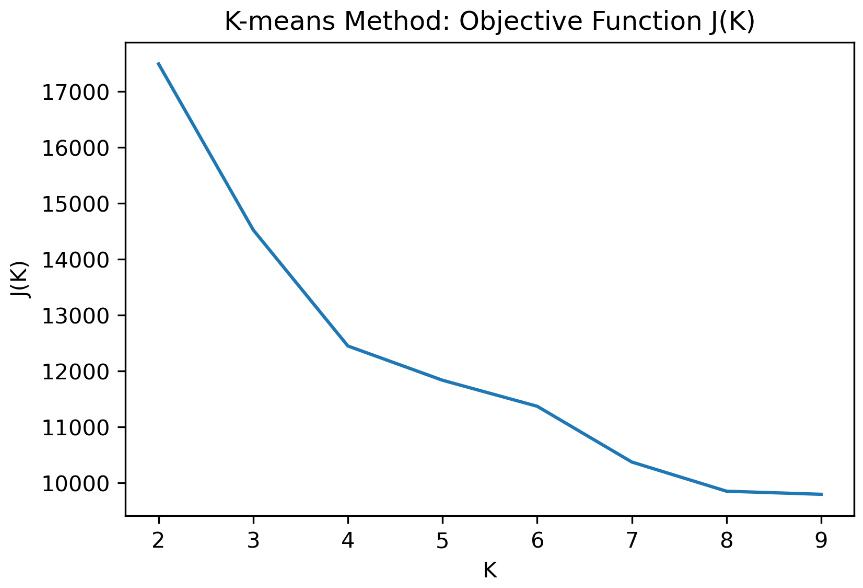
Problem 2. K-Means Clustering

Introduction: Clustering is an unsupervised learning technique that groups similar data points together based on their similarities. The K-means algorithm is one of the most popular clustering algorithms that separates data points into K distinct clusters. In this implementation, we apply the K-means algorithm on a data set of 128 data points, each with 13 features.

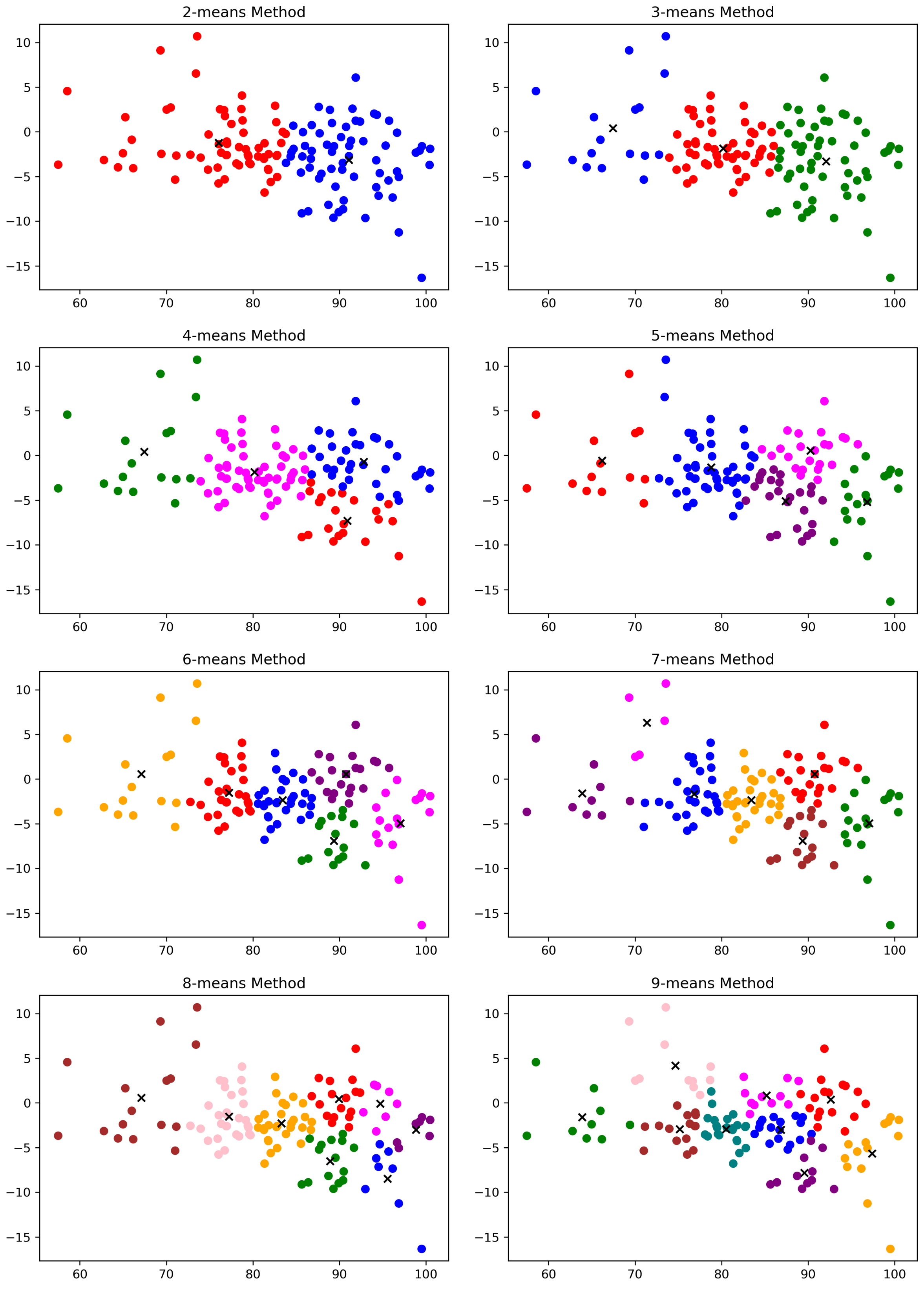
Methodology: We implement the K-means algorithm from scratch and run it with $K = 1, 2, \cdots, 9$. For each run, we initialize the cluster centers randomly among the given data. We then plot the objective function as a function of K and use only the first two features to plot the points for $K = 1, 2, \cdots, 9$ with different colors. Each center is plotted using a black $\times$.

