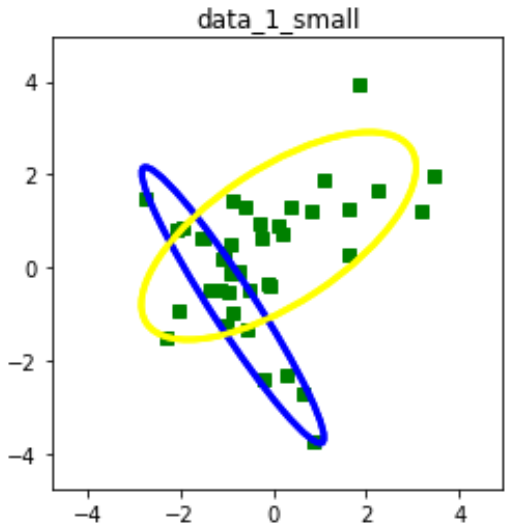
Jason Teng
Programming Assignment 3 Writeup

1. EM Algorithm

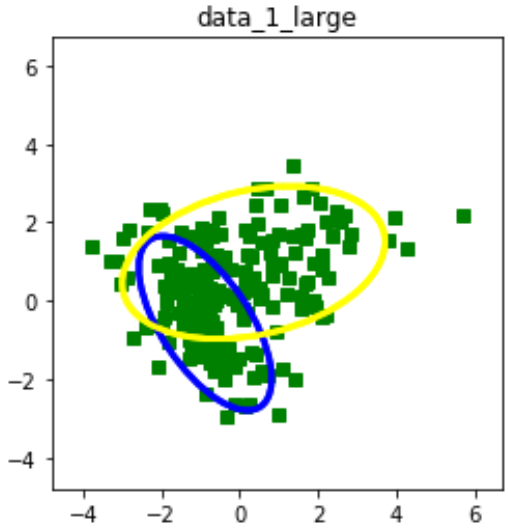
The convergence criteria I used was to converge based on the log-likelihood of the input data vs my model. Since the log-likelihood of a GMM monotonically increases under EM to a local maximum, the difference in log-likelihood between consecutive steps in the EM algorithm is guaranteed to also converge to a local maximum.

In terms of performance, the EM algorithm takes considerably longer to converge on the large datasets, often taking up to 5 minutes to complete. The speed slows further as the number of components increases.

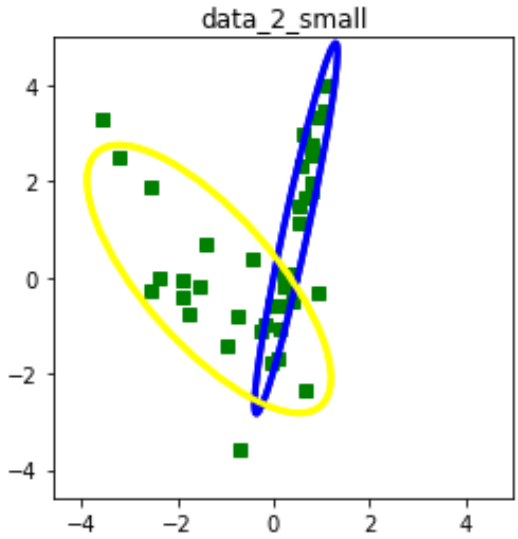
Here are the plots generated with 2 components:



