Lab Seven: K-Means Clustering and Mixture Gaussian Models

name: Huizhi Yu
number: ********
email: ********

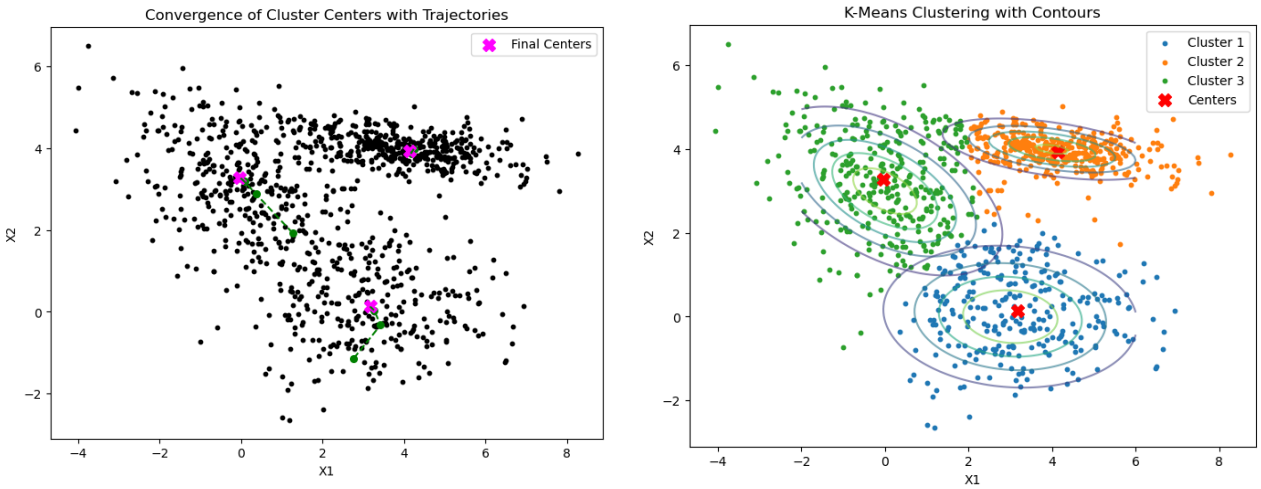
1 K-Means Clustering

1.1 achieve K-Means Clustering

To achieve K-Means Clustering, we first sample data from a mixture Gaussian density. Based on the provided code snippet (Appendix), data is generated with randomly set means, covariances, and proportions, forming a Gaussian mixture model with $K = 3$.

Following the equations 9.2 and 9.4 from Bishop PRML, the K-Means algorithm is achieved. The algorithm is then executed and converged at iteration 5, producing **Figure 1**, which illustrates the clustering results and the convergence process (including the trajectories of the cluster centers).

By plotting the contour lines of the probability density function, we can visually observe the distribution characteristics of the data points. Comparing these contours with the K-Means clustering results, we find that areas of high density generally correspond with the positions of the cluster centers.



(a) (b)

Figure 1: K-Means Clustering

1.2 Comparison between results with the Kmeans clustering algorithm in sklearn

As shown in **Figure 2**, under identical initial conditions, the clustering results of both implementations are essentially consistent, indicating that the custom algorithm aligns functionally with the implementation in scikit-learn.

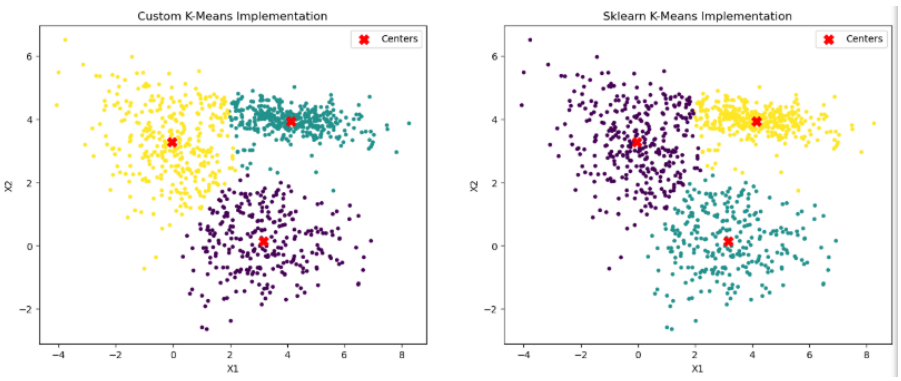


Figure 2: Comparison

1.3 Online K-Means and Batch K-Means

To implement the online version of the K-Means algorithm, we update the cluster centers using a stochastic approach. The online algorithm updates the center for each data point sequentially using the following formula:

