COGS 118B Midterm Project

Brandon Rocchio, Daniel Milton, Isabella Gonzalez

February 11, 2022

Introduction:

The MNIST data set is a data set that contains 60,000 small squared pixel grayscale images of handwritten digits between 0 and 9 (Brownlee, 2019). The goal of this project was to be able to use some unsupervised learning techniques from the first half of the class in order to identify these digits. From class, we learned that unsupervised machine learning consists of using data with no labels to create a class as an output. For the project we use three different algorithms in attempts to classify the digit images: Single Multivariate Bernoulli Distribution, Kmeans Clustering, and a Mixture of Multivariate Bernoulli Models

Methods:

Single Multivariate Bernoulli Distribution

The data was transformed into input vectors of dimension 748 in order to be more easily used, and our output would be the classification of the images of the digits. In the first section, we fitted a single multivariate Bernoulli Distribution to the data. We fit a Bernoulli distribution model because each pixel can have a value 0 or 1. And then obtained a Maximum Likelihood Estimate of the parameter for our data. Maximum Likelihood Estimation is a method used for estimating the parameters of a distribution given some observed data (Lecture 2.1). The parameter of interest is theta which we want to be the mean of each pixel throughout every single image. We start with N = 20000 which is the number of all the images we're working with, and then we create an array of 784 zeros. Then we loop through every image and every pixel, and the function takes every pixel and adds it to the corresponding place. Finally, we create a new list of length 784 with the means, turn it into an array and assign it to theta MLE.

Kmeans

In the second part, we use k-means clustering in order to classify the images of the digits. K means partition the data into k number of clusters, assigning each data point to the cluster with the nearest mean (Lecture). This algorithm is split in between assigning points to specific clusters and then recalculating the means for each cluster. The first helper function calcsqdistances assigns the data points to a cluster so that the sum of the squared distance between the data points and the center of the cluster is at its minimum, the n,k entry should contain the squared distance from the nth data vector to the kth mu vector. The second helper function determinernk determines the rank of the matrix of squared distances. Rnk should be an N-by-K matrix of values 1 or 0 whose n,k entry is set as 1 if point n is closest to cluster k, 0 if otherwise. recalcMus simply recalculates mu values based on the cluster assignments. The main function runkmeans starts with assigning the data to variable X, representing the N observations of dimension D, then initializes Kmus as a D dimensional vector with k cluster assignments, and lastly it runs 1000 iterations of the three helper functions in order until cluster centers converge and then returns kmus.

Mixture of Multivariate Bernoulli Models

In the third part, we used a mixture of multivariate bernoulli models to classify images into digits. We used 3 helper functions along with a main function to run an expectation maximization iteration 100 times. The numerator function is used to calculate the numerator portion of the E step. The calc response function then calculates the denominator and returns the new E step. We then use this E step and the recalcParams function to find the new parameters of the prior and theta. We run these functions over and over 100 times to get a new estimate of the data and predict the images.

Results:

Section 1:
Reference Figure 1

The goal of this section of the project was to find the MLE of theta which is a model showing the mean of each pixel in the dataset. The figure shown above tells us that when looking at the mean of every pixel in the data set, you would get an image like this. Looking at the image you can see the numbers 1 through 9 in it but they all have some overlap. This is due to the fact that we found the mean of every pixel in the data set which is practically the same as saying we averaged the pixels of every number.

Section 2:
Reference Figure 2 and Figure 3

The first set of images here shows the results of K means clustering when using 10 clusters, the latter shows the results when using 20 clusters. The way that these images are clustered together is by comparing the similarity of which pixels are “on” and which are “off”. In the model, there is a 784x1 array that represents the pixels in the images, if the value is 1, that would mean that the pixel is “on,” if it is 0 then the pixel would be “off”. When we only use 10 clusters, the problem comes up that not all digits 1 through 9 come up as a cluster, as we increase the number of clusters, the higher chance that each digit will be represented by its own cluster.
Section 3:
Reference figure 4

The image above shows the results when using the expectation maximization algorithm on the data MNIST data set. In comparison to the other two sections of this project, the clusters that were formed with this algorithm are the most accurate. You can see the numbers in these images much more clear than the other two.

Discussion

We found that using an expectation maximization algorithm on classifying images into the numbers to be the best algorithm of the three. We learned that there are many ways to predict an image and each way has their pros and cons, depending on what outcome we want we choose a different algorithm. Another piece of information that we learned while working on this project was that these algorithms can be coded in several different ways and return nearly the same output. We could have easily used nested for loops when we were coding the EM algorithm but we opted to use helper functions instead to avoid the shortcomings of nested for loops. EM took the longest to run but gave us the best results. Some issues we ran into this project were mostly translating equations into code. To further explore this data set we thought about using other types of unsupervised algorithms and comparing would be helpful.

