Kshitiz Rimal

E-mail: krimal@masonlive.gmu.edu

G#: CS482-001

Miner Username: DataMining

Score: 51%

Current Rank at the time of writing this paper: 221

20 November 2020

- **Objectives:**
  - Think about Best Metrics for Evaluating Clustering Solutions
  - Deal with Image data (processed and stored in vector format)
  - Implement the K-Means Algorithm

- **Language Used:**
  - The programming language used for this project is python. Python has variety of in-built functions which you will not find in any other programming languages. The language is best fit for anyone who is working on machine learning and data mining field. Another reason for choosing this language is because of Python's built-in libraries like numpy and sklearn helped me to compile code faster and more efficiently. Using sklean's pairwise_distances function I was able to compute distance between clusters.

- **Implementation:**

  Method 1: Preprocessing

  - I started by reading the image data. Stored the data into a list (InputData). I appended each line by line into list using ". sprit()" which will remove unnecessary things like space and "split()" which splits data where commas are given in the file.
  - The reason for using list is because it is easy to append data and there isn't much hassle while appending the data. I could've used array to begin with, but array requires a lot of manipulation while adding data, so I chose list instead of array to append the raw data from file.
  - Afterwards, I figured to convert list to array because converting data to array opened a lot of possibility for data manipulation. I used numpy's "np.array()" function to convert list to multi-dimensional array. Then, I also made a variable

that stores total number of data using “arrayData.shape[]” function which will be used later while implementing k-means algorithm.

Method 2: Implementing k-means algorithm

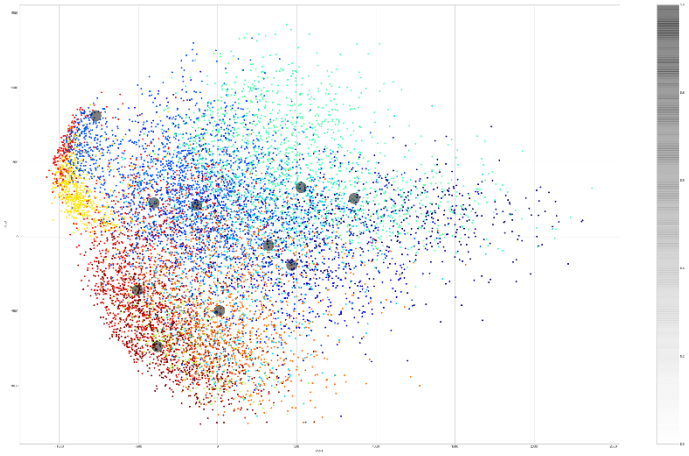
- Since, part1 and part2 of the assignment is related, I have used similar algorithm to compute both data. There are few changes than cannot be ignored. Since part2 (Image data) consists of 784 comma-delimited attributes which is fattened list of 28x28 matrix of integers for each digit there were few additional changes that needed to be changed. I used “np.random.choice” to create 10 different random data points as in homework part1.

- Once ten random data points were made, I followed the same flow as part1 and computed minimum distance from data points to its respective cluster. This time the image data was huge, and we had to each matrix to 1x784 vector. I used sklean’s “pairwise_distance” function to compute minimum distance by using cosine metric. I did not try with Euclidian metric this time because I already knew that my score would be bad if I use that metric. The “argim” function assigned minimum index from respective cluster. It ranges from 0 to 9 (k = 10).

