Extra Credit: EM

Jeong Hyun Lee
STAT 5114 – Statistical Inference

April 12, 2025

As a short precursor, the Gaussian mixture model (GMM) is a density estimator, that is, it aims to approximate especially smooth densities through multiple component Gaussian distributions.

A p-dimensional Gaussian mixture model with K components can be represents a distribution as

$$\boldsymbol{x}_i \sim \sum_{k=1}^{K} \pi_k p(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \Sigma_k, C_k), \quad i = 1, \dots, N$$

where $p(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \Sigma_k)$ is the p-dimensional multivariate normal density under $\boldsymbol{\mu}_k$ and Σ_k, and the π_k are mixing coefficients such that $\sum_{k=1}^{k} \pi_k = 1$ and $\pi_k \geq 0$ for all k.

In order to simplify the optimization process, the GMM incorporates a latent variable structure. Let C be the underlying latent variable, a categorical random variable which would represent which Gaussian generated our observation $\boldsymbol{x}_i$, with some probability. Hence,

$$C \sim \text{Categorical}(\pi)$$

and we denote C_k as when $C = k$.

As there are K component Gaussians, $(\boldsymbol{\mu}_k, \Sigma_k)$ are the parameters for the kth component Gaussian, and π_k is the probability $\boldsymbol{x}_i$ was generated from the kth Gaussian: $\pi_k = P(C^{(i)} = k)$, where the superscript indicates for the ith data point.

Thus, we get our sampling distribution for our data stated above (using Bayes' rule).

$$p(\boldsymbol{x}_i) = \sum_{k=1}^{K} p(\boldsymbol{x}_i, C^{(i)} = k) = \sum_{k=1}^{K} p(C^{(i)} = k) p(\boldsymbol{x}_i \mid C^{(i)} = k) = \sum_{k=1}^{K} \pi_k p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k, C_k)$$

Given the sampling distribution, it is only natural for us to derive the joint sampling distribution (so that we can get to our likelihood function). Hence,

$$p(\boldsymbol{x}_1, \dots, \boldsymbol{x}_N; \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) = \prod_{i=1}^{N} p(\boldsymbol{x}_i; \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)$$

$$= \prod_{i=1}^{N} \sum_{k=1}^{K} p(C^{(i)} = k; \boldsymbol{\pi}) p(\boldsymbol{x}_i \,|\, C^{(i)} = k; \boldsymbol{\mu}, \Sigma)$$

$$= \prod_{i=1}^{N} \sum_{k=1}^{K} \pi_k p(\boldsymbol{x}_i \,|\, \boldsymbol{\mu}_k, \Sigma_k, C_k)$$

Our likelihood function, which is functionally equivalent to the above, under $\mathbf{x} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, becomes the following.

$$\mathcal{L}(\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma \,|\, \mathbf{x}) = \prod_{i=1}^{N} \sum_{k=1}^{K} \pi_k p(\boldsymbol{x}_i \,|\, \boldsymbol{\mu}_k, \Sigma_k, C_k)$$

$$\ell(\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma \,|\, \mathbf{x}) = \sum_{i=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k p(\boldsymbol{x}_i \,|\, \boldsymbol{\mu}_k, \Sigma_k, C_k) \right)$$

If we knew the $C^{(i)}$ for each $\boldsymbol{x}_i$, evaluating and maximizing the likelihood would be a much easier task. However, as it is a latent variable hence unknown, we must incorporate an algorithm, called the Expectation-Maximization (EM) algorithm. The algorithm alternates between the following two steps.

1. E-step: Compute the posterior probability that each Gaussian generates each data point (as this is unknown to us)
2. M-step: Assuming that the data really was generated this way, change the parameters of each Gaussian to maximize the probability that it would generate the data it is currently responsible for.

Thus, during the E-step, since we cannot be certain of which Gaussian generated which data point, we model it as a distribution.

$$\pi_{i,k} = P(\boldsymbol{x}_i \in C_k \,|\, \boldsymbol{\mu}_k, \Sigma_k)$$

$\pi_{i,k}$ will be the *responsibility* of the kth Gaussian for the ith data point.

Then during the M-step, with the found responsibilities which we will now assume as posterior probabilities for each data point, we use maximum likelihood, satisfying the following first order condition on the set of parameters $\boldsymbol{\Theta} = \{\boldsymbol{\mu}, \Sigma\}$,

$$\frac{\partial \ln p(\mathbf{x}; \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)}{\partial \boldsymbol{\Theta}} = 0$$

to derive a closed form updates for all parameters.

Now we begin the problem, where it asks us to derive the MLE for $\boldsymbol{\mu}_k$ and Σ_k, the crucial part of the M-step.

Part (a)

We first simplify the given expression for $\pi_{i,k}$ using Bayes' rule.

$$\pi_{i,k} = p(\boldsymbol{x}_i \in C_k \,|\, \boldsymbol{\mu}_k, \Sigma_k)$$

$$\begin{aligned}
&= p(C = k \mid \boldsymbol{x}_i) \\
&= \frac{p(C = k)p(\boldsymbol{x}_i | C = k)}{p(\boldsymbol{x}_i)} \\
&= \frac{p(C = k)p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^K p(C = j)p(\boldsymbol{x}_i | C = j)} \\
&= \frac{\pi_k p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k, C_k)}{\sum_{j=1}^K \pi_j p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \Sigma_j, C_j)}
\end{aligned}$$

To find the MLE for $\boldsymbol{\mu}_k$s, we take the derivative of the log-likelihood with respect to $\boldsymbol{\mu}_k$ and set it to 0.