Results and observations: The following figure shows the values of the objective function for different values of K where the objective function J is $J(K) = \sum_{n=1}^{128} \sum_{k=1}^{K} r_{nk} \|x_n - \mu_k\|^2$:



The objective function decreases as K increases, which is expected because increasing K means more clusters and hence a better fit to the data. However, the decrease in objective function becomes smaller as K increases, which indicates that there is a diminishing return to increasing K beyond a certain point.
The optimal value of K based on the elbow method appears to be 4. The elbow point is where the slope of the objective function curve changes significantly, and we can see that this occurs around $K = 4$. This suggests that this value of K provide a good balance between model complexity and fit to the data.
We also observe that the clusters become more distinct as K increases, which is expected because more clusters mean that the algorithm can assign data points to more specific groups.
It was observed that the K-means method yields varying classifications upon multiple runs, which indicates that the initial selection of centroids is the sole factor responsible for these

Figure 1: K-means classifications for different values of K



differences. Therefore, K-means method final classification can vary depending on the initial selection of centroids, and it is important to run the algorithm multiple times with different initial centroids and choose the classification with the lowest within-cluster sum of squares (WCSS) or another relevant evaluation metric. Figure 1 shows classification of data set for all value of K between 2 and 9.

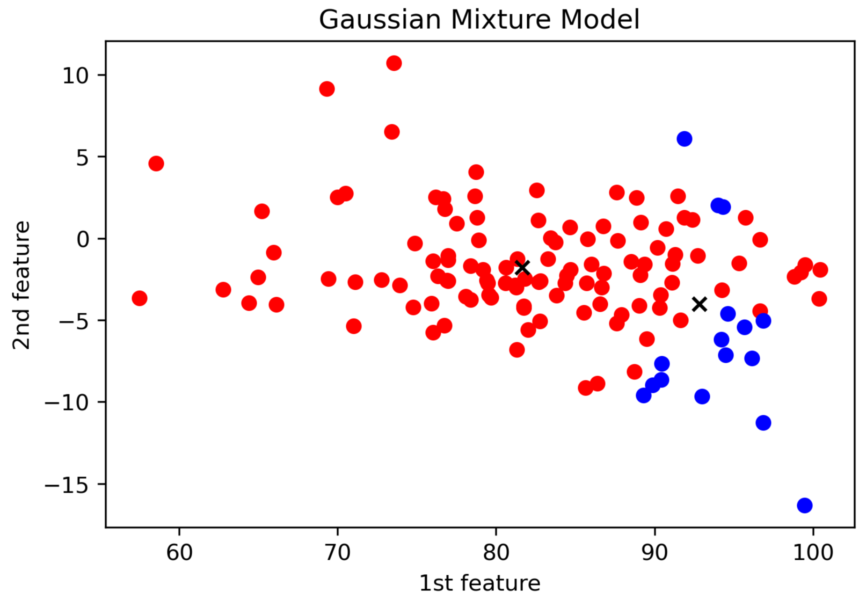
Conclusion: In conclusion, we have successfully implemented the K-means algorithm from scratch and employed it to cluster a data set consisting of 128 data points with 13 features. Our experimentation has shown that increasing K leads to a decrease in the objective function, but this reduction becomes negligible beyond a certain point. Utilizing the elbow method, we determined that the optimal number of clusters is either 4. This finding indicates that K=4 provides an optimal trade-off between model complexity and the goodness-of-fit to the data. Additionally, we noted that the initial selection of centroids can influence the final classification, leading to some variability in the results.