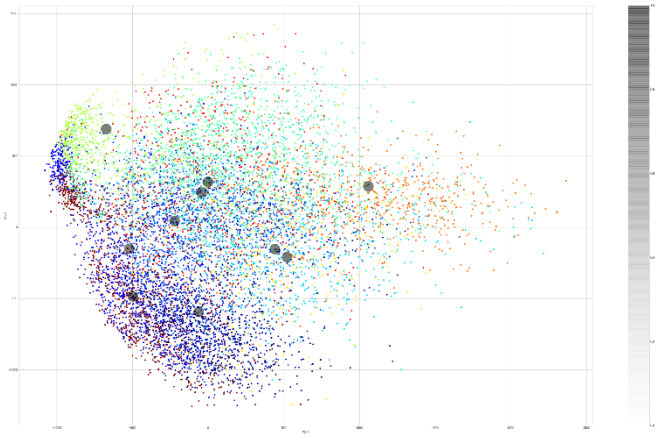
- I calculated mean of respective centroids. I used numpy’s “mean ()” function to find the minimum cluster in an array. I then appended it to a list. Now, the data comes into 1x784 vector space which is difficult to compute. So, for plotting purposes I converted list back to array with “np.array” function.

Method 3: Plotting Clusters

- I used matplotlib.pyplot library to observe the cluster as in part1. Since, the dimensionality is 784D I used sklearn.decomposition library to reduce dimension. This reduced 784D to 2D data which gave me a plot from matplotlib.pyplot library. Next, I observed scatter plot to determine how data looked and I was able to predict the score by looking at its scatter plot.

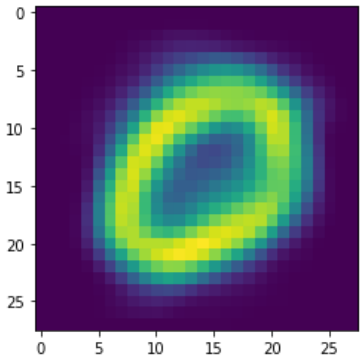
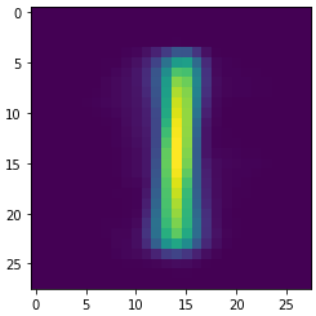


As you can see in this scattered plot the center points are spread out. Ten center points are sperated far apart from each other. This can tell us whether the data gives good score or bad score. This data plot gave me 50% on miner. So, I picked the spread-out plot to upload on miner.

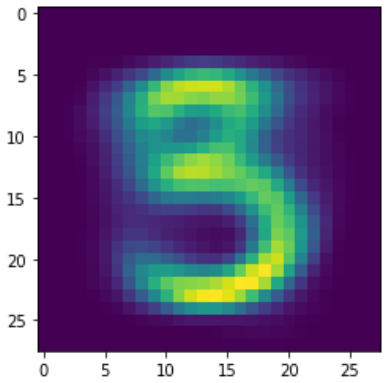
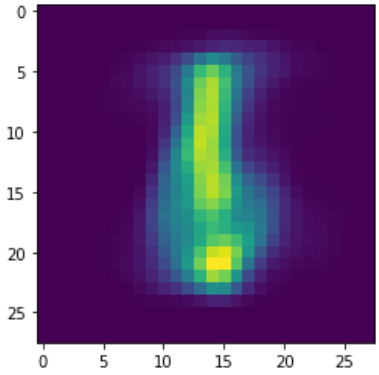


As you can see in this scattered plot the center points are clustered or very close to each other. This usually represents that scatter plot is bad and it will not give us good score.

- One major issue in my code is not being able to score more than 51% in part2. After further evaluation I figured out that k-means algorithm works best for circular form because centroids goes around surrounding area to get data, so any circular images were clear and non-circular images were unclear.

- These data were clear to cluster because of circular form. For example, 0 is in circular form so it was able to detect that image.

- Whereas, these numbers were not detected by algorithm because these numbers are not circular, so giving a clear image was difficult. There are better algorithms which can give better results than k-mean such as DBSCAN, but it is not the scope of this project.

Method 4: Output

- After every step was done from prepressing, implementing k means algorithm and Designing cluster it is time to print the values into a text file. As stated in part1's Method 4, I felt storing data into "txt" file. I followed the same step as Method 4 of part1 and added 10,000 data into a text file. My highest score in Miner is 51% and the reasons for the score is also mentioned above.