Tanasorn Chindasook
Prateek Kumar Choudhary

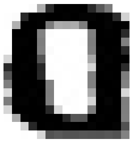
Mini-Project 1

Machine Learning
March 12, 2019

Task

The following project contains the code documentation and the discussion of the mathematical structures, visual observations, iteration comparisons and multiple digits clustering of codebook images.

As a first step in solving this task we create the visualisation function to visualise the digit data which is provided in the form of two dimensional array, with 2000 rows and 240 columns. Example image generated for one of the numbers:



Code Documentation

The attached code consists of the following three functions and its' implementation:

plot_figure(data, imgNme),

random_assign_class(df, k), and

kmeans(data, clusterSize).

The **plot_figure** function creates a 16 x 15 pixel plot of the input data and saves the output as a file; its name and format specified using the *imgNme* parameter. This function was specifically created for the visualisation of the digits data set, therefore the input data must be a row vector with 240 dimensions in the format $x \in [0, 6]^{240}$.

The **random_assign_class** function is a helper function to ensure proper initialisation of the clusters in the function kmeans. It takes two parameters: *df* (the data set one wishes to work with) and *k* (the number of clusters).

For example, let us assume the case $K = 5$ and data points $N = 200$. In the case of K = 5, five clusters will be initialised with labels {0,1,2,3,4}. The function works by first creating a list of length K (in this case 5), with all of the possible distinct values within the range {0 to K-1}. The list is then shuffled and assigned to a variable for storage. The length of the list (5) is then subtracted from the remaining N datapoints (200), which will return an integer m (195) that is used to pass into the *np.random.randint* function to generate m additional random cluster initialisations ranged {0 to K-1}. The two lists are then joined together and appended to the data set, creating the random initial clusters. This method guarantees that all K distinct clusters are initialised at the start of the algorithm, especially for large K values.

The **kmeans** function takes two parameters *data* and *clusterSize*, and utilises the **random_assign_class** function to initialise each vector to a random cluster. The algorithm will then compute centroids for each cluster by taking the mean of all the vectors in the cluster. The algorithm then calculates the Euclidean distance of each vector from the centroids of each cluster and will reassign each vector to the class of the closest centroid using the *argmin* function. The algorithm terminates when no class reassignments are made. The output of this algorithm is a list of K centroid vectors.

After implementing K-Means clustering with Python for the sub data set of '2's we get the following codebook vectors:

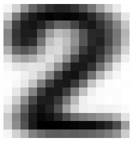
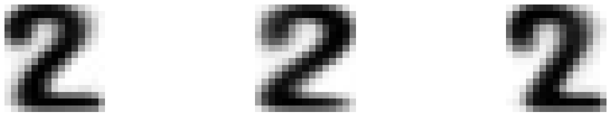


Figure 1: K=1 Codebook vector 1

(a) Vector 1

(b) Vector 2

Figure 2: K=2 Codebook vectors



(a) Vector 1

(b) Vector 2

(c) Vector 3

Figure 3: K=3 Codebook vectors

(a) Vector 15 (b) Vector 101 (c) Vector 157 (d) Vector 193

Figure 4: K=200 Codebook vectors

Discussion

0.1 Mathematical Nature of Codebook Image with $\boldsymbol{K}$=1

For the case of K=1, the grey-scale value of each pixel on the codebook vector visualisation is an average of the values from all 200 images.

$$C = \frac{\Sigma_{n=1}^{200} A_n}{200}$$

For all, $A \in [0, 6]^{240}$.

The level of "fuzziness" in the image is an indicator of how much variation there is between the 200 observations of twos.

0.2 Mathematical Nature of Codebook Image with $\boldsymbol{K}$=200

For the case of K=200, each codebook vector visualisation is simply one of the images randomly selected from the set of all images.

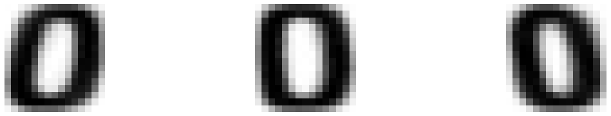
0.3 Visual Observations of Codebook Images

The level of "fuzziness" in a codebook image is an indicator of how much variation there is within the cluster. When K=1, the codebook image is extremely fuzzy compared to the much sharper looking codebook images of K=200. As K increases, the level of "fuzziness" of the codebook image decreases. This can be attributed to the fact that the codebook vector for a smaller value of K is a representation of a larger amount of data points, and will have to account for the variations between all of the data points in the cluster.

Another interesting observation that can be made is how Vector 1 in K=2 is slightly "left-leaning" whilst Vector 2 is slightly "right-leaning".

