

ASSIGNMENT 2: E-M ALGORITHMS

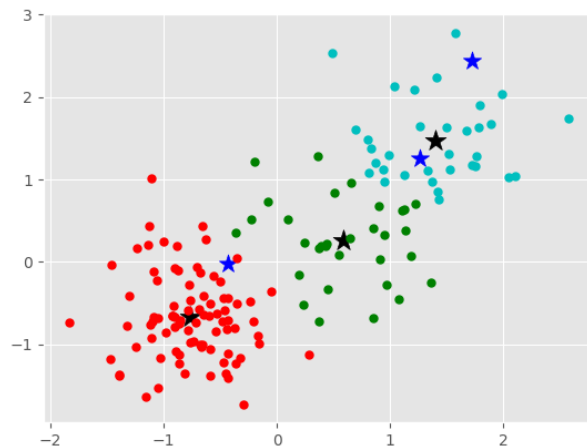
Part 1: Implementation

All codes are written in Python 3.

In this assignment, we produced 2 different types of k-means clustering. The biggest difference is of their data structure. One of them use numpy.array, and the other one used pandas.dataframe. Both of them will repeat the cluster process five times to see if the result is optimum or not.

RESULT OF K-MEANS ALGORITHM 1 (CLUSTER.PY)

Blue stars are the initial centroid, and the black ones are the eventual means of each group.



Index	0
0	False
1	False
2	False
3	False

.....

146	True
147	True
148	True
149	True

The Data Structures Used

I used pandas Dataframe to store input data, and a little bit of numpy array to efficiently do some calculation.

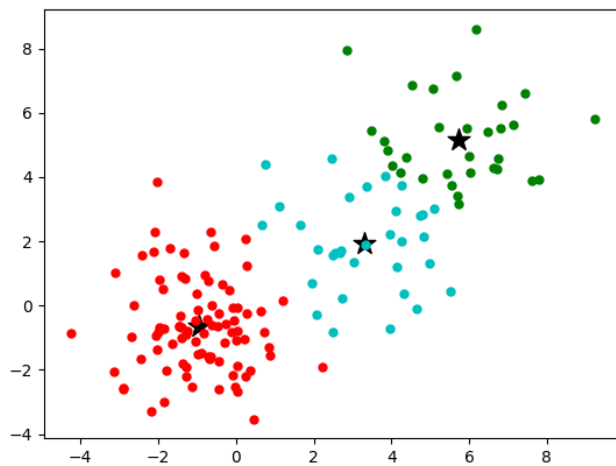
Code Level Optimization

1. Iterate through columns instead of elements, therefore the program complexity decreases.
2. Construct a boolean bitmap implying the group to which the data points are assigned. For example, look at the table on the left, the header is 0 which means it represents data point membership of group 0. This bitmap is a component to elegantly compute mean:

$$\text{new} = \Sigma (\text{bitmap} * \text{data}) / \Sigma \text{bitmap}$$

3. To avoid local minimum, the algorithm will run five times.

RESULT OF K MEANS ALGORITHM 2 (KMEANSGMM.PY)



The Data Structures Used and Description

I use numpy array to store all the data points in K-means.

Firstly, I initialize 3 centroids randomly, and then I use two steps to create three clusters:

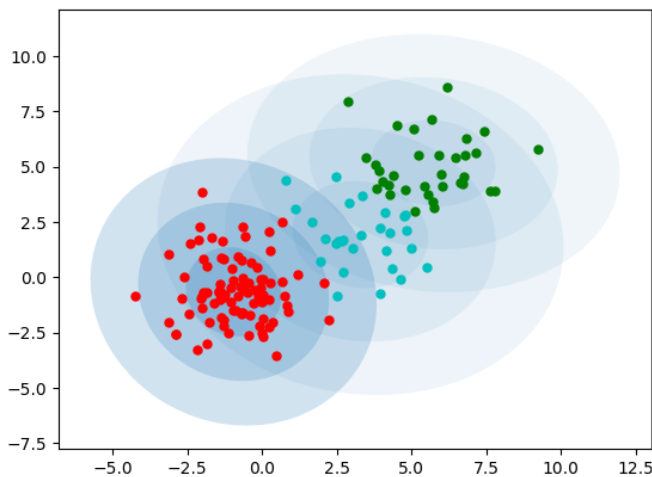
Step 1: I compute the distance between the centroids and each data point, assign points to each cluster based on its minimum distance, and

record its belonged clusters.

Step 2: I re-compute the centroids for each cluster.

Finally, I repeat two steps until no more data points move among clusters.

RESULT OF GAUSSIAN MIXTURE MODEL (KMEANSGMM.PY)



The Data Structures Used and Description

Firstly, to initialize, I use the cluster result of K-means. For example, if *data point i* belongs to *cluster c* in K-means algorithm, then in GMM, I set $R_{ic}[i][c]$ into 1; otherwise, $R_{ic}[i][c]$ is 0. R_{ic} is the matrix storing possibility of each data point in which group is allocated. R_{ic} matrix also calculates π . Then two steps are used to create three clusters.