Extension: The code is implemented to be able to run for any number of clusters between 2 and 9. (See Figure 1)

Attachments:

- HW03_23.py the Python code for this question, and next question.
  - Since the question 3 is using the result of this question, I have combined both codes.
  - The code from the beginning up to line 153 corresponds to question 2, and the remaining code pertains to question 3.
  - If someone wishes to skip running the code for question 3, they can comment out all the relevant code starting from line 157 to the end, and then uncomment the "plt.show()" code at line 152. This will allow them to display the plot associated with question 2 while bypassing the execution of the code for question 3.
  - The number of clusters, the number of dimensions, and the threshold for stopping the EM algorithm can be set at lines 158, 159, and 160 respectively.
- results_2_features.txt : Including the calculated centers for all values of $K = 2, 3, \cdots, 9$, based on 2 features.
- results_13_features.txt : Including the calculated centers for all values of $K = 2, 3, \cdots, 9$, based on 13 features.

Figure 2: Gaussian Mixture Model with 2 clusters



Problem 3. Gaussian Mixture Model (GMM)

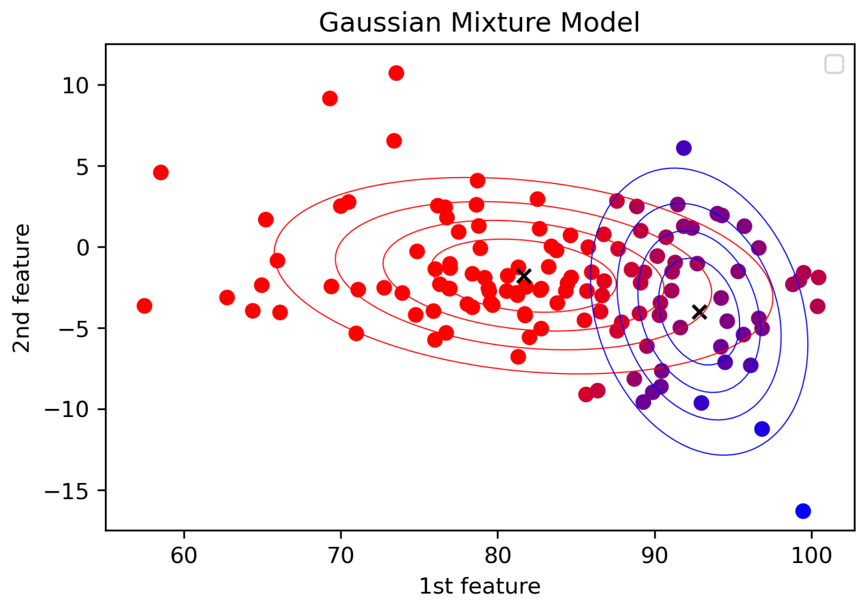
We implemented the Gaussian Mixture Model (GMM) from scratch in Python and ran it for two different values of K, i.e., K=2 and K=13, where K represents the number of features in the data set. The EM algorithm was used as a reference for the implementation, and we used the results of the K-means method to initialize the parameters. The GMM terminated after the change in the log-likelihood was smaller than 10^{-5}, indicating that the EM algorithm successfully converged to an optimal solution. This suggests that the GMM was able to estimate the parameters of the Gaussian distributions for each cluster accurately and converge to a stable solution. After conducting multiple runs of the code with 5 clusters in Gaussian Mixture Model (GMM), it has been observed that the classification results vary depending on the initial parameters.

