Case Study 1: Clustering and Classification

Kaan Dincer and Nick Falshaw

ABSTRACT— In this case study a k-means algorithm was built to cluster (classify) 28x28 images, which are each read as a 784-dimensional vector. The k-means algorithm consisted of 4 different steps: initialize the centroids, find distances of vectors to centroids, assign vectors to the centroids with the smallest distance between them, and update the centroids with the number of iterations. The number of iterations were given the value of 30, and k was given the value of 50. Through the k-means algorithm created in the study, the main goal was reached, which was to assign vectors into nearest centroids and predict the handwritten digit displayed in each image. The k-means algorithm was based on the idea of norms between vectors and centroids. The distance of each vector to each centroid was found, and each vector was assigned to the centroid closest to it. The centroids were updated with each iteration, and the final centroids were given labels regarding the vectors in the group. K-means cost, which was plotted as a function of number of iterations, showed the distances from all the centroids to their respective vectors. In addition, outliers were found and plotted on a stem graph. To determine the outliers, the first column of each 1x784 image was analyzed and the vector was flagged if the first entry was greater than zero. Finally, the accuracy of the predicted labels was determined.

I. Introduction

The goal of this case study was to classify unlabeled images given a set of vectors that are 28x28 images of handwritten digits through clustering. Classification is the idea of putting similar pieces of data into groups to determine the meaning of the data. One method of classification is clustering. Clustering aims to group vectors into *k* groups or clusters so that the vectors in each group are close to each other (Boyd). Clustering has many applications, but in this case study it was used to implement the k-means algorithm. K-means is a very popular clustering algorithm and it works by storing *k* centroids that it uses to define clusters. A vector is in a particular cluster if that centroid is the "nearest neighbor" meaning it is closest to the vector than any other centroid (Piech). The closeness of the vector and the centroids is determined using the Euclidean norm. The Euclidean norm is the square root of the sum of the squares of the entries of a vector, and the Euclidean distance between two vectors is found by subtracting the two vectors from each other and taking the norm of the result (Boyd). Once the vectors are assigned to centroids the centroids are then updated and the vectors are reassigned using the Euclidean norm in order to minimize the distances between the vectors and centroids. This process of updating and reassigning continues for a specified number of iterations or until the distances cannot be further minimized. Another important aspect of this case study was outliers. Given a test set of 200 images a procedure was coded to flag the 11 vectors that had been encoded differently than the rest. Outliers have a certain set of aspects that help find and distinguish them, and finding outliers can be very useful for analyzing or sorting data. ("Outlier.")

II. Methods

A. Classifying Using K-Means Algorithm

In order to classify the images a k-means algorithm was created. The first steps in building the k-means algorithm were to determine the number of centroids and the number of iterations, and to initialize the centroids. Originally, the code for initializing the centroids picked random vectors from the train set of 1500 images to be the centroids. However, the code was edited to have centroids picked manually instead of randomly, because this made sure that all digits (0-9) had a centroid and that the images picked as centroids were neatly written. Once the centroids were set a "for loop" was created to run the k-means algorithm. The first part of the loop went through each row vector in the train set and assigned it to the closest centroid. The nearest centroid was determined by using the Euclidean distance (norm) of each vector to all the centroids and then choosing the minimum distance to assign the vector to its nearest, most similar centroid. In the next part of the loop to enact the k-means algorithm the centroids were updated and the vectors were reassigned to their new closest centroid. The centroids were updated by taking the mean of all the vectors associated with each centroid. Then the vectors were reassigned using norm to find the closest centroid. The distances (norms) between the centroids and their respective vectors became smaller. This process was repeated for the number of iterations chosen. The distances between the vectors and centroids continued to decrease until a certain point, where it remained the same, as it couldn't get any smaller.

B. Determining Outliers

A strategy was designed and implemented to find the outliers. A for loop was created to run through each row of the test matrix. When the test matrix was examined the outliers were easy to identify, as they were numbers bigger than 0. So, inside the for loop an if statement was used, in which the ith entry of outliers (outliers(i,1)) was set to 1 if the entry in the ith row of test was greater than 0, otherwise the ith entry of outliers (outliers(i,1)) was set to 0.

C. Determining Accuracy

In order to determine the accuracy of the methods, the number of predicted labels that are equal to the actual labels was calculated. For this, the line of code "sum(correctlabels==predictions)" was used. This line of code added up the number of times the actual labels (correctlabels) was equal to the predicted labels (predictions).

Additionally, the correct labels ("correctlabels") and the predicted labels ("predictions") were graphed on the same figure. The number of intersections in the figure (*Figure 4*) of "correctlabels" and "predictions" were the number of times the correct labels were equal to the predicted labels. Therefore, the number of intersections in the graph also showed the accuracy of the methods.

III. Results

A. Quantitative Results

As seen in *Figure 3*, there were 11 outliers found from the method. The outliers that were found from the test set had the following indices: 11, 62, 76, 81, 111, 113, 124, 135, 167, 187, 194. As seen in *Figure 4* and from the given line of code "sum(correctlabels==predictions)" it was calculated that 126 out of 200 test images were predicted accurately. This was around 63% accuracy. Therefore, although the percentage wasn't extremely high, the majority of the test images were labeled correctly.