$$\mu_k^{\text{new}} = \mu_k^{\text{old}} + \eta(x_n - \mu_k^{\text{old}})$$

where η is the learning rate. We compare the objective function J of the online version with the batch version as the number of iterations increases.

With a learning rate of $\eta = 0.2$, the J value of the online algorithm decreases rapidly in the initial iterations but stabilizes later. The batch algorithm exhibits a more gradual and steady decrease in J, ultimately converging to a slightly lower value compared to the online version. Both algorithms achieve good convergence in the end, demonstrating their effectiveness.

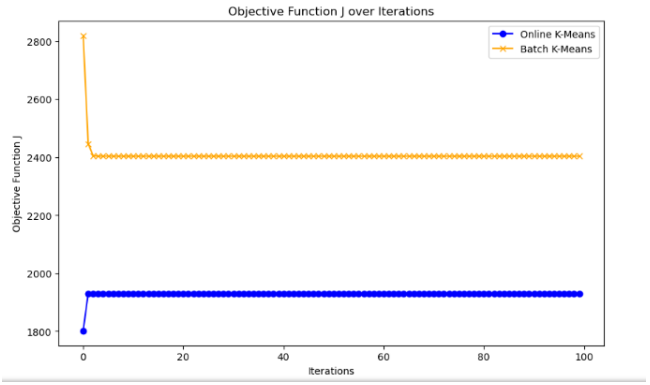


Figure 3: Comparison

1.4 Failed Initialization

As shown in Figure 4, in the case of **Failed Initialization**, the cluster centers are initialized far from the actual data clusters, leading to incorrect clustering results. In contrast, in **Random Initialization**, the cluster centers are correctly distributed near the centers of the data clusters, resulting in reasonable clustering outcomes. Therefore, selecting inappropriate initial centers can prevent the objective function from being fully optimized, resulting in inaccurate clustering. Similarly, if the value of K is chosen improperly, the clustering results may also fail.

In conclusion, the K-Means algorithm is highly sensitive to the initial guesses for the cluster center positions and the choice of K.

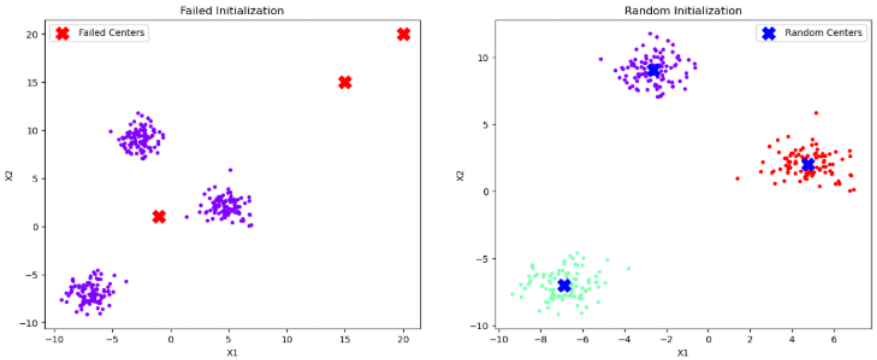


Figure 4: Failed Initialization

1.5 Wine dataset from the UCI repository

We selected the Wine dataset from the UCI repository and applied K-Means clustering after reducing its dimensionality using PCA. The clustering results, visualized in a two-dimensional space, showed that the K-Means algorithm effectively separated the data into three clusters, corresponding to the three classes in the dataset. The cluster centers were reasonably positioned near the data clusters, as indicated by the red markers in the K-Means plot. Evaluation metrics such as Adjusted Mutual Information (AMI = 0.87) and Normalized Mutual Information (NMI = 0.88) demonstrate a high correlation between the clustering results and the true labels. Despite the overall effectiveness, some points were misclassified, which could be attributed to overlapping clusters or the information loss from dimensionality reduction. Overall, the experiment confirms the applicability of K-Means clustering to the Wine dataset.

