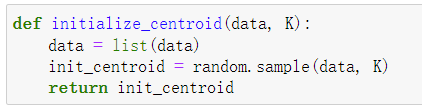
**2 Programming**

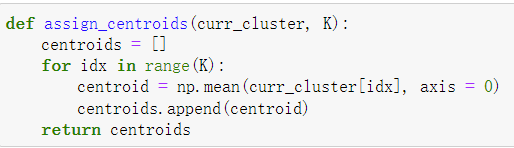
**2.1 K-means and GMM Implementation**

**2.1.1 K-means**

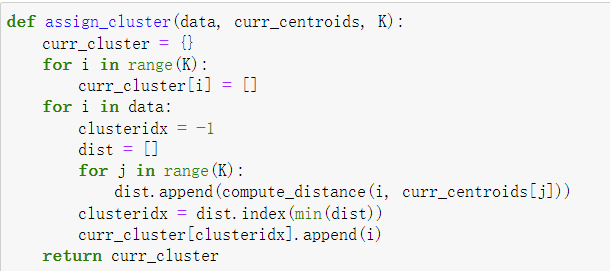
For K-means, I use 4 functions to do the clustering, they are initialize\_centroid, assign\_centroids, assign\_cluster and fit\_kmeans.



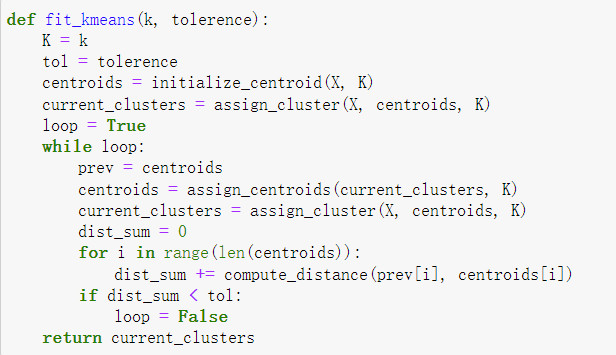
For initialize\_centroid, I randomly select 3 samples from the whole dataset to be the initial centroid.



For assign\_centroids, the current clusters generated by last loop should be the input, and we just calculate the mean value of the points in each cluster, then assign the mean value to the new centroids.



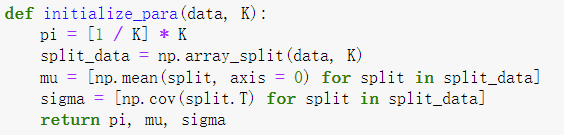
For assign\_cluster, we first generate an empty newly cluster, and iterate through all of the datapoints in the original dataset, compute the distance from each datapoint to the three newly generated centroids, then we grab the index which makes the distance from the datapoint to the centroids the smallest, to assign the datapoint to the new cluster, finally return the newly generated cluster.



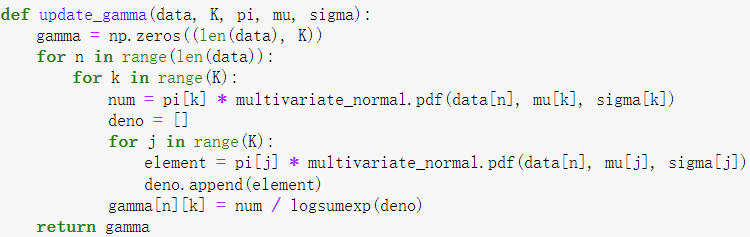
For the fit\_kmeans, we first initialize the centroids, and assign the initial clusters, and start a loop, which stops when the distance between the centroids from the last loop and the new loop is smaller than the tolerance. Finally we output the final clusters.

**2.1.2 GMM**

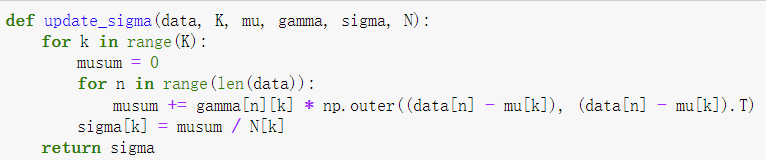
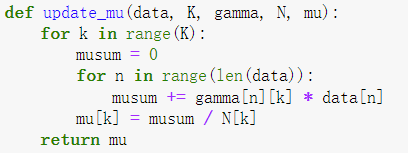
For GMM, I use 6 main functions to do the clustering. They are initialize\_para, update\_gamma, update\_mu, update\_sigma, predict and fit\_GMM.



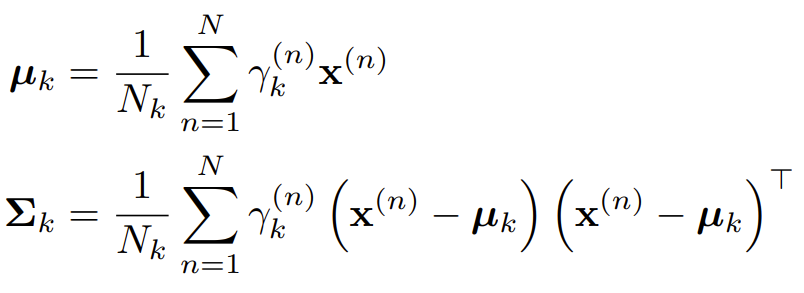
For initialize\_para, I set the pi to be uniformly distributed probabilities, mu to be the mean value of each split from the whole data, sigma to be the covariance matrix of each split of the whole data.

