Programming Assignment 2: K-Means and Expectation Maximization for clustering using a Gaussian Mixture Model

Srihari Akash Chinam
USC ID - 2953497706
Net ID - chinam

Sayan Nanda
USC ID - 2681592859
Net ID - snanda

Implementation

- Language used - Python version 2
- Libraries used - Numpy (Arrays, Matrices), matplotlib (PyPlot), SciKit-Learn [4]

K-means algorithm - Programming Assignment Implementation vs SciKit-Learn Implementation

The input file, 'clusters.txt' is stored as individual lists stored within a list, this is a nested list structure. The point belonging to each cluster is again stored in a list. The points are initially randomly assigned to one of the clusters. The re-computation of the centroids is done till there is no change in any of the means.

Table 1 Cluster Centers for Programming Assignment Implementation

	X co-ordinate	**Y co-ordinate**
Cluster 1	2.88349711	1.35826195
Cluster 2	5.43312387	4.86267503
Cluster 3	-1.03940862	-0.6791968

Table 2 Cluster Centers for SciKit-Learn's implementation

	X co-ordinate	**Y co-ordinate**
Cluster 1	5.73849535	5.16483808
Cluster 2	3.28884856	1.93268837
Cluster 3	-0.96065291	-0.65221841

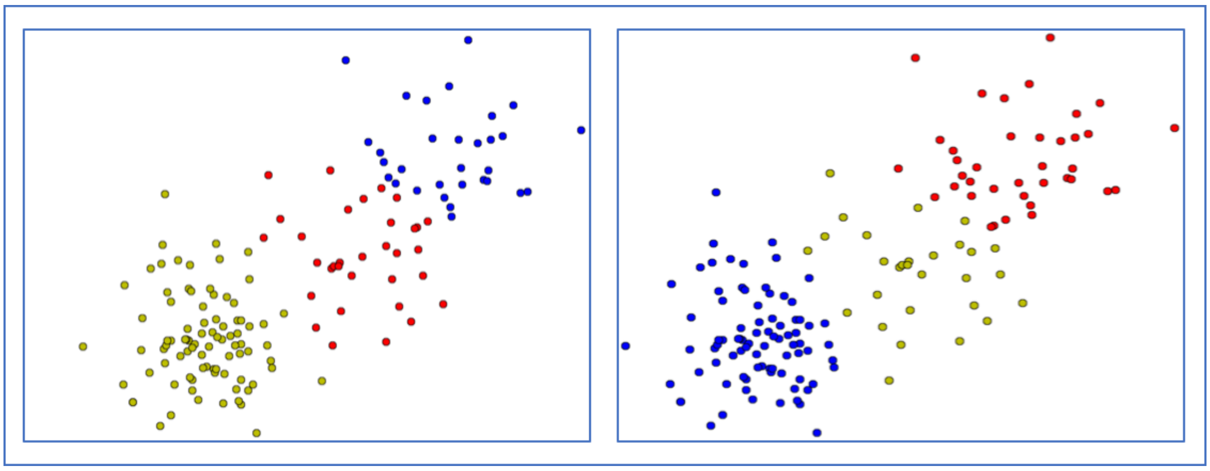
Figure 1 Implemented algorithm's output shown on left and SciKit-Learn algorithm's output shown on the right

Expectation Maximization for clustering using Gaussian Mixture Models - Programming Assignment Implementation vs SciKit-Learn Implementation

There are two ways to begin the Expectation-Maximization (EM) algorithm - beginning with random values for weights or taking values for weights from the output of k-means algorithm. By multiple attempts of trial and error, it was observed that better results were obtained by using the output taken from k-means algorithm. The EM algorithm terminated if the change in log-likelihood was less than 1.0e-7. The means, amplitudes and covariance matrices for the 3 clusters created are shown in Table 3, Table 4 and Table 5 respectively.

Table 3 Means obtained for the 3 clusters

	X co-ordinate	Y co-ordinate
Cluster 1	8.25803596	7.53712352
Cluster 2	-1.83936876	-1.21860855
Cluster 3	8.78998433	4.71220804

Table 4 Amplitudes obtained for the 3 clusters

	Amplitude
Cluster 1	0.292582239735
Cluster 2	0.556655661217
Cluster 3	0.150762099048

Table 5 Covariance Matrices obtained for the 3 clusters

	Covariance Matrix	
Cluster 1	21.53352415	18.4040288
	18.4040288	19.61218556
Cluster 2	2.26373243	0.56963591
	0.56963591	2.45627949
Cluster 3	24.09121068	13.37141198
	13.37141198	9.34679845

After implementation from scratch for the EM algorithm, the SciKit-Learn version of Gaussian Mixture Models was tested with the same data. The means, amplitudes and covariance matrices obtained for SciKit-Learns implementation are shown in Table 6, Table 7 and Table 8 respectively.

