Case Study 1: K-means and Image Processing

Ethan Nussinov
Department of Electrical and Systems Engineering
Washington University in St. Louis
St. Louis, Missouri
ethan.nussinov@wustl.edu

Seif Elkhashab
Department of Electrical and Systems Engineering
Washington University in St. Louis
St. Louis, Missouri
e.seif@wustl.edu

Abstract - In this case study, we created a program capable of identifying handwritten digits (0-9) and matching them to their digitized counterparts. After entering an image in MATLAB, it is broken down and processed as a 784-dimensional vector. We employed the k-means algorithm to cluster vectors into their respective numerical groups. After assigning each centroid its numerical meaning, we take our input image and use nearest neighbor criteria to assign it to a centroid, then check whether the meaning of that centroid matches the test image.

I. Introduction

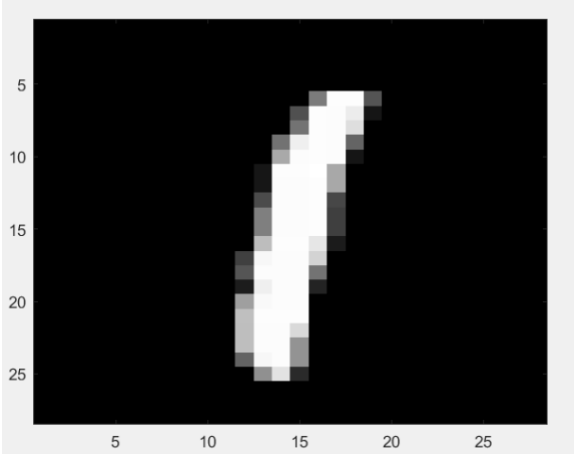
A project like this is important not only to understanding key concepts in ESE 105, but also in its application potential to the real world. Technology like this exists all around us: our iPhones with their photo to text technology, standardized testing machinery, and more. Our project aims to bring about a highly accurate method for identifying handwritten digits. The third learning objective of the ESE 105 class states that we shall be able to 'Understand the scope of ESE at Washington University and discover how your interests fit within your major.' This is an excellent example of how one can understand the scope of the class, through real world experimentation.

II. Methods

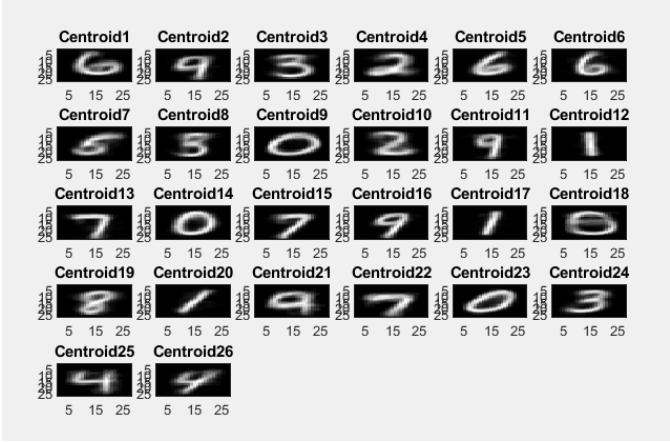
Our two goals in this project were to classify handwritten digits, and to identify outliers.

A. Classifying Handwritten Digits

The first goal is to classify handwritten digits. We did this by first building a k-means algorithm. There were three functions we needed to implement, being 'initialize_centroids', 'assign_vector_to_centroid', and 'update_centroids'. The first part, 'initialize_centroids' set up the initial values of all the centroids. We ran a training set of images and a testing set of images through MATLAB and stored the labels. After running this initial step, we had variables 'train', 'test', and 'correctlabels' in the workspace that we could manipulate. We then reshaped the 784-dimensional array into a 28x28 array. After reshaping the array, the value of k was set to 26, and a maximum number of iterations of the algorithm was set at 40.



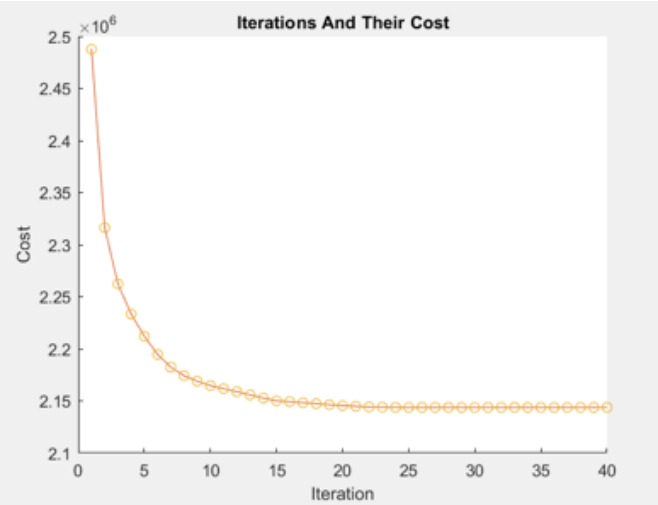
[Fig. 1 – example image illustrating the size of the vector (26x26)]



[Fig.2 – handwritten digits labeled as centroids]

After initializing our centroids and array, we set up the for loop that would go through each vector in 'train', assign it to its nearest centroid using 'assign_vector_to_centroid', then go through and update each centroid to a new, calculated centroid using 'update_Centroids', and finally, calculate the cost of each iteration using the k-means cost formula provided to us in class.

