Group Members: Ingrid Altamirano, Janty Sphabmixay, Jonathan Tran

Introduction

Our goal for this project was to use mathematical algorithms we learned in class and implement them into various methods in order to create a cluster plot for the MNIST dataset. We are trying to see if we take the numerical values of the handwritten digits in order to create a plot. This will allow us to have a visual representation of the handwritten digits while keeping the numerical dataset of MNIST. The MNIST dataset consisted of images of handwritten digits and in order for those digits to be illustrated, we had to implement various methods in order to achieve our end result. These various models are described in the three subsections below.

Methods

In section 1, our goal was to fit a single multivariate Bernoulli distribution into the dataset and have theta be the mean of each pixel across all of the images. In order for us to do so, we had to obtain the maximum likelihood estimation (MLE) of theta for MNIST. We completed this task by creating variables `N` and `D` and used `np.shape` in order to determine the length of each dimension of the array. In order to find the total sum, we made a for loop that calculated the means of `N` and `D`. After determining the total sum we divided it over `N` and then were able to make theta the mean of each pixel by using `np.array` in order to convert the input of `means` to an array.

In section 2, we applied KMeans by categorizing each image into a specific cluster by using the functions `calcSqDistances`, `determineRnk`, `recalcMus`, and `runKMeans`. In our `runKMeans` function, we allocated space for the vectors in Kmus by using `np.zeros` with parameters `K` and `D` in order to return a new array with the same shape. We then initialized the cluster centers by using `np.random.permutation(N)` to randomly pick points from the dataset. In order to specify the maximum number of iterations allowed, we created a `for` loop to assign each data vector to the closest mu vector. We first started off by calculating the squared distance matrix (`sqDmat`) by implementing `calcSqDistances` with parameters `X` and `Kmus`. From there we were able to determine the closest cluster center for each data vector and were able to create a responsibility matrix (`RnK`) that will be an N-by-K matrix of binary values. In order to do so, we multiplied `determineRnK` by `sqDmat` and stored it in `RnK`. Then, we recalculated mu values contingent on the cluster assignments. Lastly, in order to check to see if the cluster centers have converged, we created an `if` statement that determined the absolute values of our old mu subtracted from our recalculated mu. Which will then be subject to a `break` if the clusters have converged. This process allowed us to roughly see the digits represented within each cluster when using the `get_cluster_plot` function.

In section 3, our objective was to build a mixture model for our given dataset. We used the `train_EM_MOD` function in order to initialize our gamma and pi values. In order to run our function 1,000 times, we created a `for` loop in order to create a matrix with the shape of 20,000 by 784 and called this `matrix1`. We then created a `matrix2` in order to implement the equation $\theta^{x}(1 - \theta)^{(1-x)}$ which then allows use to multiply the result of `maxtrix1` by `matrix2` and store it into a variable called `matrix`. From there we set `gamma` to the `matrix` and implemented `pi` in order to run a `for` loop which initialized `theta_k` in order to reshape the results to the same shape in order for `pi` and `theta` to run the full dataset. All three sections allowed us to create clusters in order to decipher and plot the dataset of MNIST.

Results

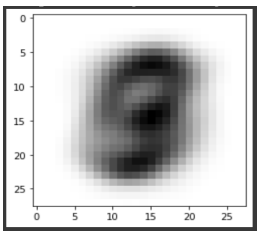


Figure 1. Resulting Figure for the implementation of a Multi-Dimensional Bernoulli Random Variable

1. Single-Bernoulli (**Figure 1)**

Our results after fitting a single Multivariate Bernoulli distribution on the given MNIST data set resulted in a blurry image of a number. To fit this data we used a Maximum Likelihood Estimate of the parameter of the Multivariate Bernoulli Distribution. This leads us to get a blurry image of the data as it isn't the most intuitive way to learn the most information about the data.

2. K-Means Clustering (**Figure 2 & 3)**

Our results after applying K-Means to the MNIST data set for the purpose of clustering the images are shown in Figure 2 and Figure 3. Figure 2 demonstrates K-means with 10 clusters and Figure 3 demonstrates the K-means with 20 clusters. We can see that at K = 10 clusters each image is a bit blurrier and the centers of each cluster are different in comparison to K = 20 clusters. When looking at K = 20 we see that the images are a bit easier to read but we continue to see the cluster center range in location meaning that this distribution does not model our data appropriately for showing a single digit.