$$\begin{aligned}
\frac{\partial \ell}{\partial \boldsymbol{\mu}_k} = 0 &= \frac{\partial}{\partial \boldsymbol{\mu}_k} \sum_{i=1}^N \ln \left(\sum_{k=1}^K \pi_k p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k, C_k) \right) \\
&= \sum_{i=1}^N \frac{\partial}{\partial \boldsymbol{\mu}_k} \ln \left(\sum_{k=1}^K \pi_k p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k, C_k) \right) \\
&= \sum_{i=1}^N \frac{1}{\sum_{j=1}^K \pi_j p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \Sigma_j, C_j)} \frac{\partial}{\partial \boldsymbol{\mu}_k} \sum_{k=1}^K \pi_k p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k, C_k) \\
&= \sum_{i=1}^N \frac{1}{\sum_{j=1}^K \pi_j p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \Sigma_j, C_j)} \frac{\partial}{\partial \boldsymbol{\mu}_k} \pi_k p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k, C_k) \\
&= \sum_{i=1}^N \frac{\pi_k p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k, C_k)}{\sum_{j=1}^K \pi_j p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \Sigma_j, C_j)} \frac{\partial}{\partial \boldsymbol{\mu}_k} \left(-\frac{1}{2} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_k) \right) \\
&= \sum_{i=1}^N \frac{\pi_k p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k, C_k)}{\sum_{j=1}^K \pi_j p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \Sigma_j, C_j)} \cdot \frac{-1}{2} (\Sigma_k^{-1} + \Sigma_k^{-1,\top}) (\boldsymbol{x}_i - \boldsymbol{\mu}_k) \cdot -1 \\
&= \sum_{i=1}^N \frac{\pi_k p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k, C_k)}{\sum_{j=1}^K \pi_j p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \Sigma_j, C_j)} \Sigma_k^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)
\end{aligned}$$

where we note that as the normal density is an exponential, its partial derivative with respect to the mean, which is only within the exponential, due to the chain rule, will result in the same exponential, hence preserving the normal density form. Secondly, the matrix differentiation for the quadratic form is $\frac{\partial \boldsymbol{x}^\top A \boldsymbol{x}}{\partial \boldsymbol{x}} = (A + A^\top)\boldsymbol{x}$.

Hence, we have

$$\begin{aligned}
0 &= \sum_{i=1}^N \frac{\pi_k p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k, C_k)}{\sum_{j=1}^K \pi_j p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \Sigma_j, C_j)} \Sigma_k^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_k) \\
&= \sum_{i=1}^N \pi_{i,k} \Sigma_k^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_k) \\
&= \sum_{i=1}^N \pi_{i,k} \Sigma_k^{-1} \boldsymbol{x}_i - \sum_{i=1}^N \pi_{i,k} \Sigma_k^{-1} \boldsymbol{\mu}_k
\end{aligned}$$

$$\begin{aligned}
\sum_{i=1}^{N} \pi_{i,k} \Sigma_k^{-1} \boldsymbol{\mu}_k &= \sum_{i=1}^{N} \pi_{i,k} \Sigma_k^{-1} \boldsymbol{x}_i \\
\Sigma_k^{-1} \boldsymbol{\mu}_k \sum_{i=1}^{N} \pi_{i,k} &= \Sigma_k^{-1} \sum_{i=1}^{N} \pi_{i,k} \boldsymbol{x}_i \\
\boldsymbol{\mu}_k \sum_{i=1}^{N} \pi_{i,k} &= \sum_{i=1}^{N} \pi_{i,k} \boldsymbol{x}_i \\
&\boxed{\hat{\boldsymbol{\mu}}_k = \frac{\sum_{i=1}^{N} \pi_{i,k} \boldsymbol{x}_i}{\sum_{i=1}^{N} \pi_{i,k}}}
\end{aligned}$$

Part (b)

Now, given $\pi_{i,k}$ and $\boldsymbol{\mu}_k$, we solve for Σ_k.

$$\begin{aligned}
\frac{\partial \ell}{\partial \Sigma_k} = 0 &= \frac{\partial}{\partial \Sigma_k} \sum_{i=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k, C_k) \right) \\
&= \sum_{i=1}^{N} \frac{\partial}{\partial \Sigma_k} \ln \left(\sum_{k=1}^{K} \pi_k p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k, C_k) \right) \\
&= \sum_{i=1}^{N} \frac{1}{\sum_{j=1}^{K} \pi_j p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \Sigma_j, C_j)} \frac{\partial}{\partial \Sigma_k} \sum_{k=1}^{K} \pi_k p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k, C_k) \\
&= \sum_{i=1}^{N} \frac{\pi_k}{\sum_{j=1}^{K} \pi_j p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \Sigma_j, C_j)} \frac{\partial}{\partial \Sigma_k} p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k, C_k)
\end{aligned}$$

For this last partial, we do the log trick, where

$$\frac{\partial \ln(f_k)}{\partial \Sigma_k} = \frac{1}{f_k} \frac{\partial f_k}{\partial \Sigma_k} \implies \frac{\partial f_k}{\partial \Sigma_k} = f_k \frac{\partial \ln(f_k)}{\partial \Sigma_k}$$

