CSP 304: MACHINE LEARNING LAB (SPRING 2023)

HOMEWORK 3

REPORT

NAME: SAMARTH BHATT

ROLL NO: 2020KUCP1068

LAB BATCH: A3

TOPIC: Clustering and Expectation Maximization

Experimental Tasks:

This homework illustrates the use of expectation maximization (EM) methodology for data reduction or clustering. Image compression is performed for K = [4, 8, 12] number of clusters or Gaussian distribution. Compressed images for each value of K are displayed and plotted the graph between log-likelihood after the E-step and M-step and iteration number in one curve for all value of K, to visualize the convergence of EM algorithm.

Introduction:

K-means is the unsupervised clustering technique used to cluster the data into various cluster identities. In-case of K-means, it follows hard assignment of cluster identities to data points. Gaussian Mixture Models are type of probabilistic clustering model that takes into account the probability of assignment of cluster identities to each data point. In a way it is referred as soft assignment of cluster identities.

Expectation Maximization is a technique that is used to find the missing data that could or must have been derived from the Gaussian distribution given the parameters that satisfy the distribution of the dataset. This same can be applied in reverse as estimating the missing parameter of Gaussian distribution if the dataset is derived from some normal distribution.

This feature of EM algorithm is used to compress the image by clustering the pixels using GMM i.e., Mixture of K Gaussian distributions. By using the EM algorithm we estimate the probability of each pixel value to belong to particular distribution in the Mixture of K Gaussian distributions. The detail method is explained in further section.

Process and Setup:

Data Preprocessing

The Image compression is done on the 200 x 67 bitmap image of stadium (see figure 1). To read the image, imread() function of io module of skimage package is used. The image contains 3 channels corresponding to RED, GREEN and BLUE. Therefore, the 3D image produced has the shape of [200, 67, 3]. This 3D image data is converted to 2D data by multiplying the dimensions value, the reshape() function of numpy library is used. The resultant matrix has a shape of [13400, 3]. Where the rows represent the pixel value and column represent the three channels i.e., RED, GREEN and BLUE.
The number of clusters or say Gaussian distribution is given by an array K = [4, 8, 12]. The image is compressed for each value of K. The function EMG() that implements the EM algorithm takes two parameters. First is the image path and second is the K value.

Figure 1: Stadium.bmp image

Initial Parameters Estimation

The first step in the EMG() function is to estimate the initial parameters for the K Gaussian distribu-tions. K means algorithm is used to estimate the initial parameters like mean vector, prior probability and covariance matrices. First, the mean vector is estimated from the k means using the cluster cen- ter attribute of K means model object. The shape of mean vector is [K, 3] where rows represent cluster number and column represent the values in RGB channels.

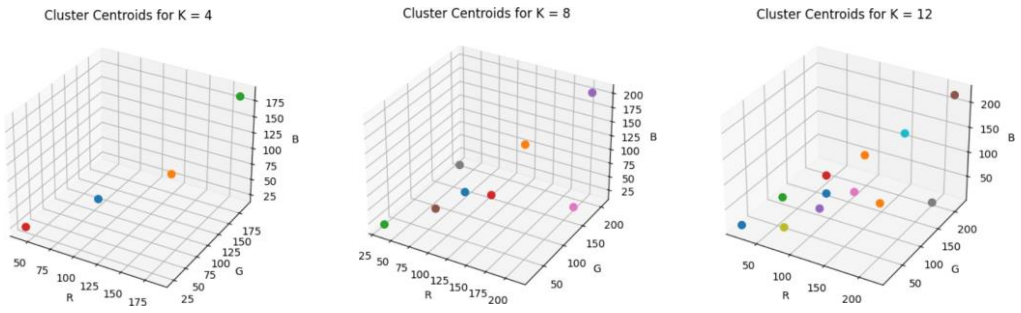

Figure 2: The initial centroids of K clusters estimated using K-means Algorithm

Second, the covariance matrix for each K cluster. The covariance matrix for each cluster is also estimated from the k means. Since the shape of covariance matrix is [3, 3] and this is for all K clusters. Therefore, the resultant matrix has a shape of [K, 3, 3].