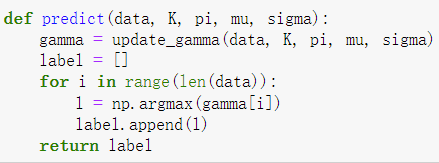


For update\_gamma, I use the formula to update it.

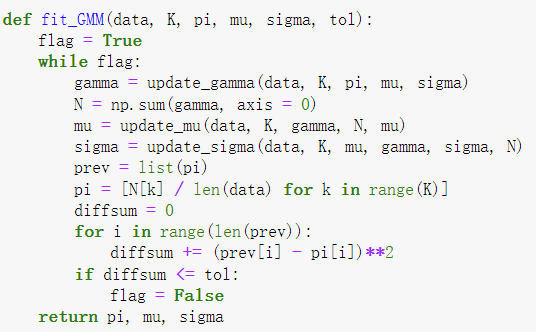


For update\_mu and update\_sigma, I use the formulas below to update them.





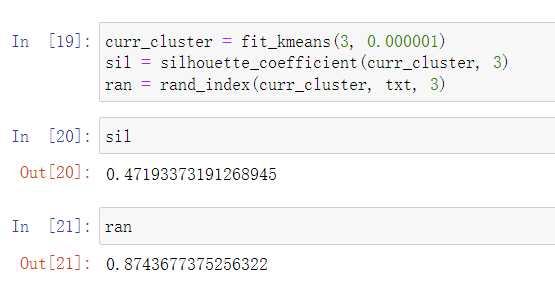
For predict, I choose the index which has the largest gamma value to be the cluster that it belongs to, and output the current labels.



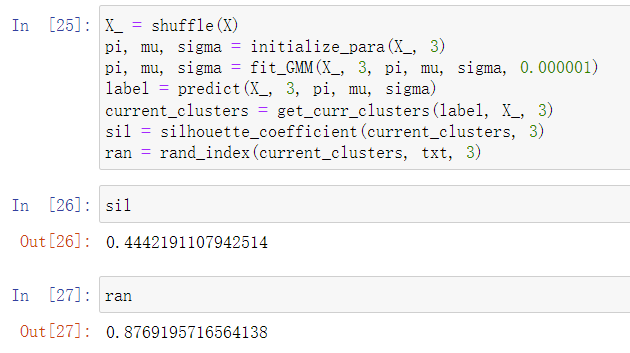
For fit\_GMM, I just create a loop to iterate the whole process, to update the parameters each time. The stopping criterion is that, when the distance between the current and in the last iteration is smaller than or equal to the tolerance. Then we finally output the last value of the three parameters.

**2.2 Silhouette Coefficient and Rand Index**

We calculate silhouette coefficient and rand index for Kmeans clustering:



We also calculate silhouette coefficient and rand index for GMM clustering:



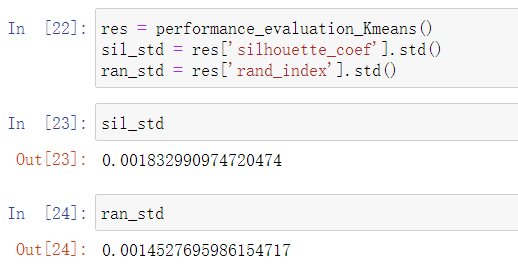
From the results, we can see that for the two distinct clustering algorithms, for silhouette coefficient, Kmeans > GMM. Since silhouette coefficient represents the distance between a point to its same cluster and its nearest cluster, if the distance between the point and the points in the same cluster is larger, the performance for the clustering algorithm should be good. So the performance of Kmeans is better than GMM judged by the silhouette coefficient.

However, for the rand index, we can see that GMM > Kmeans, rand index is useful when the true label is provided, if the rand index is larger and closed to 1, we say that the clustering algorithm generated a result that is closed to the ground truth. Thus if we judge the performance of these two algorithms by rand index, we can say that GMM has better performance than Kmeans.

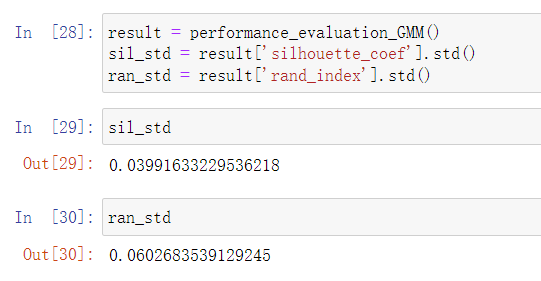
**2.3 The sensitivity to the initialization of Kmeans and GMM**

For each algorithm, we randomly assign the initialization, and run the two algorithms for ten times each.

We calculate the standard deviation of the two evaluation metrics for Kmeans:



We calculate the standard deviation of the two evaluation metrics for GMM:



From the above results, we can observe that the standard deviations for these two evaluation metrics are quite small, which further implies that these two clustering algorithms works properly with different initializations.