$$\begin{aligned}
0 &= \sum_{i=1}^{N} \frac{\pi_k}{\sum_{j=1}^{K} \pi_j p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \Sigma_j, C_j)} p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k, C_k) \frac{\partial}{\partial \Sigma_k} \ln(p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k, C_k)) \\
&= \sum_{i=1}^{N} \frac{\pi_k p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k, C_k)}{\sum_{j=1}^{K} \pi_j p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \Sigma_j, C_j)} \frac{\partial}{\partial \Sigma_k} \ln \left(\frac{1}{\pi^{p/2} |\Sigma_k|^{1/2}} e^{-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)} \right) \\
&= \sum_{i=1}^{N} \frac{\pi_k p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k, C_k)}{\sum_{j=1}^{K} \pi_j p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \Sigma_j, C_j)} \frac{\partial}{\partial \Sigma_k} \left[-\frac{p}{2} \ln(\pi) - \frac{1}{2} \ln |\Sigma_k| - \frac{1}{2} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_k) \right] \\
&= \sum_{i=1}^{N} \frac{\pi_k p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k, C_k)}{\sum_{j=1}^{K} \pi_j p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \Sigma_j, C_j)} \frac{\partial}{\partial \Sigma_k} \left[-\frac{p}{2} \ln(\pi) + \frac{1}{2} \ln |\Sigma_k|^{-1} - \frac{1}{2} \mathrm{tr}((\boldsymbol{x}_i - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)) \right] \\
&= \sum_{i=1}^{N} \frac{\pi_k p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k, C_k)}{\sum_{j=1}^{K} \pi_j p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \Sigma_j, C_j)} \frac{\partial}{\partial \Sigma_k} \left[-\frac{p}{2} \ln(\pi) + \frac{1}{2} \ln |\Sigma_k^{-1}| - \frac{1}{2} \mathrm{tr}((\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1}) \right]
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=1}^{N} \frac{\pi_k p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k, C_k)}{\sum_{j=1}^{K} \pi_j p(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \Sigma_j, C_j)} \left[\frac{1}{2}\Sigma_k - \frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^\top\right] \\
&= \sum_{i=1}^{N} \pi_{i,k} \left[\Sigma_k - (\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^\top\right] \\
&= \sum_{i=1}^{N} \pi_{i,k}\Sigma_k - \sum_{i=1}^{N} \pi_{i,k}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^\top
\end{aligned}
$$

where we used the facts that $|A|^{-1} = |A^{-1}|$, $\frac{\partial \ln |A|}{\partial A} = A^{-\intercal}$, and $\frac{\partial \text{tr}(BA)}{\partial A} = B^\top$.

$$
\begin{aligned}
\implies \sum_{i=1}^{N} \pi_{i,k}\Sigma_k &= \sum_{i=1}^{N} \pi_{i,k}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^\top \\
\Sigma_k \sum_{i=1}^{N} \pi_{i,k} &= \sum_{i=1}^{N} \pi_{i,k}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^\top \\
&\boxed{\hat{\Sigma}_k = \frac{\sum_{i=1}^{N} \pi_{i,k}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^\top}{\sum_{i=1}^{N} \pi_{i,k}}}
\end{aligned}
$$

Part (c)

The EM algorithm is proven to improve the maximum value of the likelihood. Through Jensen's inequality, one could easily show that through repeated E- and M-steps, the lower bound on the likelihood function will continue to improve (get higher). In fact, it is known for the algorithm to converge, though it may not be the global maximum. We refer to Figure 1 for a succinct illustration on how the EM algorithm converges.

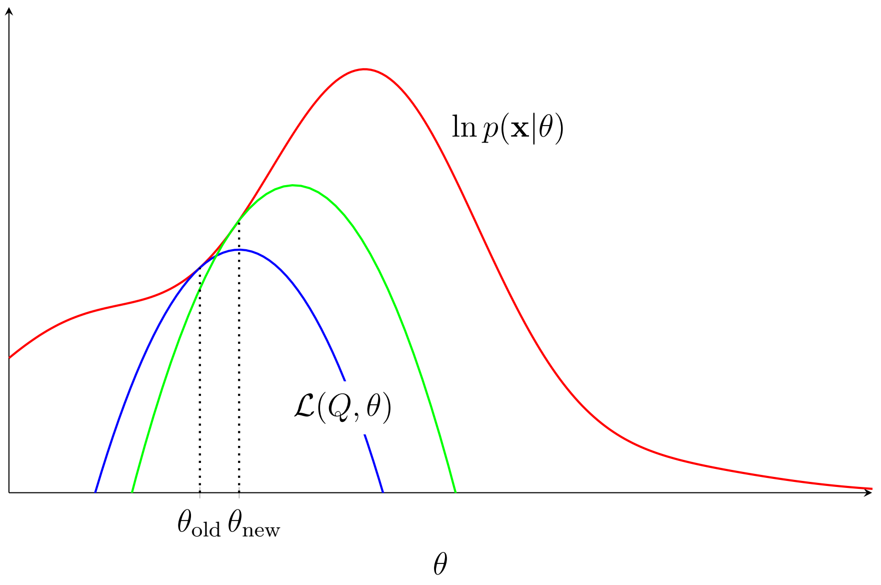


Figure 1: Illustration on how Q converges

In implementing our EM algorithm, the choice of initialization matters. As we will see in Example 2, a 'bad' set of initialization means and covariances can lead to an incorrect identification of the component Gaussian and their parameters. In our implementation, we provide two options: k-means or random from data initialization, by specifying the `init_kmeans` parameter of the `EM` function. K-means initialization uses the standard k-means clustering algorithm to group the data into k clusters by l_2 distance, and we take the mean and covariance of each cluster as our initial guess to begin the algorithm. As you see throughout the first three examples, under fairly isolated clusters, this leads to an incredibly efficient runtime, and accurate result of the EM algorithm as the clusters identified by the k-means is most likely the original clusters. In fact, one can interpret the k-means algorithm as also fitting a GMM with spherical errors. Through the k-means initialization, we also minimize wrongly fitted GMMs.

The second method of initialization, random from data, literally randomly selects k data points to be the initial mean vectors. We then initialize the covariances by taking the covariance of the entire dataset given each mean vector as the truth. This inadvertently creates initial Gaussians with large covariances, as shown in the 2D examples (Examples 1, 2 and 4), however will minimize the 'mistake' of incorrectly identifying a cluster. Yet, as Example 2 will show, even then, such initialization scheme is prone to misidentified models if the initially selected mean points are 'bad.'