B. Figures

Figure 1: Centroids

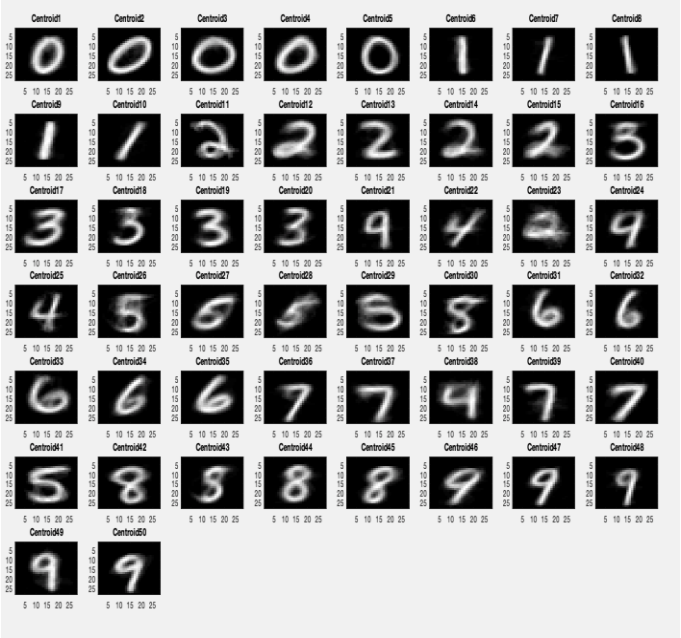


Figure 2: K- means Cost As A Function Of The Number Of Iterations

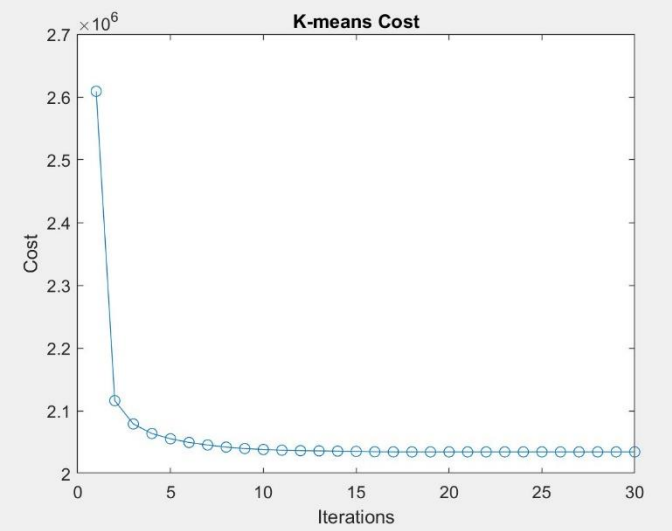


Figure 3: Outliers

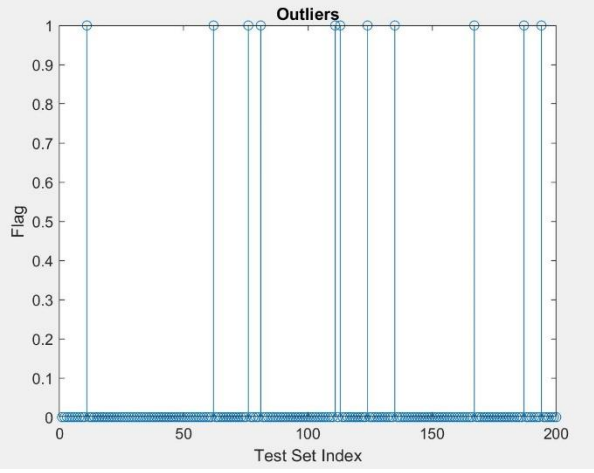
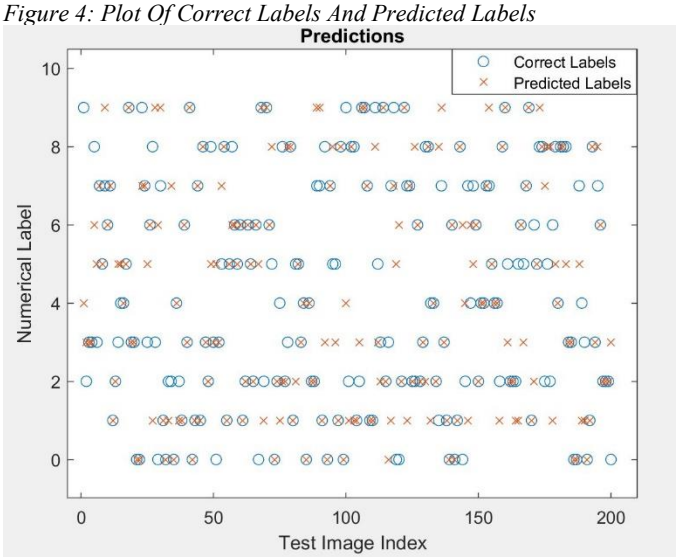


Figure 4: Plot Of Correct Labels And Predicted Labels



IV. Conclusions

After examining the results of the study on clustering images and determining accuracy it was determined that the k-means algorithm created in the study predicted labels accurately to a decent extent. However, the accuracy of predicted labels wasn't very high, so the k-means algorithm isn't perfect. In addition, the method to determine accuracy worked efficiently, and *Figure 4*, the graph that plotted "correctlabels" and "predictions" showed how accurate the predicted labels were to a very high extent.

Further, the method used in this study determined 11 outliers, as it can be seen in *Figure 3*, and 11 outliers was the expected result. Therefore, after examining the results of the study on finding the outliers it was determined that the method used to find the outliers worked very efficiently.

References

1. Boyd, Stephen P., and Lieven Vandenberghe. *Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares*. Cambridge University Press, 2018.

2. Piech, Chris. "K Means." *CS221*, 2013, stanford.edu/~cpiech/cs221/handouts/kmeans.html.

3. "Outlier." Outlier - an Overview | ScienceDirect Topics, www.sciencedirect.com/topics/mathematics/outlier.

4. "MATLAB." *MATLAB Documentation*, www.mathworks.com/help/matlab/.