Figure 2. Resulting Figure for implementation of KMeans on K = 10 Clusters each graph labeled with it's cluster number.

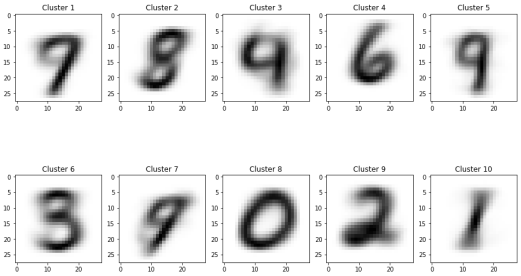
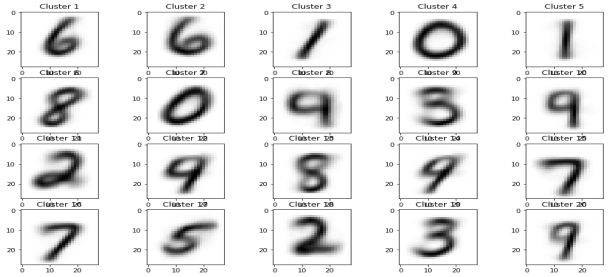


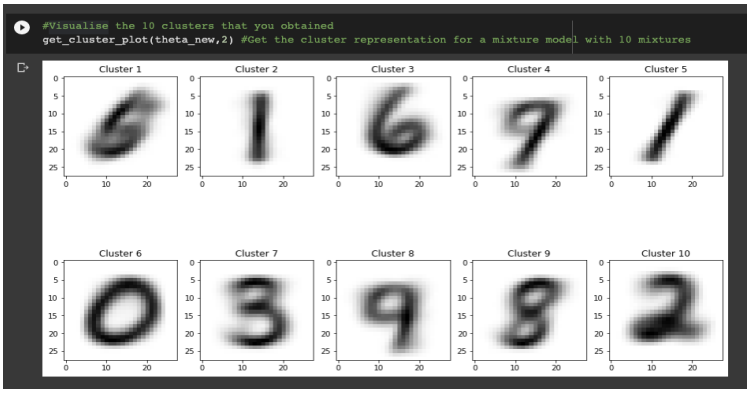
Figure 3. Resulting Figure for implementation of KMeans on K = 20 Clusters, each graph labeled with it's cluster number.



3. Fitting a Mixture of Multivariate Bernoulli Models (**Figure 4**)

Our results after implementing a Mixture of Bernoulli Models using EM into the MNIST data set are shown on Figure 4. By using a mixture model of 10 mixtures the data is able to find these clusters and apply them appropriately to each digit. In comparison to using KMeans and Single-Bernoulli, which only samples a single distribution, we see the cluster centers and clusters themselves do not sample the single digit accurately.

Figure 4. Resulting Figure for implementation of fitting a Mixture of Bernoulli Models for 10 mixtures



Discussion

What we found from this project was:

- The resulting figure after fitting a single Multivariate Bernoulli distribution on the given MNIST data set was blurry. It seems to represent multiple different numbers.
- With K-Means Clustering, we found that using K=10 gives a figure where each image is a little bit blurrier than the images presented from K=20. The images with K=20 are a little bit more clear, but we are still unable to see a single digit.
- With fitting a Mixture of Multivariate Bernoulli Models, we still saw each image showing a blurred representation of a number.

From the results of these three methods, we can say that K-Means Clustering or fitting a Mixture of Multivariate Bernoulli Models would be more appropriate than fitting a single Multivariate Bernoulli distribution since they provide a more accurate representation of the image of handwritten digits. Some things we learned about this project is that a lot of these “bigger” unsupervised learning algorithms make use of smaller algorithms. This means having a good understanding of “smaller and simpler” algorithms helps with creating and using more complex ones. For example, from this project, we saw that fitting a single Multivariate Bernoulli distribution did not give good results so we fitted a Mixture of Multivariate Bernoulli Models that gave a more accurate one. Some difficulties we encountered were long runtimes and vectorizing. It was difficult seeing if we got the correct results since the K-Means took about 2 minutes to complete and Mixture of Multivariate Bernoulli Models took about 10 minutes; therefore, whenever we changed a piece of code, we had to wait and see if we should change any other code. Since we are dealing with a large dataset, using nested for-loops would take a significant amount of time, so we needed to vectorize the code to shorten the run time. We were not too familiar with vectorizing so we had to look at discussions and office hours recordings to grasp how to use them and to familiarize ourselves with the methods.