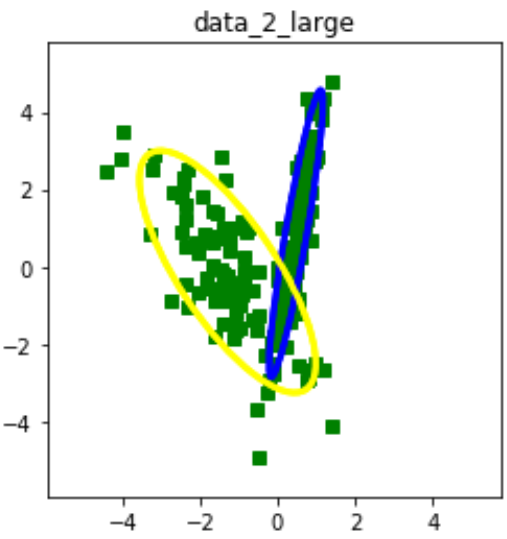
log-likelihood: -123.915955871



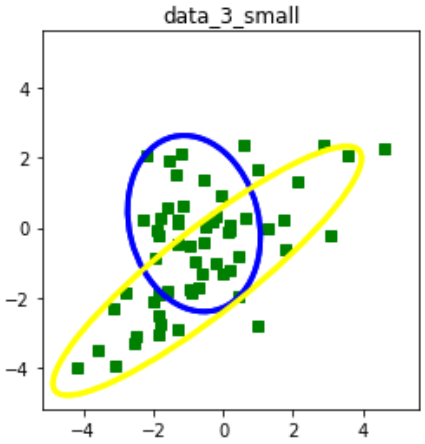
log-likelihood: -662.920781498



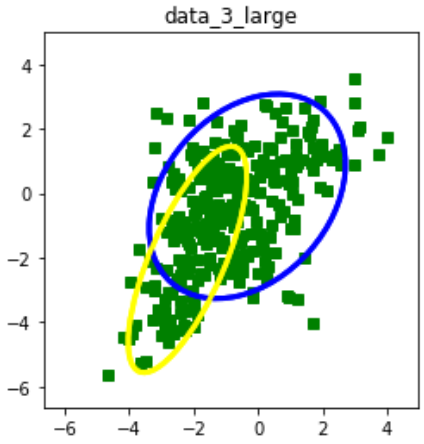
log-likelihood: -107.979504627



log-likelihood: -550.316559044

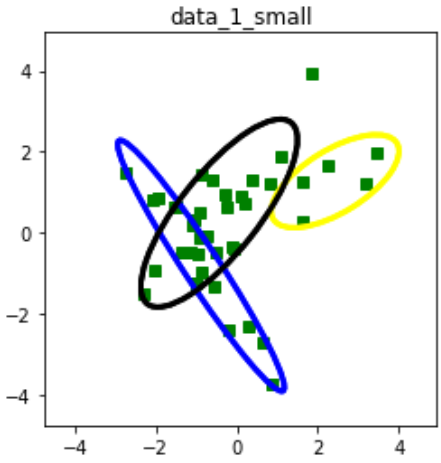


log-likelihood: -214.126537767

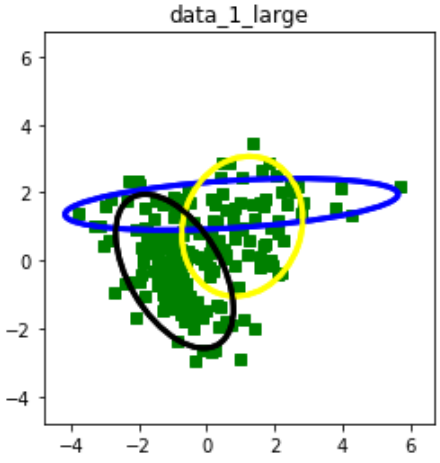


log-likelihood: -1114.52579241

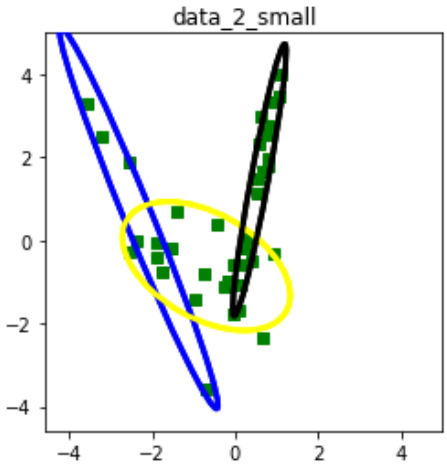
Increasing the number of components to 3:



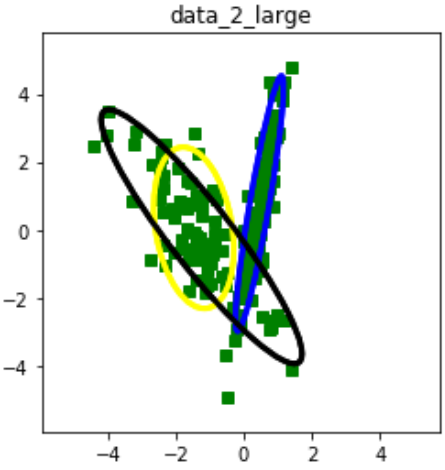
log-likelihood: -118.584522472



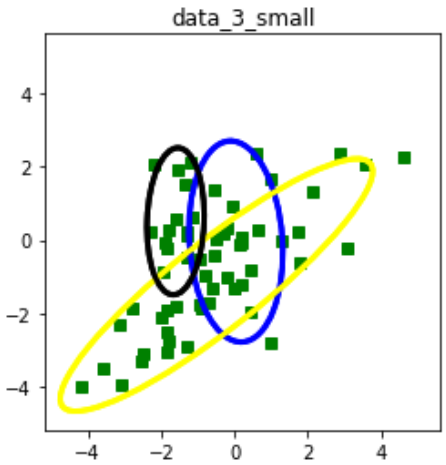
log-likelihood: -658.477442795



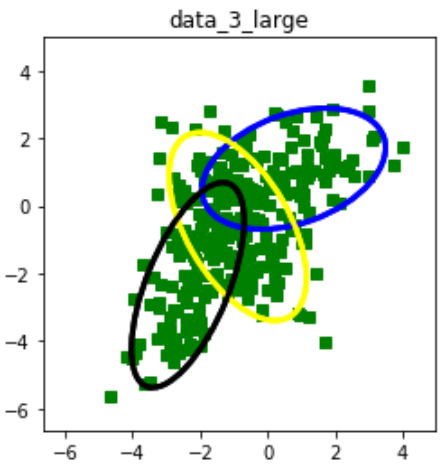
log-likelihood: -101.510843662



log-likelihood: -545.917833912



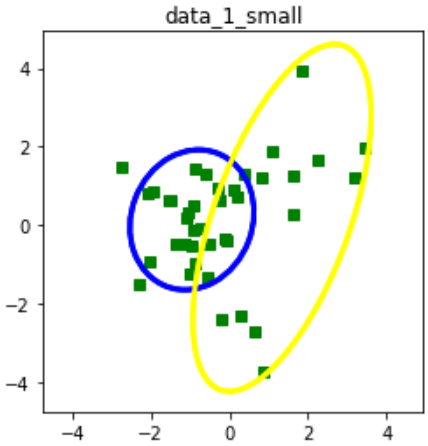
log-likelihood: -211.198203975



log-likelihood: -1088.44644364

One phenomenon I discovered was that if I initialized the covariance matrices with values that were too close to zero, then I would often get errors while running the EM algorithm. My solution was to initialize the covariance matrix with a large scale to minimize the probability of a zero value occurring in the matrix.

Increasing the threshold (initially set at 1e-9) predictably sped up the computation, while causing the results to be less accurate. For example, this plot was generated with a threshold of 1e-6:



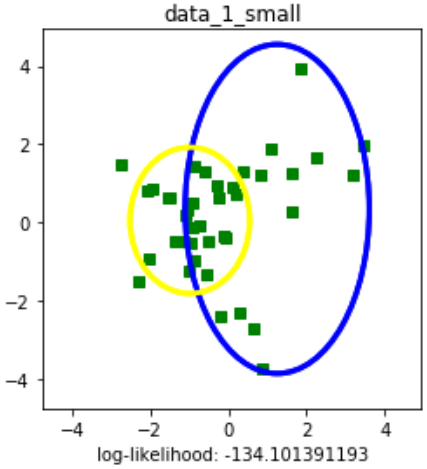
log-likelihood: -662.920788416

2. Variations

1. The primary difference between diagonal and full matrices is in the shape of the resulting Gaussian curve. With full matrices, the Gaussian curve can be an oval oriented with a long axis in any direction. With a diagonal matrix, however, the curves align with either the horizontal or vertical axis. In terms of implementation, the only difference in the algorithm is in the initialization (where the initial covariance matrix must be diagonal) and update of the covariance matrix (where only the diagonals of the covariance matrix are updated).