Last, priors of each cluster. It is a prior probability that represents the proportion of data points that belong to that cluster, they determine the contribution of each cluster to the overall probabilitydensity function of the data.The prior probability is estimated by counting the number of pixels in each K clusters.

Expectation Step

In the E-step, the algorithm computes the posterior probability, which is the probability of how much each pixel belongs to each cluster. The algorithm do so by estimating the probability density function of image for each cluster and then multiplying it with the priors of each cluster to get the final posterior probability of each pixel in the image.

Maximization Step

Now, that we have computed the posterior probabilities of each pixel for each cluster. The maximization of previously estimated parameters is done, that means estimation of new values of the previously set parameters. First, priors are estimated. To estimate the new values of priors for each cluster, the posterior probabilities of each pixel in the image are summed up for each cluster. Then, this summed up values of each cluster is divided by the total number of number of pixels in the image that gives the new set of priors for each cluster.

Second, new centroids of each cluster are estimated by multiplying the transpose of posterior probabilities matrix with the image data matrix. The resultant matrix is the weighted sum of RGB values for each cluster. Now, this matrix is divided by the sum of posterior probability of each cluster to obtain the new values of centroids. Similarly using the formula for estimating covariance matrices of each cluster the same is done.

Results and Discussion:

The Expectation and Maximization step is iterated for around 200 times so that better convergence can be estimated for our parameters before applying compression to image. In each iteration the value of log-likelihood is stored. The plot between complete log-likelihood and iteration number is plotted to visualize the convergence of parameter by observing the pattern in log-likelihood values over the specified iterations. Later, for each value K, the image is compressed based on the EM algorithm.

Complete Log-Likelihood vs Iteration Graph

From the Plot of complete log-likelihood vs iteration number it can be seen that with initial parameters set by kmeans algorithm converged very quickly in case of K = 4 clusters (in blue) then K = 12 took less time to converge and K = 8 took the most time or iteration to converge. Also, the maximum iteration set for EM algorithm can be lower down from 200 to 150 to improve timing and computation. The time taken by EM algorithm to compute for all K values comes out to be around 13.28 seconds in VS code environment.

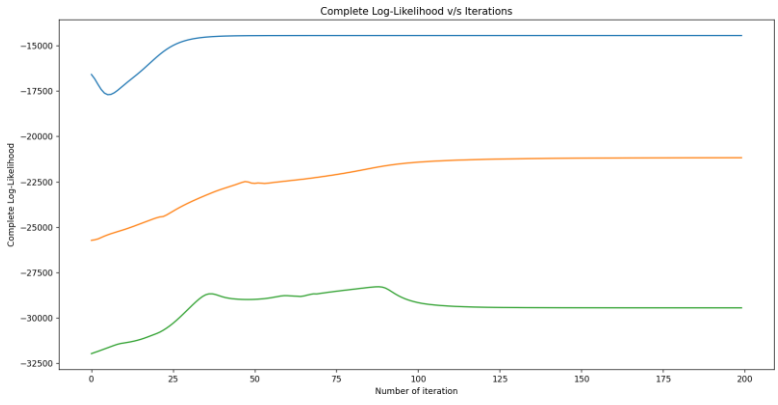

Figure 3: Complete log-likelihood vs Iteration plot for k = 4, 8 and 12

After convergence it can be seen that some of the initially estimated centroids of clusters are shiftedbecause of EM algorithm.