Further as the purpose of the EM algorithm is to find the set of parameters that maximizes the likelihood, we introduce a stopping criterion, where once the log-likelihood improvement is less than the threshold, we stop the iterative process.

To illustrate the process of the EM algorithm and its weakness, we provide four examples. For the first three examples, we randomly generate a Gaussian mixture model, while for the fourth example, we specify parameters to create a less isolated dataset. (Interactive visualizations available at `jeonghlee12.github.io/STAT5114/EM/EM_test.html`)

Example 1

In our first example, we employ $k = 3$ component Gaussians under $p = 2$ dimensions. We set the number of iterations as 20. For a total of 5000 samples, the generated sample is displayed in Figure 2, where we draw the contour lines 2σ from the mean.

Then in Figure 3 and Figure 4, we show for both initialization method, the initialized parameters, and then the updated parameters after some iterations, along with the final iteration. Again, each red ellipse represents the 2σ contour of each component Gaussian at the current parameters.

For the K-means initialization, we find that indeed, the algorithm finishes in 2 iterations under the stopping criterion. From initialization, we find the parameters are already close to the maximum likelihood, and hence only a few more iterations are needed before convergence.

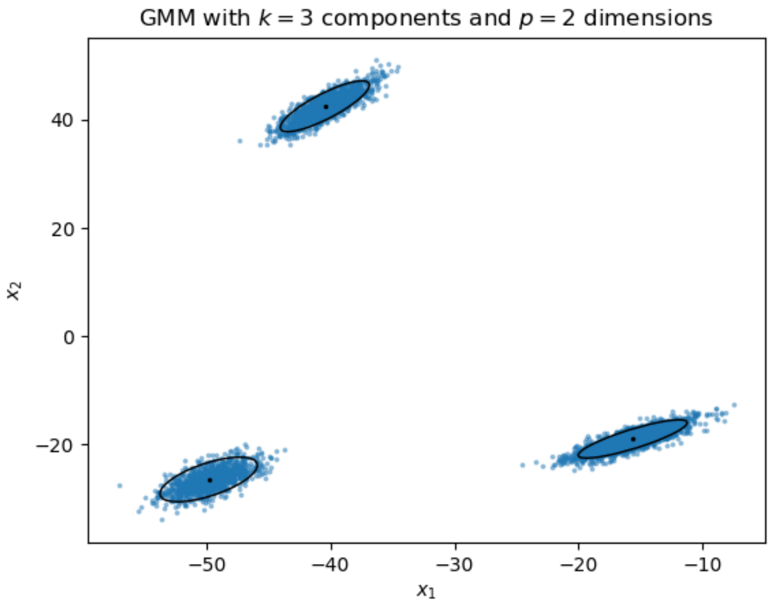
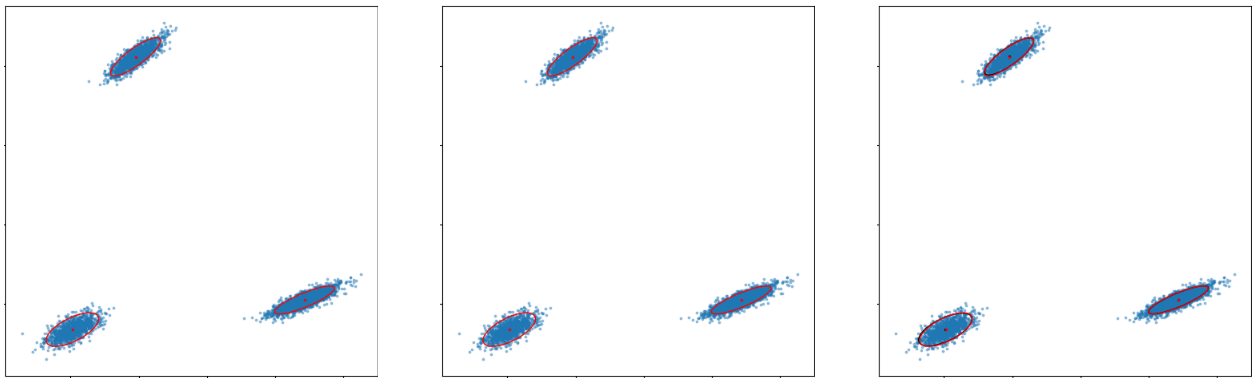