2. To implement K-means, I created a separate function that takes in the dataset and the number of clusters, and returns the means calculated through the K-means method. I then modified the EM algorithm to initialize the means using K-means rather than randomly sampled points.

3. When using diagonal matrices, the models predictably became much less accurate, as none of the data contained clusters with horizontal or vertical gaussian centers. Here is one example of a poor model generated with diagonal matrices:



3. Model Selection

Here is the ranking of models based on average log-likelihood:

Data_1:

1. 5 components, diagonal matrix (avg LL = -2.7)
2. 2 components, diagonal matrix (avg LL = -3.44)
3. 3 components, diagonal matrix (avg LL = -3.465)
4. 1 component, full matrix (avg LL = -3.49)
5. 2 components, full matrix (avg LL = -3.515)
6. 1 component, diagonal matrix (avg LL = -3.518)
7. 4 components, diagonal matrix (avg LL = -3.66)
8. 3 components, full matrix (avg LL = -3.87)
9. 5 components, full matrix (avg LL = -4.33)
10. 4 components, full matrix (avg LL = -4.4)

Data_2:

1. 2 components, full matrix (avg LL = -2.87)
2. 4 components, full matrix (avg LL = -2.97)
3. 3 components, diagonal matrix (avg LL = -3.27)
4. 2 components, diagonal matrix (avg LL = -3.34)
5. 5 components, diagonal matrix (avg LL = -3.41)
6. 4 components, diagonal matrix (avg LL = -3.45)
7. 3 components, full matrix (avg LL = -3.49)
8. 1 component, diagonal matrix (avg LL = -3.6056)
9. 1 component, full matrix (avg LL = -3.606)
10. 5 components, full matrix (avg LL = -3.72)

Data_3:

1. 1 component, full matrix (avg LL = -3.79)
2. 3 components, diagonal matrix (avg LL = -3.826)
3. 2 components, diagonal matrix (avg LL = -3.83)
4. 2 components, full matrix (avg LL = -3.85)
5. 3 components, full matrix (avg LL = -3.953)
6. 1 component, diagonal matrix (avg LL = -3.959)
7. 4 components, diagonal matrix (avg LL = -4.006)
8. 4 components, full matrix (avg LL = -4.03)
9. 5 components, full matrix (avg LL = -4.11)
10. 5 components, diagonal matrix (avg LL = -4.19)

