HW 3 – Image Clustering

“For this assignment, we were required to implement the K-Means algorithm without the use of libraries except for pre-processing. The Input Data (provided as new\_test.txt) consists of 10,000 images of handwritten digits (0-9). The images were scanned and scaled into 28x28 pixels. For every digit, each pixel can be represented as an integer in the range [0, 255] where 0 corresponds to the pixel being completely white, and 255 corresponds to the pixel being completely black. This gives us a 28x28 matrix of integers for each digit. We can then flatten each matrix into a 1x784 vector. No labels are provided. Format of the input data: Each row is a record (image), which contains 784 comma-delimited integers.” (Project Specs).

Approach:

I decided to do this project in Java since I had no prior experience with Python. Using Python may have been easier because of the various built in libraries that assists with feature/ dimensionality reduction.

Part One, Iris Clustering:

First, I developed my K – means algorithm on a sample Iris data set. This data set is considerably smaller and contains only 4 features which allows for easier testing. The algorithm was tested on this data set to ensure correctness before moving on to the main portion of the project, image clustering.

1. Data Processing:

First, we need to open the test file and create the matrix. The matrix is represented as a list of lists. The number of total lines is 150 and the number of features is 4. No dimensionality reduction was needed due to the size of this data set. Once we have our matrix, we proceed to our K means algorithm.

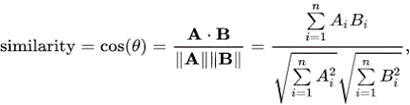
Part Two, Image Clustering

1. Data Processing:

First, we need to open the test file and create the matrix. The matrix is represented as a list of lists. The total number of lines 10,000 and the number of features is 784. We do preprocess on the data by removing all instances of commas using Java regex. Because this data set is quite large, we may want to implement some sort of feature selection or dimensionality reduction. I attempted to use truncated SVD to reduce the total number of features. However, the accuracy of the results was far less after performing this method and the runtime was also greatly increased.

2. K means

First, we initialize k means or k points randomly. To do this, we choose k random points from our test data and store them in a separate matrix. For the Iris data set, we select k to be 3 because we have 3 clusters that our instances can be classified into. For the Image data set we choose k to be 10 because we have 10 clusters that our instances can be classified into. Next, we iterate through the test data and classify each item to its closest mean. In other words, for each instance we find which of those k points it is closest to. To do this we use cosine similarity as our distance metric. Cosine similarity is a metric that measures how similar two vectors are. The formula used is given as



The cosine similarity represents the distance to the means, and we calculate this measure for every item. This is also our metric for internal evaluation. To see how similar the points are to our clusters we take the average of all these distances. Next, we find the mean closest to the item. This is simply the largest cosine similarity value and we keep track of that cluster labeled as either cluster 0, cluster 1, etc. Once we have assigned the item to the mean we must update the mean’s coordinates. This is the average of all the items categorized in that mean. We repeat this process for a given number of iterations. For both data sets I tried with 5, 10, 100 iterations. The biggest jump in accuracy was from 5 to 10 because the means changed the most. From 10 iterations to 100 iterations the accuracy did not change.

3. Runtime

O (N \* K \* I \* F)

N: represents the number of points,

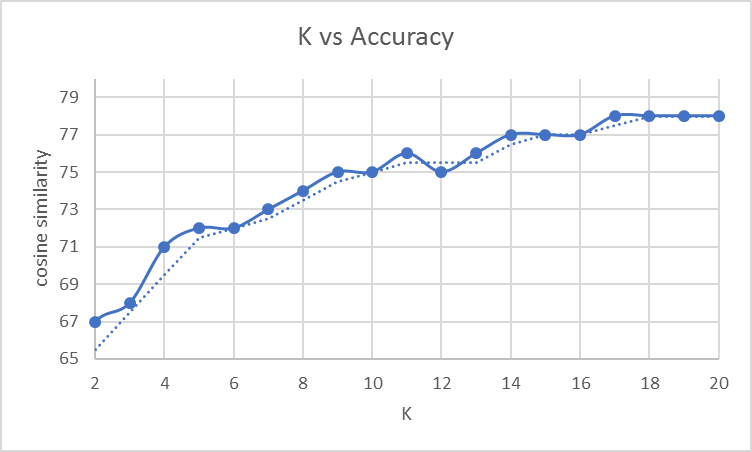
K: represents the number of clusters,

I: represents the number of iterations

F: represents the number of features

The runtime of this algorithm is slow. For large datasets the algorithm may take long time to run. For K = 10 it takes ~ 10 seconds to run.

|  |  |
| --- | --- |
| K | Cosine Similarity Metric |
| 1 | 64 |
| 2 | 67 |
| 3 | 68 |
| 4 | 71 |
| 5 | 72 |
| 6 | 72 |
| 7 | 73 |
| 8 | 74 |
| 9 | 75 |
| 10 | 75 |
| 11 | 76 |
| 12 | 75 |
| 13 | 76 |
| 14 | 77 |
| 15 | 77 |
| 16 | 77 |
| 17 | 78 |
| 18 | 78 |
| 19 | 78 |
| 20 | 78 |



The relationship between K and our internal accuracy measure is approximately logarithmic.

4. Results

The username registered on miner website is: (kgu2). At the time of this report, the current rank is current accuracy is 95 for Iris data and 52 for Image data. One thing that could have been done to improve results was creating more clusters and merging them based on the existing known cluster.