To implement the K-Means algorithm and Expectation Maximization Algorithm for clustering using a Gaussian Mixture Model.

PROJECT DESCRIPTION

- Programming language used : Python
- Data Structure : Lists, Matrix
- File Name : Cluster_K_GMM.py
- Inputs: clusters.txt, set of 2dimensional - 150 data points
- Output:

  KMeans – centroid of each cluster, and its length

  GMM – Mean, Amplitude, Co-Variance of the probability distribution

IMPLEMENTATION

- Modules created :

1. ReadFile(): to save the input file into List of points for the given dimensions
2. CreateClusters(): Calculating clusters based on distance from the centroid
3. CalculateMean(cluster): Finds mean / centroid for each cluster
4. kmeans(): recursive function, that is used as the caller for :
   - creating clusters, based on the mean value, randomly generated
   - calculating mean for the new clusters
   - checking if the old mean is the same as new mean of cluster
   - if the means are same, therefore, no further cluster modifications are required and we have reached the optimum solution
   - if means are not same, kmeans performs the computation again, with the new mean values.
5. Gj(A,B,C) : For computing the value of the expression and returning G as:

   $$g_j(x) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_j|}} e^{-\frac{1}{2}(x-\mu_j)^T \Sigma_j^{-1}(x-\mu_j)}$$

   Where A = $(x-\mu_j)^T$, B= $(x-\mu_j)$, C = Σ_j (covariance matrix for cluster j), x = datapoint
6. amplitude(j) : For computing the value of the expression and returning W as:

   $$w_j^{(i)} = \frac{g_j(x)\phi_j}{\sum_{l=1}^{k} g_l(x)\phi_l}$$

   j=cluster #, ϕj = probability of cluster j
7. sumAmplitude(j): For calculating the sum of all the W, for a given cluster.
8. GMM_cluster(cluster,Mean, CoV) : For computing values and calling Gj(A,B,C) and passing it further, assuming KMeans clusters and Mean as the input.
9. CalculateD(cluster, Mean) : For computing the D matrix : X(i) -μj , used to find Covariance, for a given cluster and Mean
10. CalculateTranspose(matrix): For computing the transpose of any matrix passed to it.

11. MeanMaximization(j): For computing the Mean in the maximization step as per the below expression, where j is the cluster number, X(i) is the datapoint and Wj value has been computed for the cluster:

$$\mu_j \quad := \quad \frac{\sum_{i=1}^{m} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}}$$

12. CovMaximization(j): For computing the Covariance in the maximization step as per the below expression, where j is the cluster number, X(i) is the datapoint and μj is the maximization mean for the cluster:

$$\Sigma_j \quad := \quad \frac{\sum_{i=1}^{m} w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{m} w_j^{(i)}}$$

13. main(cluster1, cluster2, cluster3,Mean1,Mean2, Mean3, CoV1, CoV2, CoV3): Recursive function call for the complete Expectation Maximization.
14. checkListEqual(Mu, Mean1): Checks if the old mean for the given cluster is equal to the new mean or in a range of 3% error, returns 1 for the case when

- Mean1(old Mean with -3% error) <=Mu(new mean after maximization)>=Mean1(old Mean with +3% error) or (Mean1 = Mu)

- <u>Termination Condition</u> :
  - KMeans: No change in the centroid from the previous iteration and the new iteration.
  - EM : No change in New Mean and old Mean values

- <u>Complexity:</u>

  The code complexity does not exceed O(n) in the worst case scenario as well, where n= #of datapoints in the input set.

- <u>Result Interpretation:</u>

  The Kmeans output presents fixed clusters depicting hard membership of a data point to a cluster.

  Gaussian mixture model determines these clusters without associating each sample with a cluster. It brings out a probabilistic approach of soft membership of each datapoint to the dataset.