Figure 2: Example 1 data, overlayed with the mean and 2σ contour lines



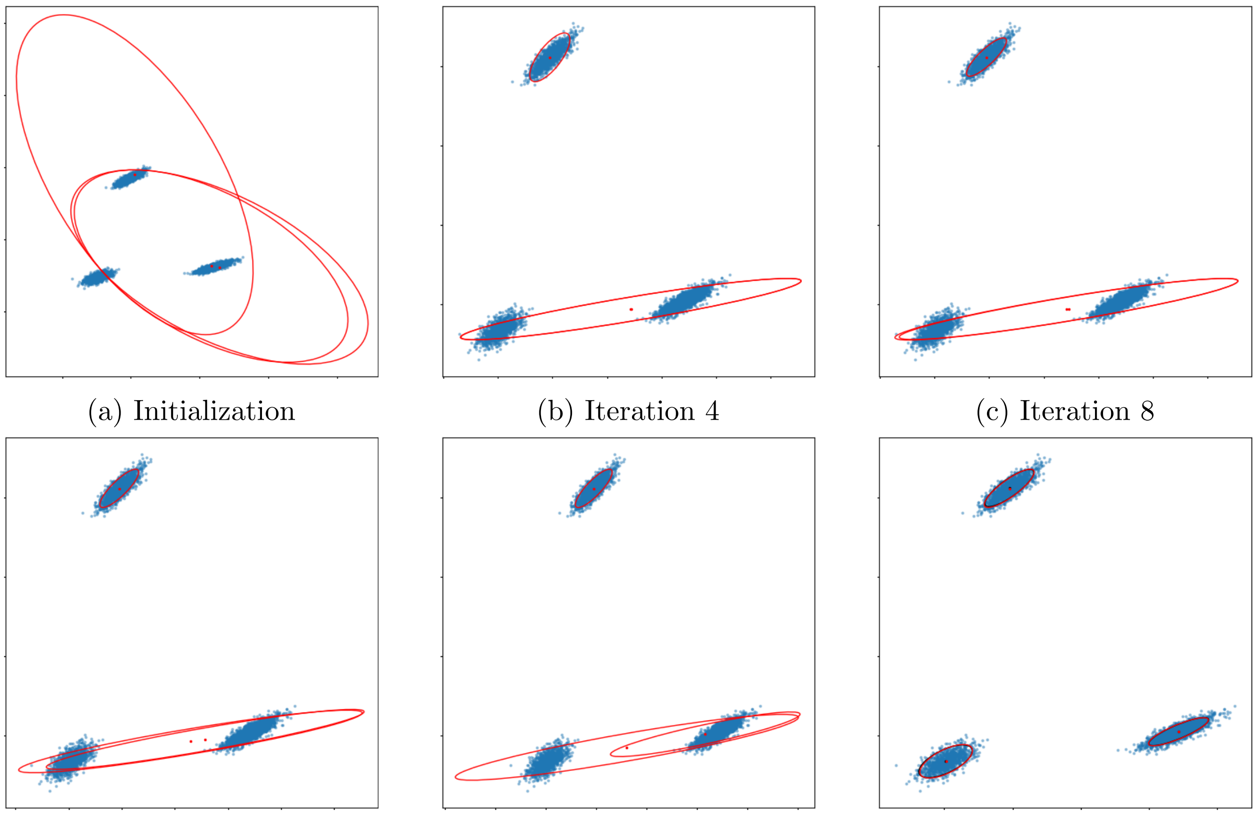
(a) Initialization (b) Iteration 1 (c) Iteration 2

Figure 3: Example 1 — K-means initialization

For the random from data initialization in Figure 4, we find that the algorithm does construct initial Gaussians with large covariances, and for the two component clusters at the bottom, only at the very end of the 20 itnerations does the algorithm seem to converge near the true values.

Figure 5 shows the progression of improvement of the log-likelihood over each iteration in each of the initialization method. Both likelihoods converge to -23628.576.

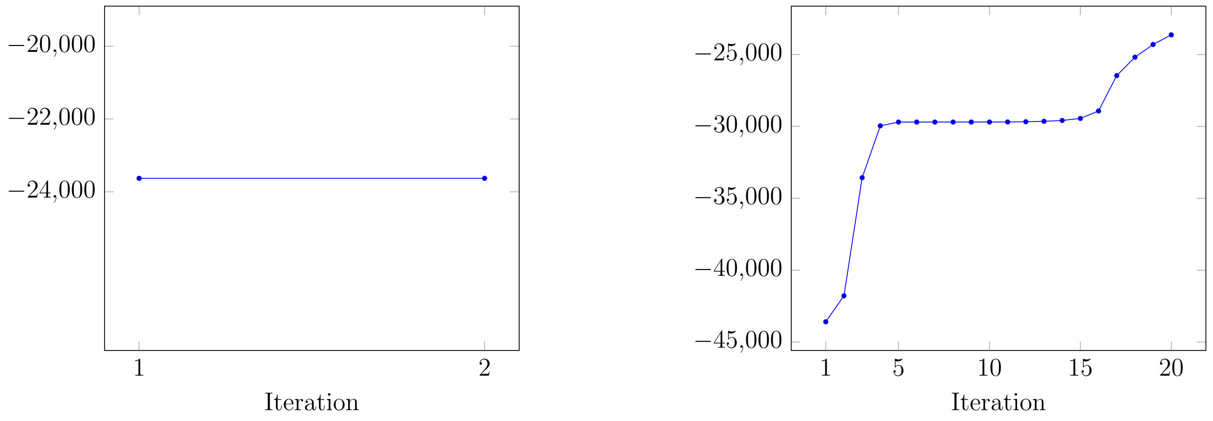
As this is a simulation study, we have access to the true parameters. Hence, we compare the MSE of the means and various scalar measures for the covariance to evaluate the accuracy of our method. Since the EM algorithm updates the parameters, there is no order to which set of parameters correspond to a component Gaussian. Thus we structure our loss metric



(a) Initialization (b) Iteration 4 (c) Iteration 8

(d) Iteration 12 (e) Iteration 16 (f) Iteration 20

Figure 4: Example 1 — Random from data initialization



(a) K-means initialization (b) Random from data initialization

Figure 5: Example 1 log-likelihoods

in generality as follows.

As all our examples concern a low number of component Gaussians (maximum of 4), we compute the MSE for the means in each permutation order of the sets of parameters (for $K = 3$, we would test (C_1, C_2, C_3), (C_1, C_3, C_2), so forth) and find the order with the lowest MSE.

This will be our order for evaluating the covariance metrics. For evaluating the covariances, we provide three metrics: the average $||\Sigma| - |\hat{\Sigma}||$, the average $|\text{tr}(\Sigma) - \text{tr}(\Sigma)|$, and the average $||\Sigma - \hat{\Sigma}||_F$ across the components (F for the Frobenius norm). We then finally report the KL divergence $D_{KL}(C \,|\, \hat{C})$, to evaluate the information loss when we use our estimate $\hat{C}$ for the true C. We do this also for the mixing distribution $\boldsymbol{\pi}$.