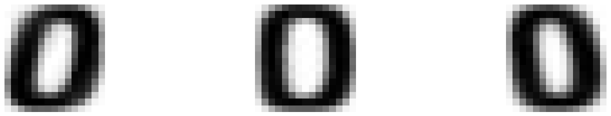
0.4 Investigation of Random Initialisation Effects

Furthermore, in order to explore the effects of different cluster initialisations, the k-means algorithm was run on the zeros data set five times for $K = 3$. The results show that the three codebook vectors possess somewhat similar characteristics across all iterations, with each iteration consisting of a codebook vector visualising a “straight” zero, a “left-leaning” zero and a “right-leaning” zero. The first prominent difference is the order in which the three types of zeroes appear, denoting how k-means does not produce identical results each time. The slightly less noticeable variations are between the shapes of each type of zero, which is an indication that observations can be assigned to different clusters according to their initialisation values.



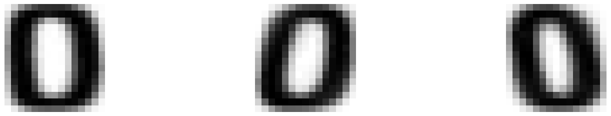
(a) Vector 1 (b) Vector 2 (c) Vector 3

Figure 5: K=3, Run 1 Codebook vectors for the zeroes data set



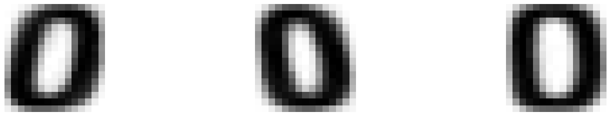
(a) Vector 1 (b) Vector 2 (c) Vector 3

Figure 6: K=3, Run 2 Codebook vectors for the zeroes data set



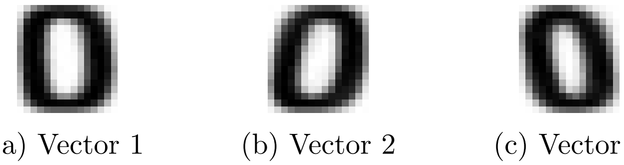
(a) Vector 1 (b) Vector 2 (c) Vector 3

Figure 7: K=3, Run 3 Codebook vectors for the zeroes data set



(a) Vector 1 (b) Vector 2 (c) Vector 3

Figure 8: K=3, Run 4 Codebook vectors for the zeroes data set

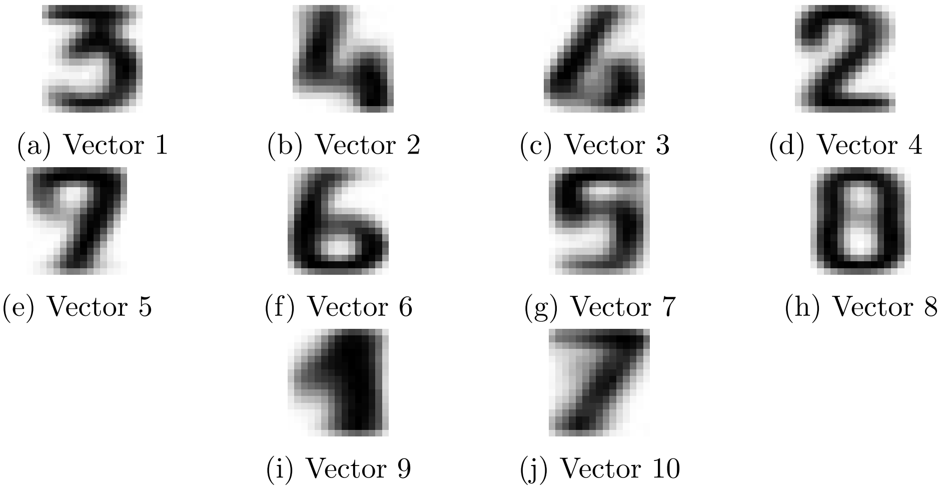


(a) Vector 1 (b) Vector 2 (c) Vector 3

Figure 9: K=3, Run 5 Codebook vectors for the zeroes data set

0.5 Clustering of Entire Digits Dataset

In order to investigate the effectiveness of the algorithm, K-means clustering was also performed on the entire digits dataset with $K = 10$ to observe whether or not it would be able to correctly distinguish between the 10 digits. The graphic below depicts how the algorithm is able to cluster the data points into its respective digit groups rather well; however, the algorithm has difficulties differentiating between certain numbers such as: 0 and 8, 4 and 6, 7 and 9. For digits that the algorithm is able to properly differentiate, the codebook vectors are visually similar to the K=1 codebook vector for their respective digits.



(a) Vector 1 (b) Vector 2 (c) Vector 3 (d) Vector 4
(e) Vector 5 (f) Vector 6 (g) Vector 7 (h) Vector 8
(i) Vector 9 (j) Vector 10

Figure 10: K=10 Codebook vectors for the whole Digits data set