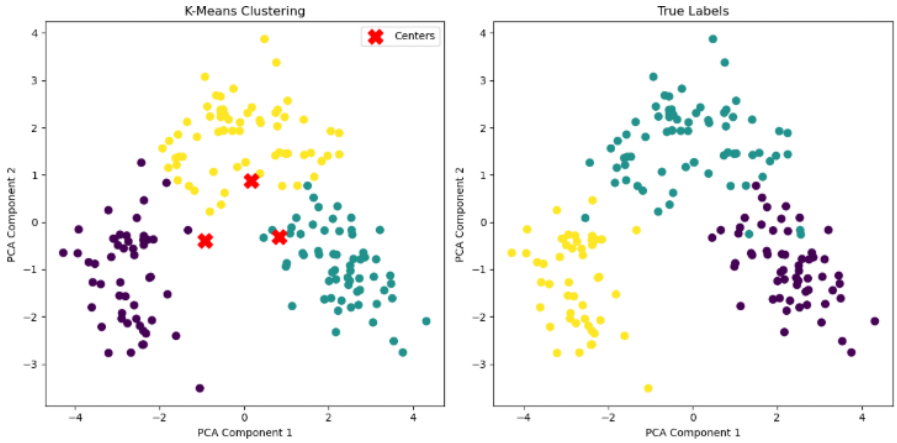


Figure 5: Wine dataset from the UCI repository

2 Gaussian Mixture Models

The Gaussian Mixture Model (GMM) fitting results demonstrate the effectiveness of the algorithm in capturing the underlying data distribution. In the figure, the blue points represent the synthetic data sampled from a mixture Gaussian density, while the red, blue, and green ellipses illustrate the GMM-fitted Gaussian components. Each ellipse corresponds to one Gaussian component, with its center representing the estimated mean and its shape and size defined by the covariance matrix. The fitted ellipses align well with the data clusters, indicating the GMM's ability to model the data accurately.

The parameter variations across multiple runs are analyzed through the Mean Squared Error (MSE) of the estimated means, covariances, and weights. The MSE for the means ranges from 0.0007 to 5.69, with minimal error observed in the third run due to favorable initialization. The covariance matrices exhibit MSE values between 0.0025 and 0.0052, reflecting high stability in estimating the distribution's shape. The weights, which represent the proportion of each component, show very small MSE values between 2.8e-05 and 0.0073, confirming the model's consistency in capturing the mixture proportions.

In summary, the GMM fitting procedure achieves robust and accurate results. Despite some variability across runs due to the randomness in data sampling and parameter initialization, the model consistently produces low estimation errors. This demonstrates the stability and reliability of the GMM for modeling Gaussian mixture distributions. Future improvements, such as optimized initialization methods or increasing the sample size, could further reduce parameter estimation errors and enhance the model's performance.

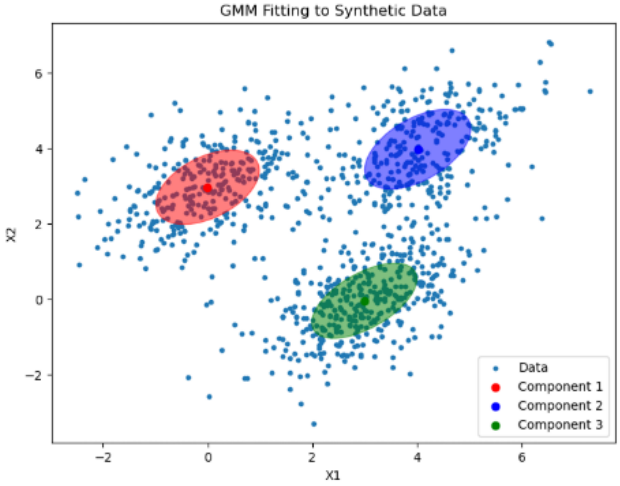


Figure 6: Gaussian Mixture Models