We now present the loss metrics in Table 1. We find for both initialization methods, the losses

	K-means	Random from data
MSE($\hat{\mu}$)	0.0072187	0.0072196
avg. $\Big\|\|\Sigma\| - \|\hat{\Sigma}\|\Big\|$	0.12845	0.12893
avg. $\|\text{tr}(\Sigma) - \text{tr}(\Sigma)\|$	0.19148	0.19162
avg. $\|\|\Sigma - \hat{\Sigma}\|\|_F$	0.19148	0.19162
avg. $D_{KL}(C \,\|\, \hat{C})$	0.0022516	0.0022504
avg. $D_{KL}(\boldsymbol{\pi} \,\|\, \hat{\boldsymbol{\pi}})$	0.00013779	0.00013779
Log-likelihood	-23628.576	-23628.576

Table 1: Accuracy metrics for Example 1

are overall low, indicating our found estimates are close to the truth. We see in Example 2 how these values drastically change when the algorithm misidentifies the model.

Example 2

In our second example, we also have $k = 3$ component Gaussians under $p = 2$ dimensions and 20 iterations. We have again a total of 5000 samples displayed in Figure 6.

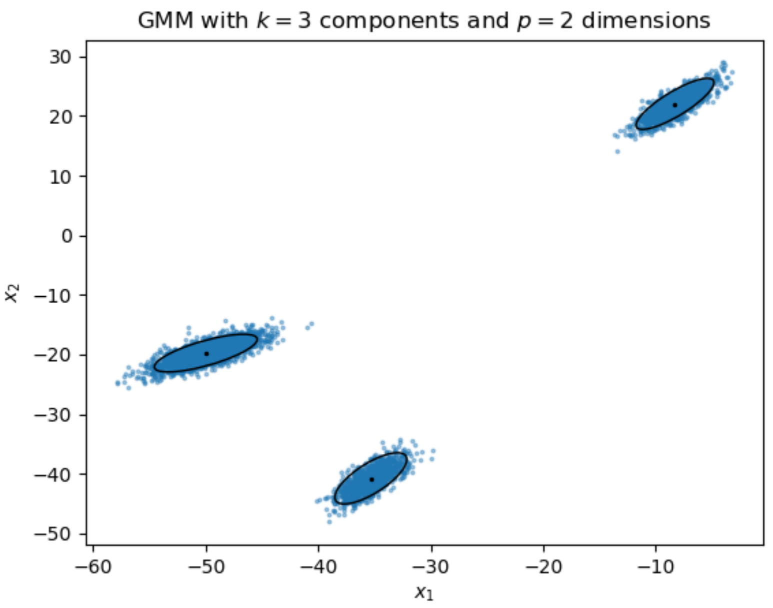


Figure 6: Example 2 data, overlayed with the mean and 2σ contour lines

Then in Figure 7 and Figure 8, we show for both initialization method, the initialized parameters, and then the updated parameters after some iterations, along with the final iteration.

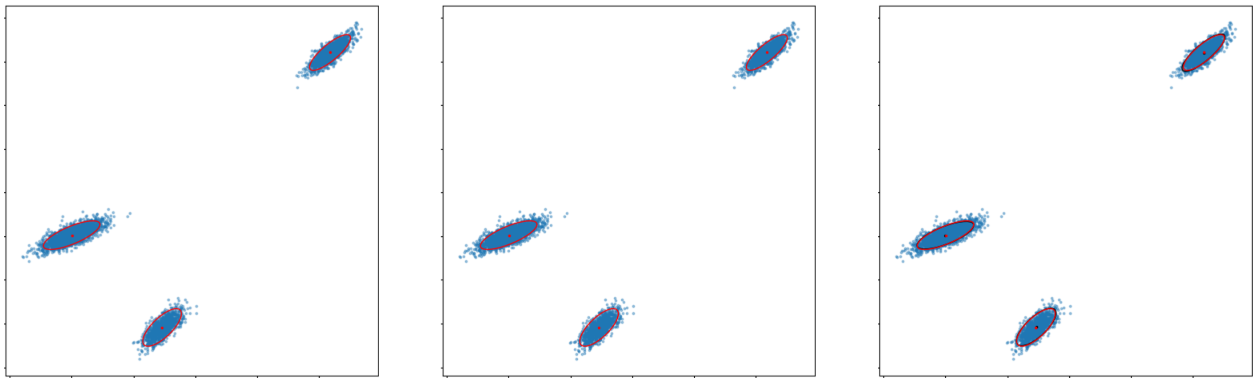
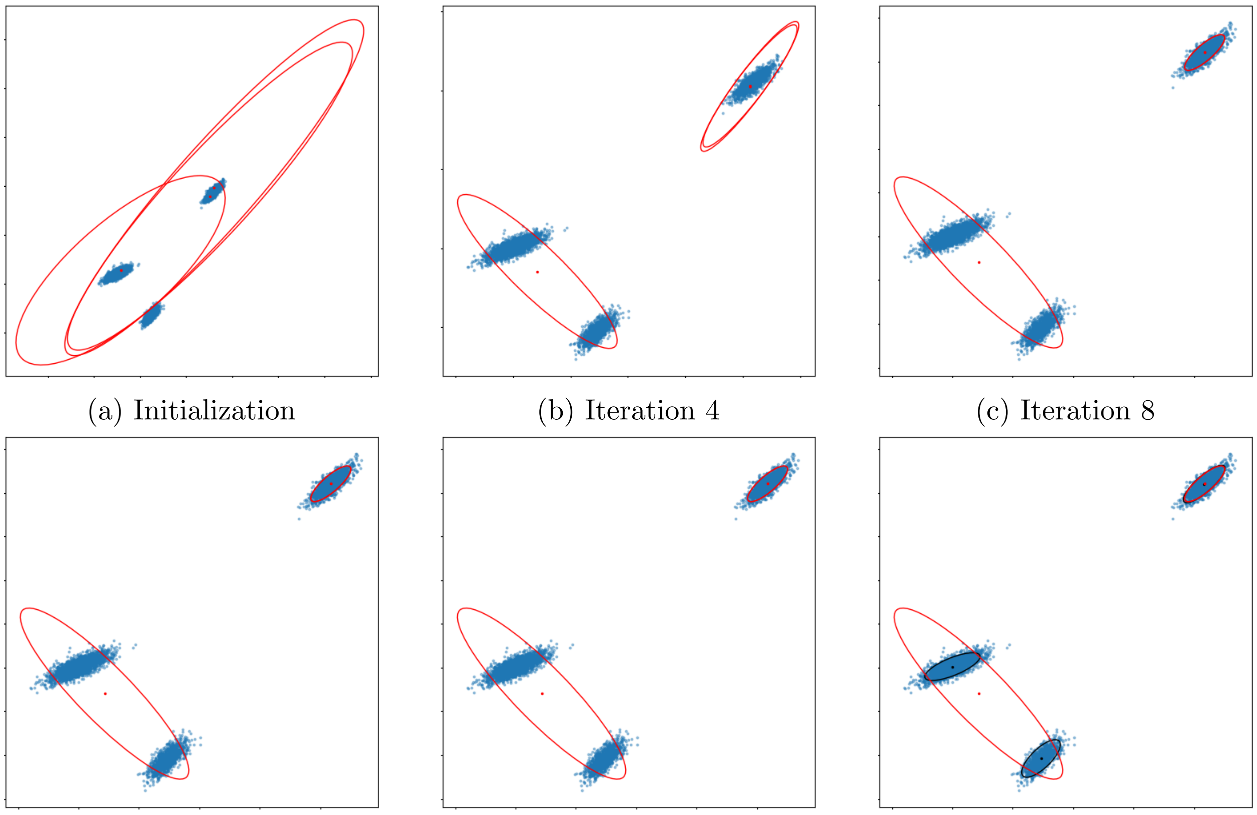