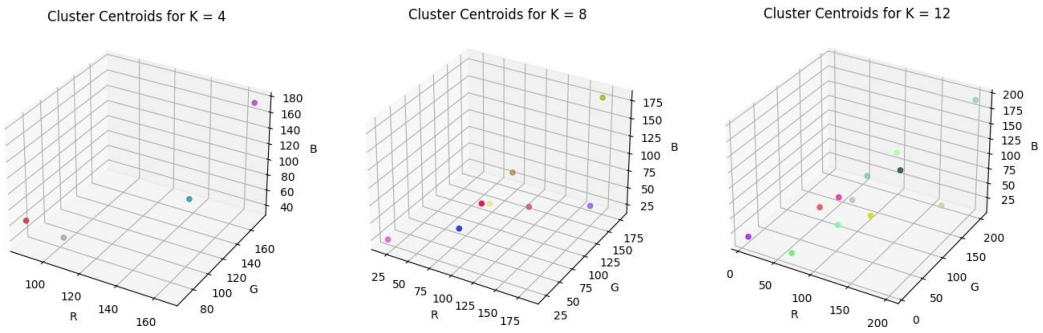

Figure 4: The converged or final centroids of K clusters estimated using EM Algorithm

Image Compression for K Clusters

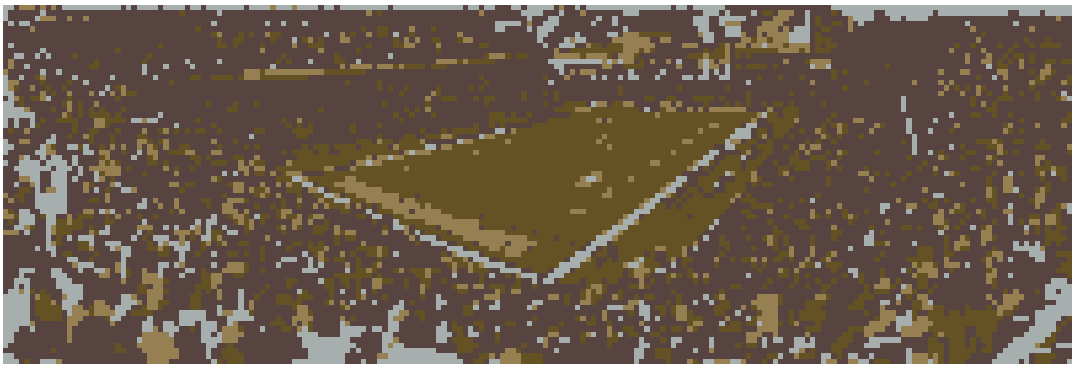

Figure 5: Compressed image for K = 4

Figure 6: Compressed image for K = 8

Figure 7: Compressed image for K = 12

The compression is performed by first declaring and initializing a zero matrix image of same shape as original image. Then, the pixel membership is computed for each cluster by performing argmax function on the posterior probability matrix. Through which we get the cluster numbers for each pixel to which they belong. After that a mask matrix of same dimension as of image is made which contains the true values for pixel corresponding to clusters. Using this mean values of each cluster is replaced sequentially and compressed image is formed.

Conclusion/Comments:

In this project, we implemented the Expectation-Maximization algorithm with Gaussian Mixture Models to compress an image. The EM algorithm is an iterative algorithm that alternates between the E-step, where the posterior probability of each data point belonging to a cluster is computed, and the M-step, where the model parameters are updated based on these probabilities. We applied the EM algorithm with K-means initialization for K = [4, 8, 12] clusters.

One interesting observation from the log-likelihood plot is that the algorithm converged relatively quickly for all values of K, as the log-likelihood function plateaued after a few iterations. This suggests that the algorithm found a local optimum for the model parameters. We also observed that as K increased, the log-likelihood increased, which is expected since more clusters provide a better fit to the data.

In terms of the compressed images, we can see that increasing the number of clusters leads to a more detailed image. However, we also observed that there is a trade-off between compression and quality, as a higher number of clusters led to larger file sizes. Overall, we can conclude that the EM algorithm with Gaussian Mixture Models is an effective technique for image compression.

In conclusion, the EM algorithm with Gaussian Mixture Models is an effective technique for image compression and can be used in various applications where data compression is needed.