Table 6 Means obtained for the 3 clusters in SciKit-Learn's implementation

	X co-ordinate	Y co-ordinate
Cluster 1	-0.96174678	-0.63618636
Cluster 2	5.62092993	4.99410684
Cluster 3	3.2842769	1.9291939

Table 7 Amplitudes obtained for the 3 clusters in SciKit-Learn's implementation

	Amplitude
Cluster 1	0.57095326
Cluster 2	0.21343871
Cluster 3	0.21560803

Table 8 Covariance Matrices obtained for the 3 clusters in SciKit-Learn's implementation

	Covariance Matrix	
Cluster 1	1.23856256	-0.09316563
	-0.09316563	2.02291629
Cluster 2	2.2216601	0.15446244
	0.15446244	2.094973
Cluster 3	1.91930032	0.14344635
	0.14344635	2.57669337

After obtaining the results from each implementation, their outputs were visualized, shown in Figure 2. The output on the left is obtained from the programming assignment's implementation while the one on the right is obtained from SciKit-Learn's implementation. In Figure 3, Log-Likelihoods from 4 cases are visualized as graphs, showing how the EM algorithm is terminated as the graph flattens out.

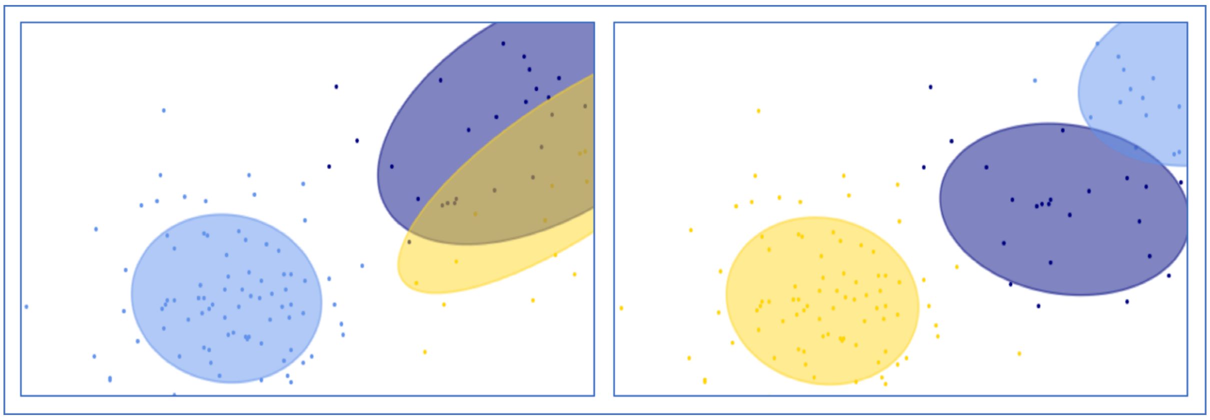

Figure 2 Implemented algorithm's output shown on left while SciKit-Learn algorithm's output shown on right

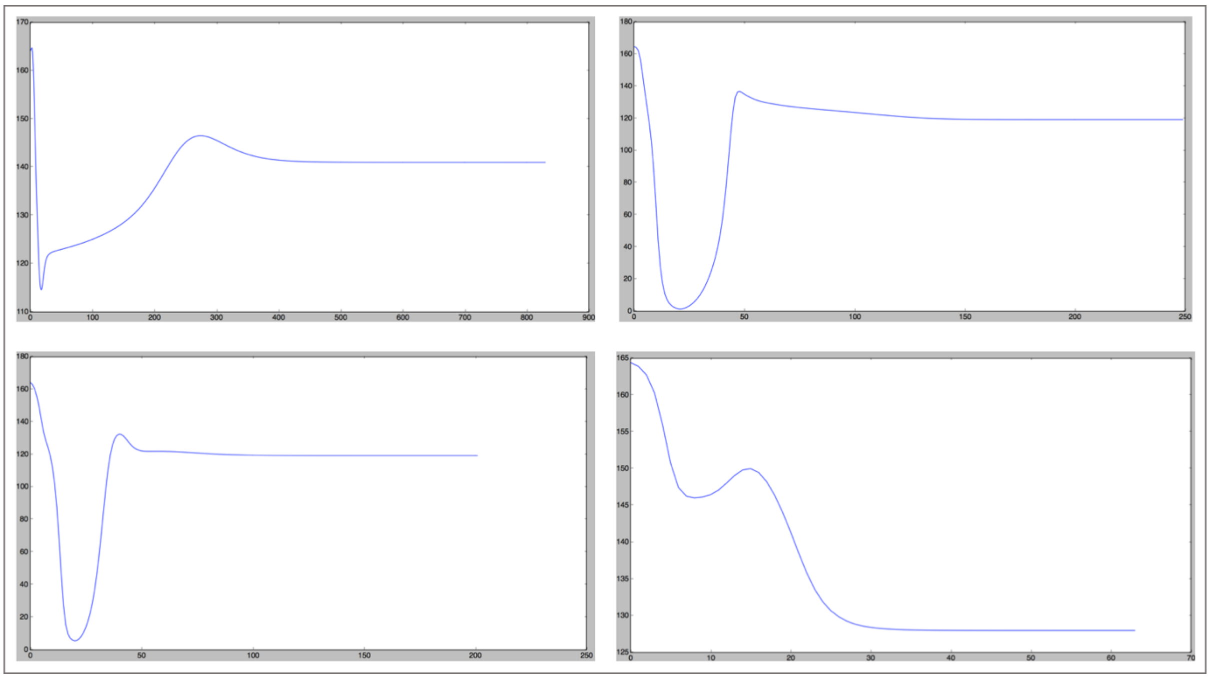

Figure 3 Log-Likelihood for every iteration as shown in 4 cases, showing the convergence of clusters

The difference between GMM and k-means is that GMM allows soft membership for a point. This in effect means that a single point can belong to more than one cluster. This can be seen in the visualization of the two outputs. We can view k-means as a special case or generalization of GMM wherein one point must only belong to a single cluster rather or one in which no soft membership is allowed.