(a) Initialization (b) Iteration 1 (c) Iteration 2

Figure 7: Example 2 — K-means initialization


(a) Initialization (b) Iteration 4 (c) Iteration 8

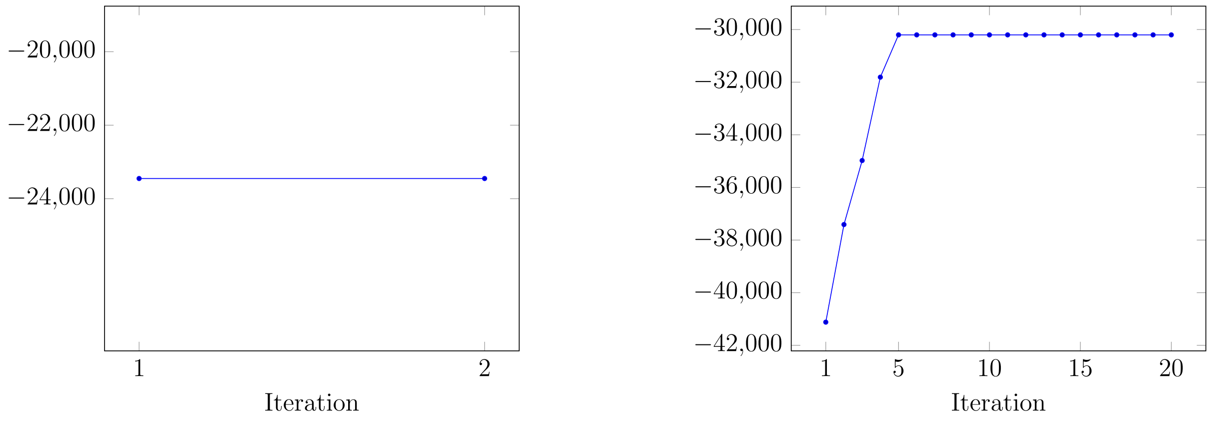
(d) Iteration 12 (e) Iteration 16 (f) Iteration 20

Figure 8: Example 2 — Random from data initialization

For the K-means initialization, we find that indeed, the algorithm finishes in 2 iterations under the stopping criterion. The red lines indicate the Gaussian components illustrated at the current parameter values and the 2σ contour, which we see from initialization, is already reflective of each component.

For the random from data initialization in Figure 8, we find that due to the ‘bad’ initialization, by iteration 20, the algorithm has completely misidentified the Gaussian components. The two components on the bottom left are grouped into one, and the algorithm updates two components for the single cluster on the top right. Hence, this example showcases how bad initialization can lead to misidentified models and further suggests the use of a more structured initialization scheme such as the k-means.

Figure 9 shows the progression of improvement of the log-likelihood over each iterations in each of the initialization method. The contrast in the identified and misidentified algorithm results is show here. Clearly, the final log-likelihood from the k-means initialization, -23450.980, is much higher than that from the random from data initialization, -30203.271.



(a) K-means initialization

(b) Random from data initialization

Figure 9: Example 2 log-likelihoods

We now present the loss metrics in Table 2. We find for the k-means initialization method,

	K-means	Random from data
MSE($\hat{\mu}$)	0.0072187	1273.615
avg. $\left\| \|\Sigma\| - \|\hat{\Sigma}\| \right\|$	0.006552	300.7040
avg. $\|\mathrm{tr}(\Sigma) - \mathrm{tr}(\Sigma)\|$	0.32335	46.24345
avg. $\|\|\Sigma - \hat{\Sigma}\|\|_F$	0.32335	46.24345
avg. $D_{KL}(C \mid \hat{C})$	0.0028346	107.166
avg. $D_{KL}(\boldsymbol{\pi} \mid \hat{\boldsymbol{\pi}})$	0.00018153	0.59109
Log-likelihood	-23450.980	-30203.271