In the next section of the code, we plotted the k-means cost as a function of the number of iterations, then initialized the centroids.



[Fig.3 – Cost vs Iteration graph]

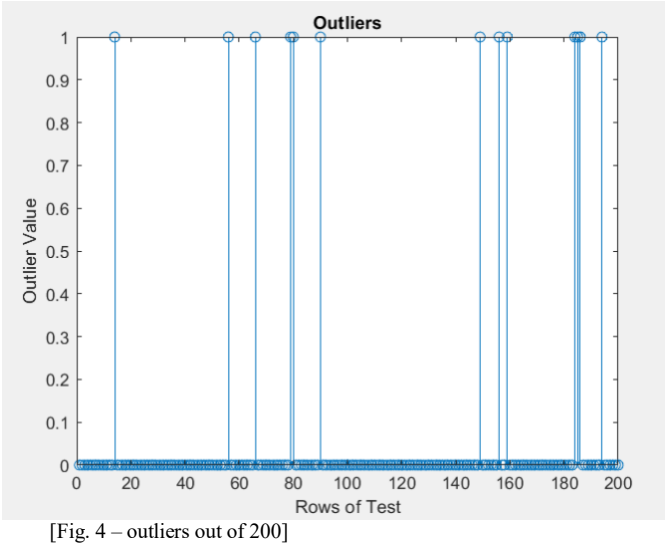
We created a function to pick the closest centroid using norm and distance. An easy way to represent this is the Euclidean distance formula, which is as follows:

$$d(a,b) = ||a - b|| . \quad (1)$$

Euclidean distance is simply the length of a line segment stretched between two points, in the equation above, those points are a and b. The Euclidean distance between two vectors is simply the exact distance between those vectors.

B. *Identifying Outliers*

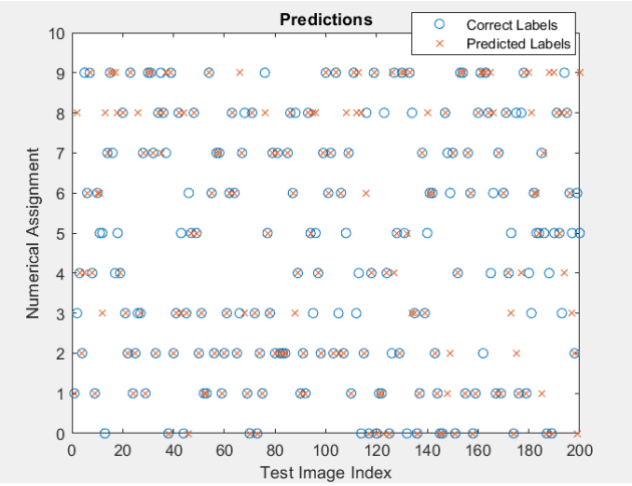
The outliers were calculated by finding the average distance between the centroids and the vectors of test. This was accomplished by using the calculate distance function that was created earlier which worked by calculating the Euclidian norm. After the average distance was calculated each vectors distance to its centroid was compared to that centroid's average distance. If the distance was 25% greater than the average distance. That vector was labeled as an outlier by assigning a 1 to its index in the outliers vector. In the outliers vector, a one equates to the vector being an outlier and a 0 equates to the vector not being an outlier.



[Fig. 4 – outliers out of 200]

III. Results and Discussion

Out of the 200 assigned labels, we correctly hit 147, taking outliers into account. With a 73.5% accuracy rate, we had a respectably accurate system in play. The image below illustrates the accuracy. The setbacks we experienced were the limitations of the k-means algorithm, discussed in greater detail in the conclusion section. Where the centroids themselves initialized could also have a positive or negative effect on our accuracy rate, as they are limited to two dimensions, so our visualization of them is not entirely accurate. This accuracy was calculated by finding the number of times that predictions equals the correctlabels. To find the percentage that number was divided by 200 which is the number of vectors in test. This method of analysis provided a quantitative method for calculating how well the k-means method was preforming and how well the design of the system worked.



[Fig. 5 – predictions/correct labels out of 200]

IV. Conclusion

After examining the results of the k-means algorithm and determining the accuracy between the predicted image labels and the correct labels, a conclusion can be drawn that K-means was successful in assigning the images to the correct number to a certain extent. K-means was a good method to process the handwritten numbers and assign them to their respective labels. Its advantages were that it was simple to implement, guaranteed convergence, and can be easily scaled to a large dataset if we had chosen to include a larger data set. The shortcoming of the algorithm is it does not have a plane to deal with outliers which can change a centroid from its optimum space. Even though a good plan was made to find the outliers, this was done after the k-means had already been run, so the centroids had already been finalized. Another downfall of the k-means algorithm is k must be chosen manually and it is not easy to determine the optimum number of centroids. Having an algorithm determine K would be more efficient and reliable.

V. References

[1] S. Boyd, L. Vandenberghe, Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares, 1st ed., vol. 1. Cambridge University Press, 2018

[2] C. Moler, "MATLAB " MATLAB Documentation www.mathworks.com/help/matlab