Software Familiarization

The python library SciKit-Learn offers an implementation of k-means and the Gaussian mixture model.

For their k-means implementation, they allow parameters for initializing the centroids and clusters. The method we have used is like their ‘random option’. There is another option ‘k-means++’ which initializes the clusters in a smart way to speed up the convergence. There is also an option to run multiple times and choose the best output. This will prevent getting stuck in the local minima. There is also an option to precompute the distance. This requires more space but will run faster. There are also options to allow some tolerance for the convergence.

Some improvements that can be made to our algorithm are, running multiple times with random seeds and then choosing the best solution. This will increase the chances of finding the global minima. We can also allow tolerance for convergence but as the number of points is not that large (150 as opposed to a much greater number), the function reaches convergence easily, so there will be no real benefit of implementing this step.

The SciKit-Learn implementation of Gaussian mixtures allows the users to choose amongst diagonal, spherical, tied and full. Our implements a full covariance matrix. It uses a Bayesian information criterion to assess the number of clusters in the data. For the initialization step, there are two options, one is to randomly assign weights and the other uses the k-means clusters. It also allows parameters for initialization of means, weights and inverse of covariance matrix. Convergence threshold is also set. This allows the program to terminate once the change is minimal.

Some of the improvements from the SciKit-Learn have been implemented in our version. We initialize using the k-means output. This improved the result. We also use a log estimation for the convergence threshold. Once the log function displays very small change, it is assumed that convergence has been reached. Possible improvements that we can make are, running the function multiple times to find the best result and to use a ‘warm start’, where in the results from the previous iteration are used to initialize the next iteration. This can speed up convergence.

Applications

K-means algorithm is easy to implement and is scalable when using effective heuristics such as Lloyds algorithm. The algorithm originated from signal processing and is still used for in computer graphics, color quantization. K-means has been combined with simple linear classifiers for semi-supervised learning in NLP and Computer vision applications. K-means has found its way into financial applications as well such as market price and cost modelling. Other applications include pricing segmentation, loyalty segmentation [7], spend behavior and other types of segmentation [8], server clustering [1].

Gaussian Mixture Models (GMM) find their applications across a variety of disciplines. For example, in studies which deal with ethnicity, the soft membership of GMM is pivotal to a useful outcome. An example of this is its use studies related to human skin color [2]. Due to this soft membership, GMM are used in statistical testing. Multivariate Gaussian mixtures or continuous mixtures are also modelled using GMM. It is currently being used in video background subtraction [3], speaker identification and verification [5], shadow detection [6], amongst other applications.

Individual Contributions

Sayan Nanda

- Implementation of models
    - Preprocessing data - reading and formatting from text file
    - K-means algorithm

- Documentation
    - Implementation of K-means algorithm
    - Software Familiarization
    - Applications of K-means algorithm

Srihari Akash Chinam

- Implementation of models
    - EM algorithm for clustering using a Gaussian Mixture Model
    - Implementation of K-means and GMM from SciKit-Learn’s library
    - Visualization of K-means, GMM and Log-Likelihood’s outputs

- Documentation
    - Implementation of GMM
    - Comparison of performances of assignment implementation and SciKit-Learn’s implementation
    - Applications of GMM

References

[1] http://dni-institute.in/blogs/k-means-clustering-examples-and-practical-applications/

[2] Yang, Ming-Hsuan, and Narendra Ahuja. "Gaussian mixture model for human skin color and its applications in image and video databases." Storage and Retrieval for Image and Video Databases (SPIE). 1999.

[3] Lee, Dar-Shyang. "Effective Gaussian mixture learning for video background subtraction." IEEE transactions on pattern analysis and machine intelligence 27.5 (2005): 827-832.

[4] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[5] Reynolds, Douglas A., Thomas F. Quatieri, and Robert B. Dunn. "Speaker verification using adapted Gaussian mixture models." Digital signal processing 10.1-3 (2000): 19-41.

[6] Zivkovic, Zoran. "Improved adaptive Gaussian mixture model for background subtraction." Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on. Vol. 2. IEEE, 2004.

[7] Füller, Johann, and Kurt Matzler. "Customer delight and market segmentation: An application of the three-factor theory of customer satisfaction on life style groups." Tourism management 29.1 (2008): 116-126.

[8] Chen, Ja-Shen, Russell KH Ching, and Yi-Shen Lin. "An extended study of the K-means algorithm for data clustering and its applications." Journal of the Operational Research Society 55.9 (2004): 976-987.