For K=2, we plotted the points using the first two features of the data set and used blue and red colors to distinguish between the two clusters. (see Figure 2) However, we observed that the scatter plot showed overlapping of clusters. This suggests that there may be some regions in the data set where the data points from both clusters exhibit similar feature values. This is expected in cases where the clusters are not perfectly separated or when there are some shared characteristics between the two clusters. We see that the points in between the two clusters have a close posteriors. This resulted in points lying in overlapping classes where the decision boundary between the two clusters was not well-defined.

$$\mu_0 = (92.84, -4.01)^T$$
$$\mu_1 = (81.65, -1.79)^T$$

To gain further insights into the distribution and shape of the clusters, we plotted the contour

Figure 3: GMM with 2 clusters and Normal pdf contour levels with color mixing



levels of the normal distributions around each cluster's center.(see Figure 3) This allowed us to visualize the probability density of the Gaussian distributions in the GMM and provided a better understanding of the shape and extent of each cluster.
Additionally, we noticed that the absence of a strict boundary, as seen in the K-means method, was expected in GMM due to the probabilistic nature of the model. GMM assigns probabilities to each point for belonging to each cluster based on the posteriors, which can result in softer cluster boundaries with overlapping clusters.
To better represent the regions where the clusters blend together, we mixed the colors for points lying between clusters, using purple color and posteriors ratios in our visualization. This helped convey the uncertainty or overlapping nature of those points. In summary, our results from the GMM implementation showed overlapping of clusters in the scatter plot, which can be attributed to the close posteriors assigned by the GMM to those points. Contour plots of the normal distributions provided insights into the distribution and shape of the clusters. The absence of strict borders in GMM, compared to K-means, was observed due to the probabilistic nature of the model. Color mixing techniques, such as using purple color for points between clusters, helped better represent the classes of uncertainty or overlapping. We also observed that the classification in GMM depends on the initial parameters

Extension: As an extension, the code has been implemented to run for any number of clusters from 2 to 9. Simply, one can change the number of cluster at line 158. For example see Figure 4 and Figure 5.

