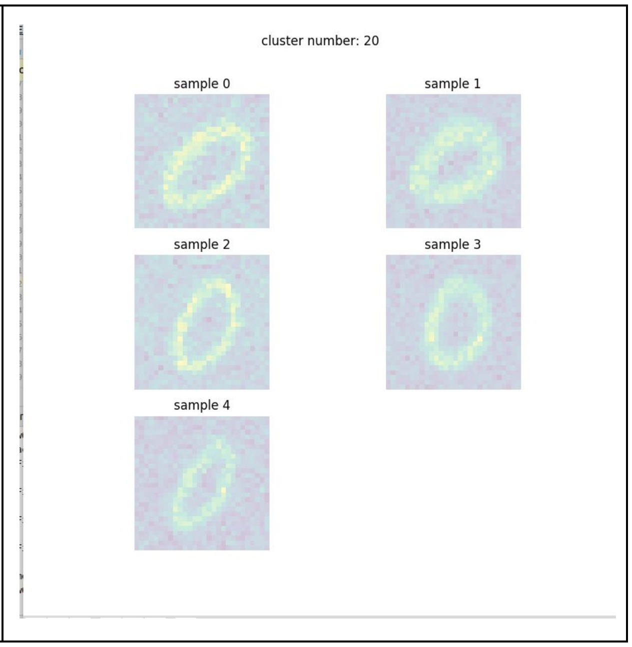
Bonus
A Gaussian Mixture Model is **a generative model**, meaning that it can not only cluster points but can also generate new points. Pick a digit class from the testing set of MNIST (e.g., 5) and fit a Gaussian Mixture Model to all examples from that class. Train 4 GMMs with the following values of n_components: 1, 4, 10, and 20 (Note: a component is another term for a cluster). For each trained GMM, sample 5 images using the GMM.sample function and show them to us. How does the number of components affect the quality of the sampled images?

Answer:









The more components, the better the image quality. That is because as the number components increases, each cluster's images are more and more similar. Therefore, the covariance of each cluster is smaller and smaller, and our random sample's pixel intensities tend to the mean.