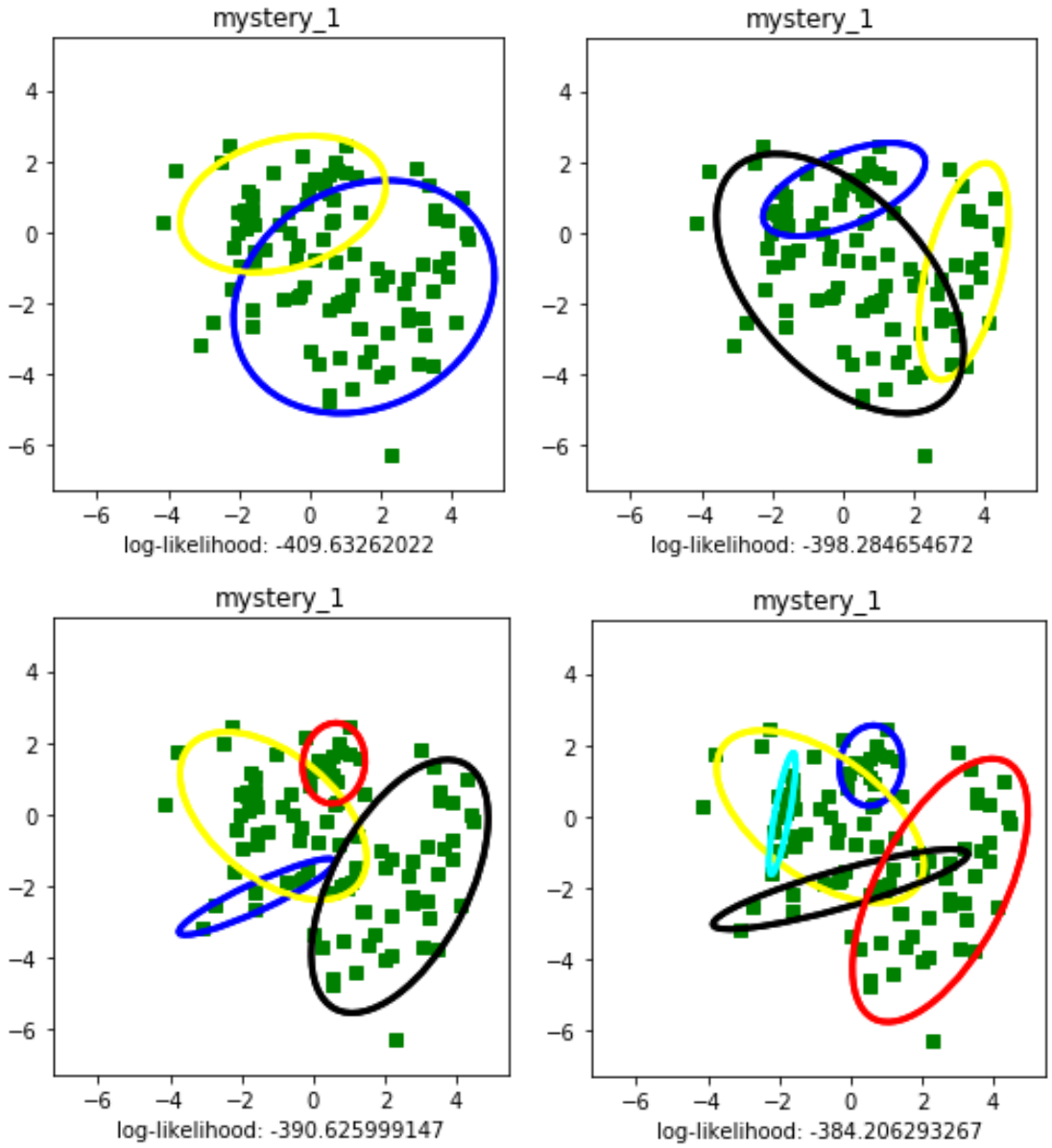
Rankings of k-cross validation:
Note: For some reason, I was experiencing errors when trying to run k-cross validations with 5 components, so I only ran k-cross for up to 4 components.
K = 5:
Data_1:
1. 4 components, full matrix
2. 1 component, diagonal matrix
3. 1 component, full matrix
4. 2 components, full matrix
5. 2 components, diagonal matrix
6. 4 components, diagonal matrix
7. 3 components, full matrix
8. 3 components, diagonal matrix