Figure 4: two different clustering for data set with 5 clusters

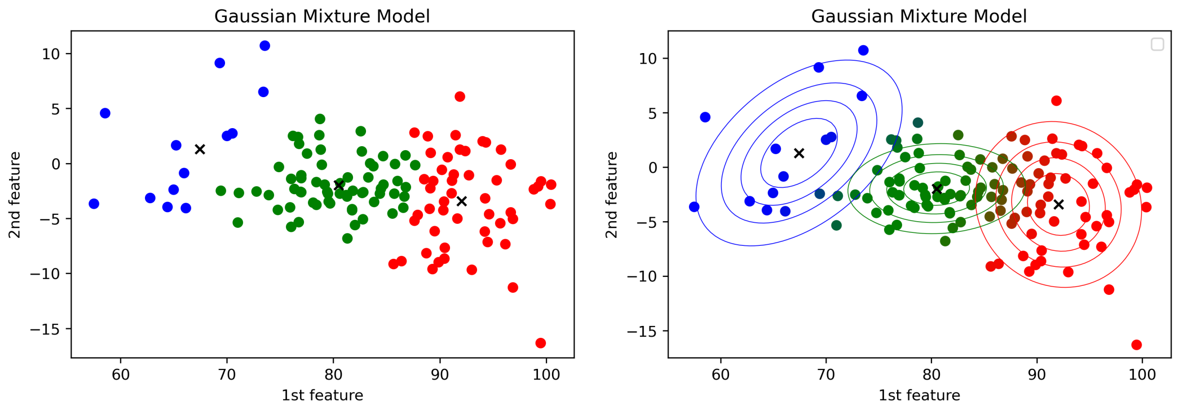
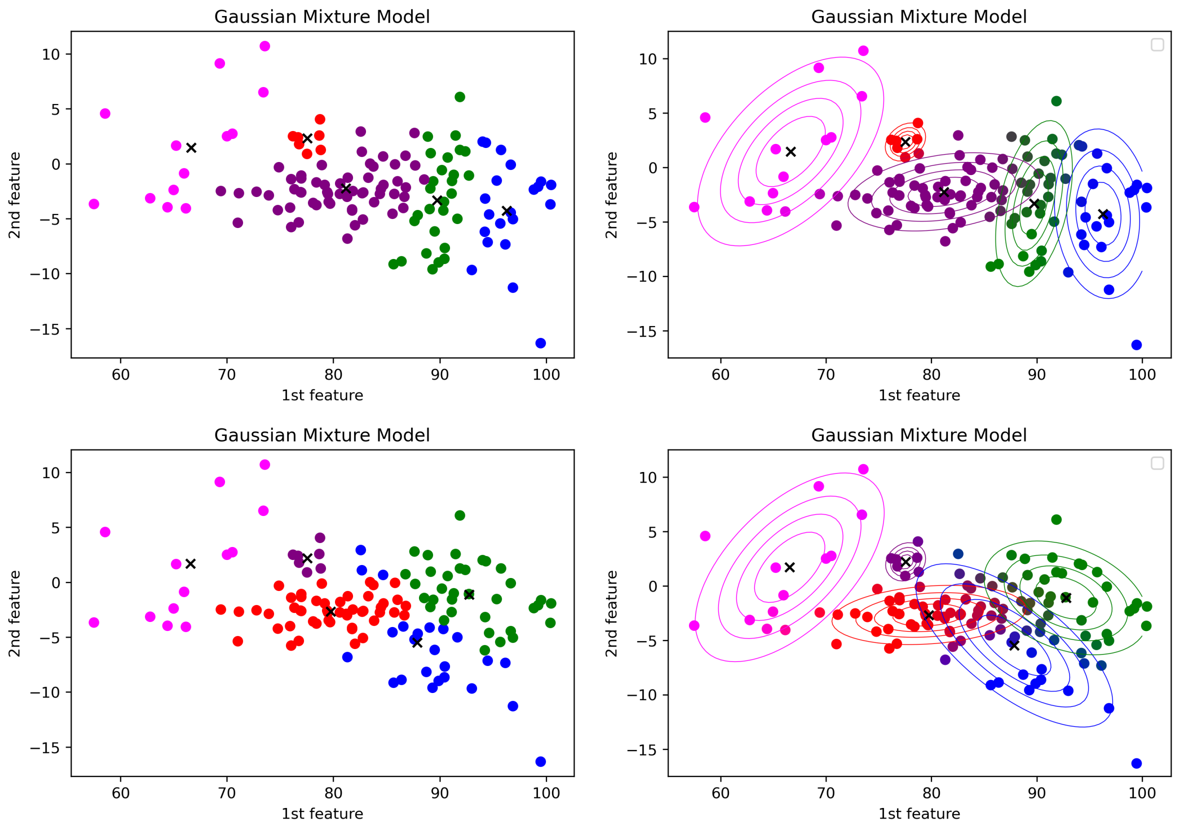


Figure 5: two different classification with 5 clusters



Attachments:

- `HW03_23.py` the Python code for question 2 and question 3.
  - Since the question 3 is using the result of question 2, this code include question 2 code as well.
  - Dimension (number of features) can be set at line 159 to any integer between 2 and 13.
  - The number of clusters are set to 2 by default as instructed by the question. However, that can be set to any integer between 2 and 9.