References:

https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-digit-classification/#:~:text=The%20MNIST%20dataset%20is%20an,digits%20between%200%20and%209

Figures:

Figure 1:

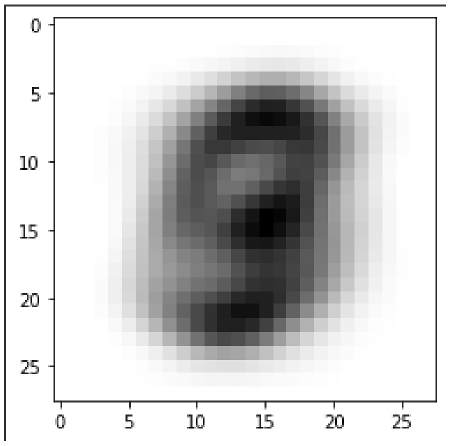


Figure 2:

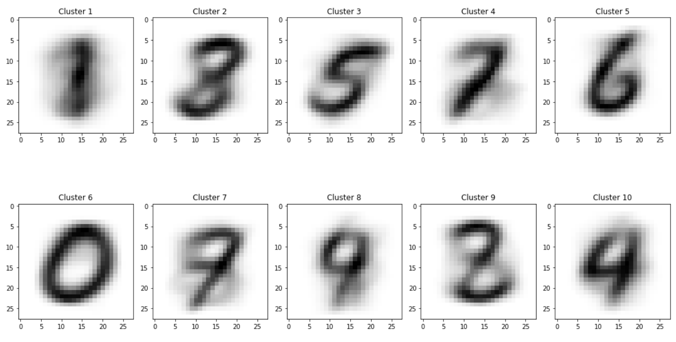


Figure 3

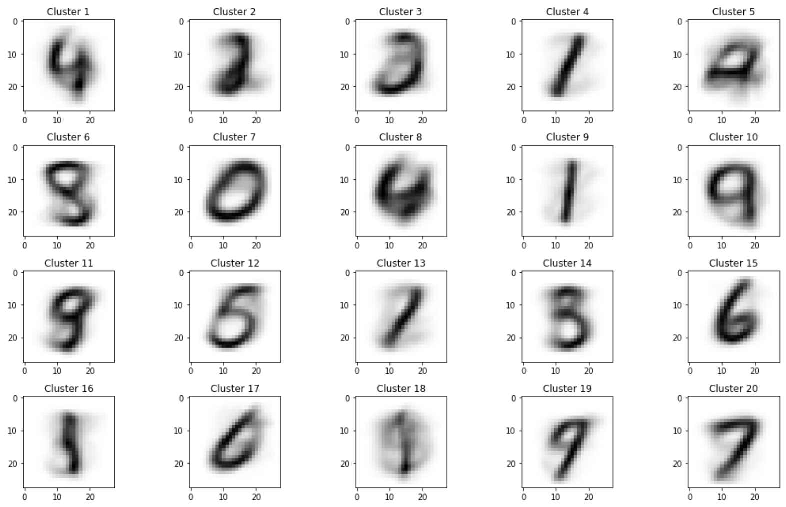
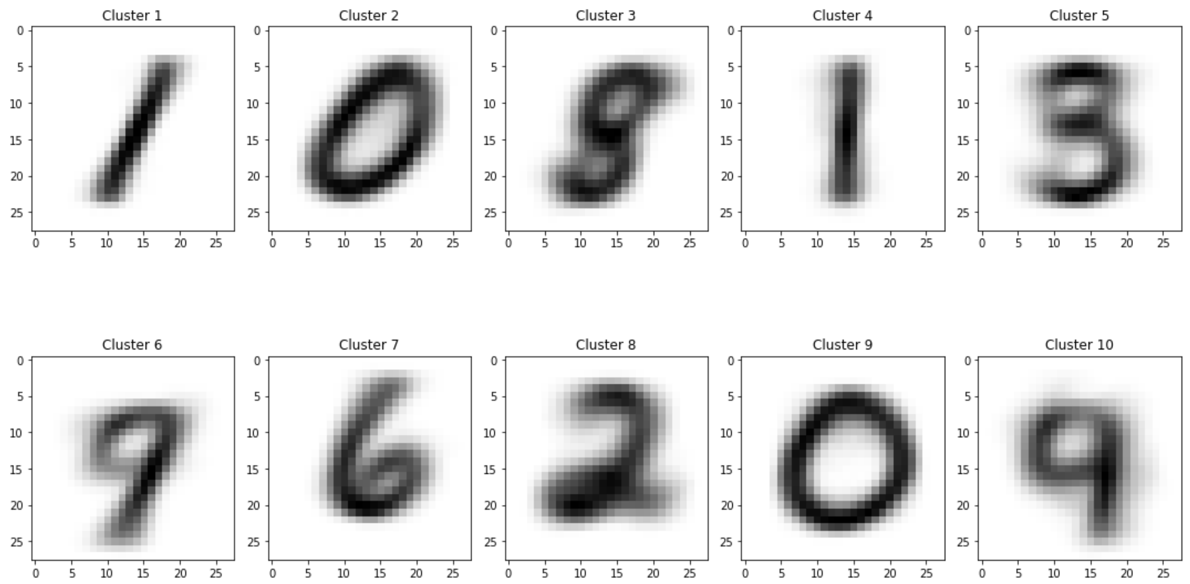


Figure 4:


Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8
Cluster 9
Cluster 10
0
5
10
15
20
25