Step 1: Using R_{ic} and π to compute μ_c and Σ_c

Step 2: Calculating probabilities of each data points in all k Gaussian functions, re-computing the R_{ic} and π , and assigning each point to its new cluster with maximum $R_{ic}[i][c]$, $\{j \in 0-k\}$.

Finally, repeating these 2 steps until no more data points move among clusters.

The Challenges

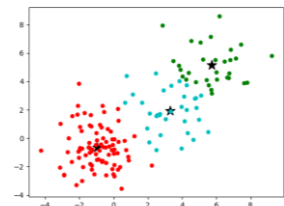
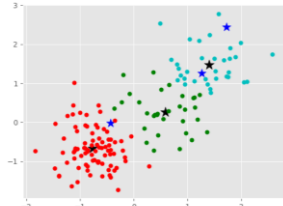
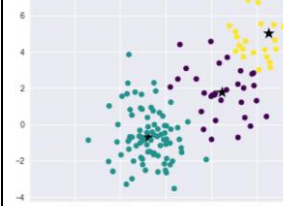
The end condition in GMM I set is that if less than 5 data points change their clusters, I will end the loop. The reason why I use Maximum Likelihood Estimation is that the covariance matrix contains numbers that are much smaller than others, which results in the singular matrix error. This situation equals to covariance matrix containing 0, and thus inverse covariance matrix (Σ) won't exist. This is the largest challenge I met. So, I changed the end condition.

How to Improve My Code:

The library GMM and K-means functions are faster than mine. So, I should use a better data structure like pandas in python instead of numpy array. It's much easier and faster for the matrix processing. It will efficiently decrease the running time.

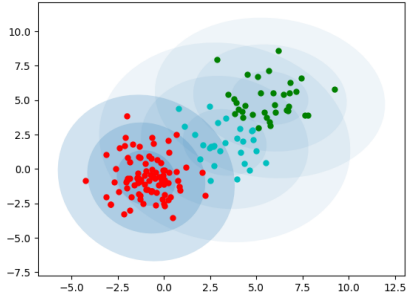
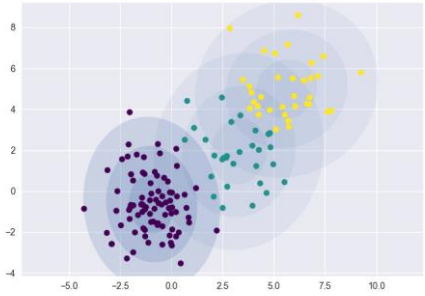
Part 2: Comparison

K-Means Clustering

	KmeansGMM.py	cluster.py	sklearn.cluster.KMeans
Operation Time(sec)	0.047 sec	0.027 sec	0.02 sec
Output Result			
Conclusion	Comparing to the package, the outputs are almost the same as package. Furthermore, the running time of the module cluster.py is almost as fast as the package.		

Gaussian Mixture Model

	KmeansGMM.py	sklearn.mixture.GaussianMixture
Operation Time(sec)	0.29sec	0.269sec

Output Result		
Result Stability	Sometimes it would generate weird results, such as groups that are the same and cover the whole dataset.	No matter how many times it have been executed, the results are basically the same.
Conclusion	Our code is unstable and a bit slower. In contrast, the package is much reliable and fast..	

Part 3: Applications

K-means

Rather than defining groups before looking at the data, clustering allows us to find and analyze the groups that have formed organically. Besides, we could use different clusters to classify its features, then use these features to create business opportunities. For example, stores can use K-means to classify each customer with some similarity in same groups, and then each group represents different purchasing behavior. It would be an efficient and suitable way to assist stores to target their main customer groups and bring profits.

Gaussian Mixture Model (GMM)

In Background Subtraction research, GMM is a common way to establish background image processing since the pixel value of image is not fixed. There are two reasons that the pixel value is not fixed.

1. When an actual object is moving such as a walking man, a floating cloud, winded leaves and so on, these moving behaviors bring some changes making the pixel value unfixed.
2. In static images, even if there are no moving objects, due to the influence from the external environment, it still generates brightness changes. Take sunlight as an example, when observing the sea, we would notice that there are distinct colors reflecting from sunlight during different time or viewing points.

These two changing reasons result in small change in the pixel value. So, it would be a suitable way to take advantage of GMM to simulate color distribution in the background image.

As a conclusion, either K-means or GMM is the way to define clusters. However, GMM is much better than K-means since it is required to consider Normal Distribution in GMM, which makes the result much closed to the real situation and more convincing.

GROUP MEMBER

Jung-Kang, Su	2389753352	Research for application, implement packages, edit
Yuhsi Chou	6048573191	Write cluster.py, visualize cluster result, edit
Zhang Mujie	2621330761	Main author of KmeansGMM.py

References:

<https://ir.nctu.edu.tw/bitstream/11536/68068/7/251107.pdf>

<https://www.datascience.com/blog/k-means-clustering>