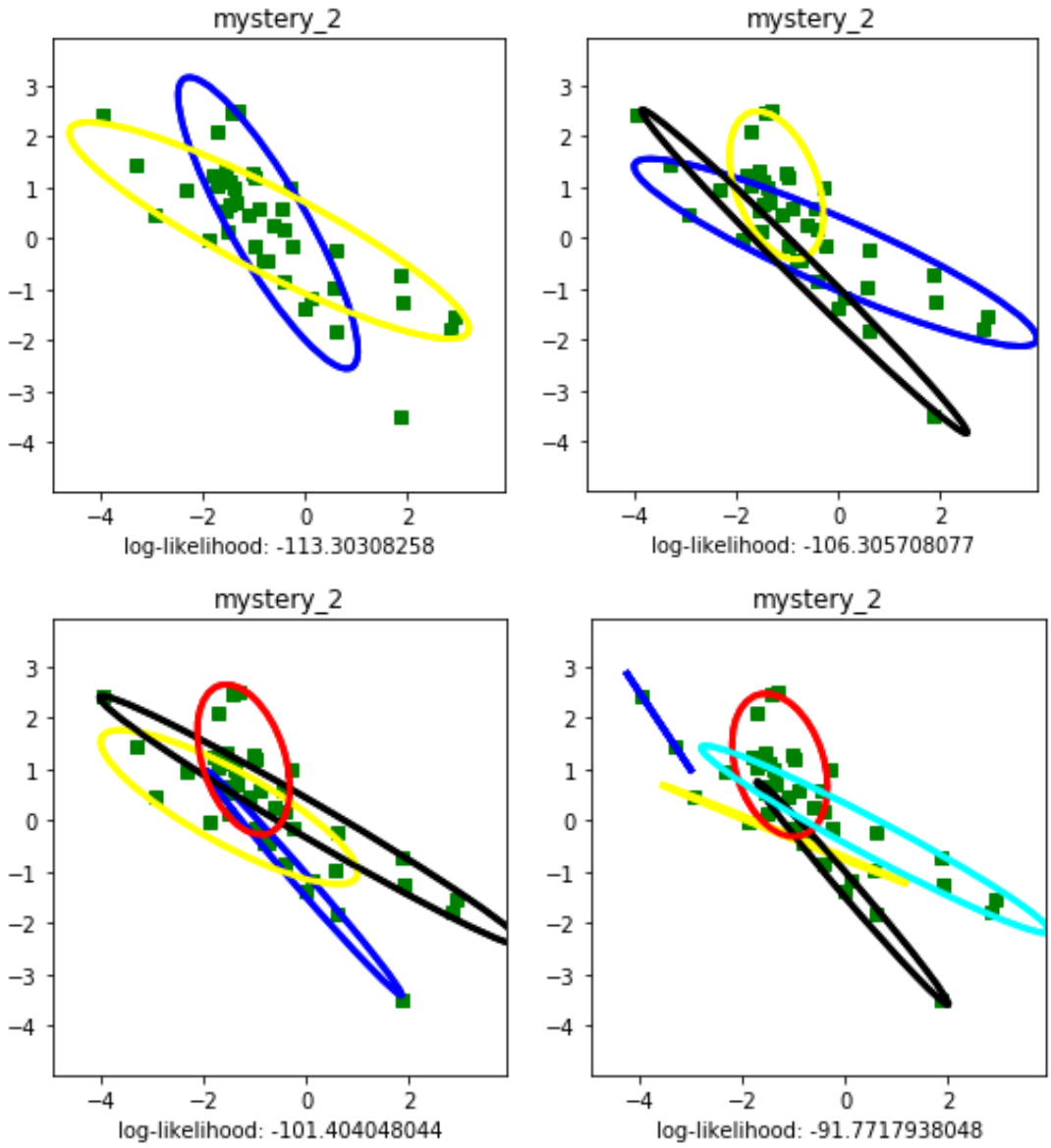
Data_2:
1. 4 components, diagonal matrix
2. 3 components, full matrix
3. 2 components, full matrix
4. 3 components, diagonal matrix
5. 1 component, diagonal matrix
6. 2 components, diagonal matrix
7. 1 component, full matrix
8. 4 components, full matrix

Data_3:
1. 4 components, diagonal matrix
2. 1 component, full matrix
3. 2 components, full matrix
4. 1 component, diagonal matrix
5. 2 components, diagonal matrix
6. 3 components, diagonal matrix
7. 3 components, full matrix
8. 4 components, full matrix

4. Predictions

Here are 5 graphs for each mystery data set, with the number of components ranging from 1 to 5:





Based on these graphs and their corresponding log-likelihoods, I think that mystery_1 is best fit with a GMM of 3 components, and mystery_2 is best fit with a GMM of 2 components.