Problem 1

Worked with Puya Gharahi

1. K-MEANS OUTPUT:

k = 12 mean = 1233.714809 variance = 32475.417484

k = 18 mean = 1058.719527 variance = 42097.109244

k = 24 mean = 912.744755 variance = 30301.055859

k = 36 mean = 835.758682 variance = 19020.551579

k = 42 mean = 743.754035 variance = 10913.512184

--- 4.195016185442607 minutes elapsed ---

2. When writing the code for this algorithm, there were several issues that I encountered. The primary issue was that my code would take a SIGNIFICANT amount of time to complete, so it was extremely difficult and time consuming to debug.

Here is some partial output for the GMM algorithm:

k = 12 mean = 5145.572891 variance = 46485.506672

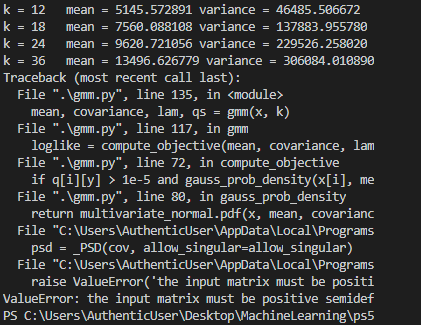
k = 18 mean = 7560.088108 variance = 137883.955780

k = 24 mean = 9620.721056 variance = 229526.258020

k = 36 mean = 13496.626779 variance = 306084.010890

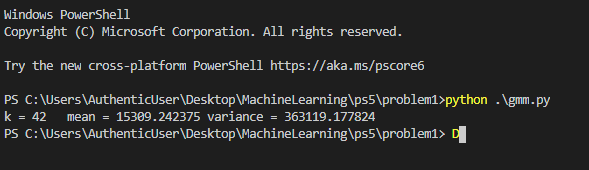
k = 42 mean = 15286.39354268349 variance = 565558.3699227025 (Courtesy of Puya Gharahi)

The algorithm crashed due to an unforeseen error which is shown here:



Parts of the message were cut off due to the vscode text editor truncating output when the terminal was idling. Luckily enough the program crashed after computing k = 36, so I have enough information to answer part 3.

UPDATE: I didn’t have time to completely rewrite this portion so I will just attach my own data for k = 42 since it finished computing recently.



3. Judging from the output given, I would prefer to use the k-means clustering algorithm as the variance on k = 36 is higher for the GMM than it is for the k-means algorithm. The higher variance is an indicator of the GMM having a difficult time finding good clusters given the same random initialization factors that the k-means algorithm has. However, the GMM algorithm is a superset of the k-means algorithm, and though it may take longer to execute, it should generally be able to produce better clusters than the k-means since it does not make assumptions on the shapes of the clusters, which the k-means algorithm always assumes circular type clusters. Even if the clusters are difficult to form, there should still be a difference in the variances of the two algorithms, since the GMM allows for uncertainty in where the datapoints should be assigned to. Though if enough datapoints are still fairly ambiguous in where they should belong (i.e. having roughly the same chance of belonging to several clusters) then the variance would be larger than having just hard assigned the datapoint to a single cluster.

4.

K-MEANS OUTPUT:

k = 12 mean = 1349.250883 variance = 100060.472855

k = 18 mean = 1148.784182 variance = 50519.792251

k = 24 mean = 953.969593 variance = 24450.987490

k = 36 mean = 777.388661 variance = 9421.251937

k = 42 mean = 771.310370 variance = 10281.994671

K-MEANS ++ OUTPUT:

k = 12 mean = 905.816598 variance = 1897.204150

k = 18 mean = 703.179827 variance = 975.917913

k = 24 mean = 560.720476 variance = 297.675276

k = 36 mean = 389.962608 variance = 198.218630

k = 42 mean = 344.116010 variance = 85.606957

GMM OUTPUT:

k = 12 mean = 5145.572891 variance = 46485.506672

k = 18 mean = 7560.088108 variance = 137883.955780

k = 24 mean = 9620.721056 variance = 229526.258020

k = 36 mean = 13496.626779 variance = 306084.010890

k = 42 mean = 15309.242375 variance = 363199.17824

GMM ++ OUTPUT:

k = 12 mean = 5582.266684 variance = 16660.574918

k = 18 mean = 7982.725727 variance = 43115.320409

k = 24 mean = 10161.021626 variance = 91003.707596

k = 36 mean = 14386.500999 variance = 70727.789959

k = 42 mean = 16087.810778 variance = 80620.262172

In the case of the K-MEANS, variance was reduced significantly, so the K-MEANS++ provided an improved clustering of like data.

In the case of GMM, the algorithm seemed to perform worse after applying the ++ initialization, so the original GMM should be preferred over GMM++.

5. To modify the EM algorithm to accept diagonal matrices with diagonal entries that are strictly greater than zero, we modify the M step updates for the covariance matrix update specifically. Since the initialization of the covariance matrix is already the identity, which satisfies the new requirements, that does not need to be modified. One way to modify it is to assign all non-diagonal entries to be 0 and keep the current method of updating, but the current method of updating does not guarantee that the diagonals themselves will be positive values. Those diagonal entries represent the variance of each Gaussian mixture component, and so to satisfy the diagonal restriction we could instead take the absolute value of the components of (x(i) – mut + 1y) to be the diagonals of the new covariance matrix.

Problem 2:

Learned w: [ -6.08510226 -10.28999668 -16.29878248 13.94465505 13.11237706

16.32236034 14.70945943 16.32720122 20.49134538 19.71430646

20.10067608 18.41666625 19.75525719 20.10191463 9.43327593

-17.83077248 4.19251905 5.4202896 25.37544557 26.0305132

20.53543531 24.98397087]

Learned b: 38.59912342729282

Accuracy on the training dataset: 0.7051282051282052

Accuracy on the test dataset: 0.6949152542372882

When the data is seperable, the weights will want to become larger and larger to emphasize the difference between the two classes. From the loss function’s perspective, it always has an incentive to make the weights larger and larger. If you double the size of the weights, one of the classes will gain larger log probabilities, and the other class will therefore lose log probability.