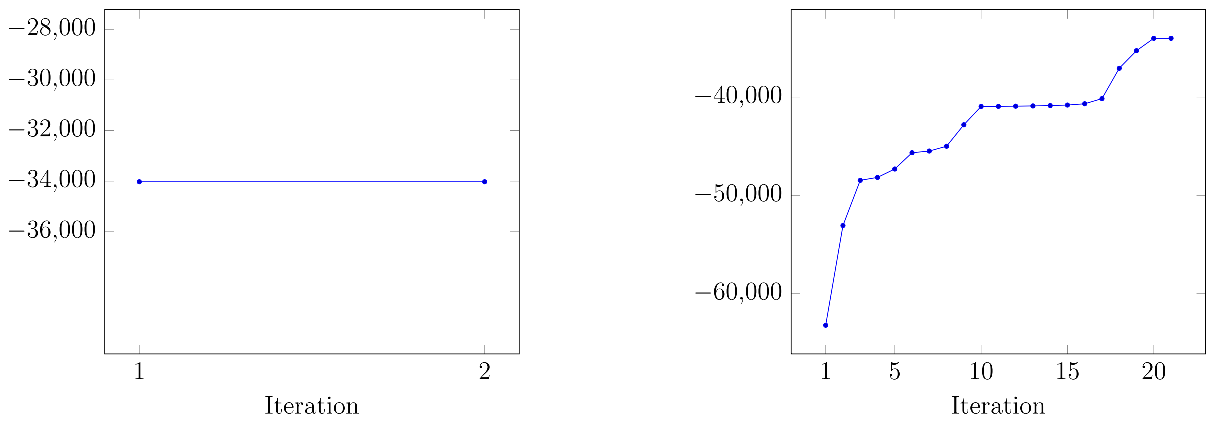
Table 2: Accuracy metrics for Example 2

which we verified visually in Figure 7 to be close to the true model, the losses are overall low, indicating our found estimates are indeed close to the truth. However, under the random from data initialization, all the metrics seem to be high, indicating the discrepancy between the estimated covariances and the truth. This does indicate that our model is most likely misidentified for the random from data initialization.

This further highlights under random from data initialization, multiple runs must be made.

Example 3

In our third example, we have $k = 4$ component Gaussians under $p = 3$ dimensions. Due to a larger dimension and component size, we set the number of iterations as 50, though in both initialization scheme, we find the algorithm to stop early. Again for a total of 5000 samples, since we cannot visualize 3-D Gaussians, we proceed right to the analysis. Figure 10 shows the progression of improvement of the log-likelihood over each iterations in each of the initialization method. As we see both log-likelihoods converging to the same value of -34025.391, we can infer that our algorithm does fit the correct GMM, again where we note both methods converge early on.



(a) K-means initialization (b) Random from data initialization

Figure 10: Example 3 log-likelihoods

We now present the loss metrics in Table 3. We find for both initialization methods, the

	K-means	Random from data
MSE($\hat{\mu}$)	0.0068474	0.0068473
avg. $\left\lvert \lvert\Sigma\rvert - \lvert\hat{\Sigma}\rvert \right\rvert$	0.58063	0.58063
avg. $\lvert \mathrm{tr}(\Sigma) - \mathrm{tr}(\Sigma) \rvert$	0.15421	0.15421
avg. $\lvert\lvert \Sigma - \hat{\Sigma} \rvert\rvert_F$	0.15421	0.15421
avg. $D_{KL}(C \mid \hat{C})$	0.0027228	0.0027228
avg. $D_{KL}(\boldsymbol{\pi} \mid \hat{\boldsymbol{\pi}})$	0.00019839	0.00019839
Log-likelihood	-34025.391	-34025.391

Table 3: Accuracy metrics for Example 3

losses are overall low, and further, identical given the rounding. This indicates both methods have converged to the same model, further supporting that our algorithm has most likely converged to the true model, and the low loss values indicate that our estimates are indeed close to the true parameters.

Example 4

In our previous examples, we've had component Gaussians that were all isolated. In this example, we specifically specify the true parameters (as opposed to randomly generating them previously) to have our data be 'connected.' Given the more 'difficult' nature of identifying the GMM here, we specify 50 iterations. We generate a GMM with $k = 3$ components and $p = 2$ dimensions. Under 5000 samples again, Figure 11 shows the visualization of the data.

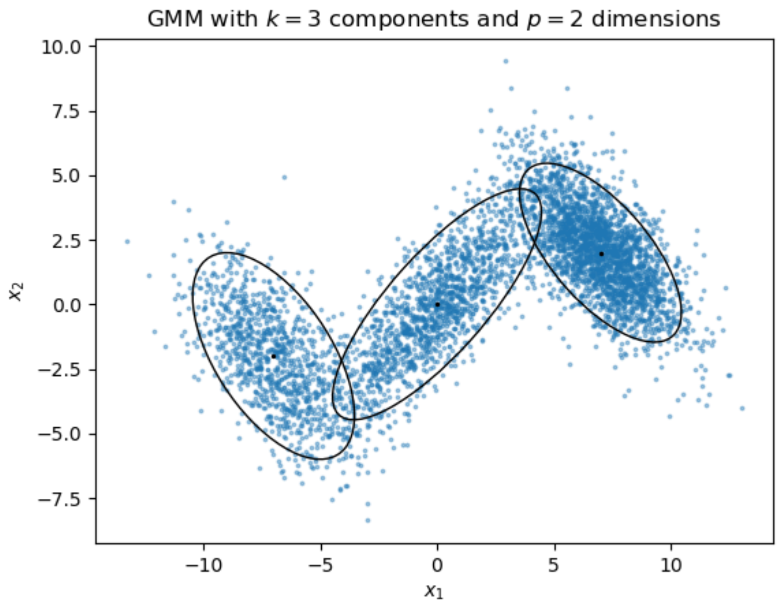
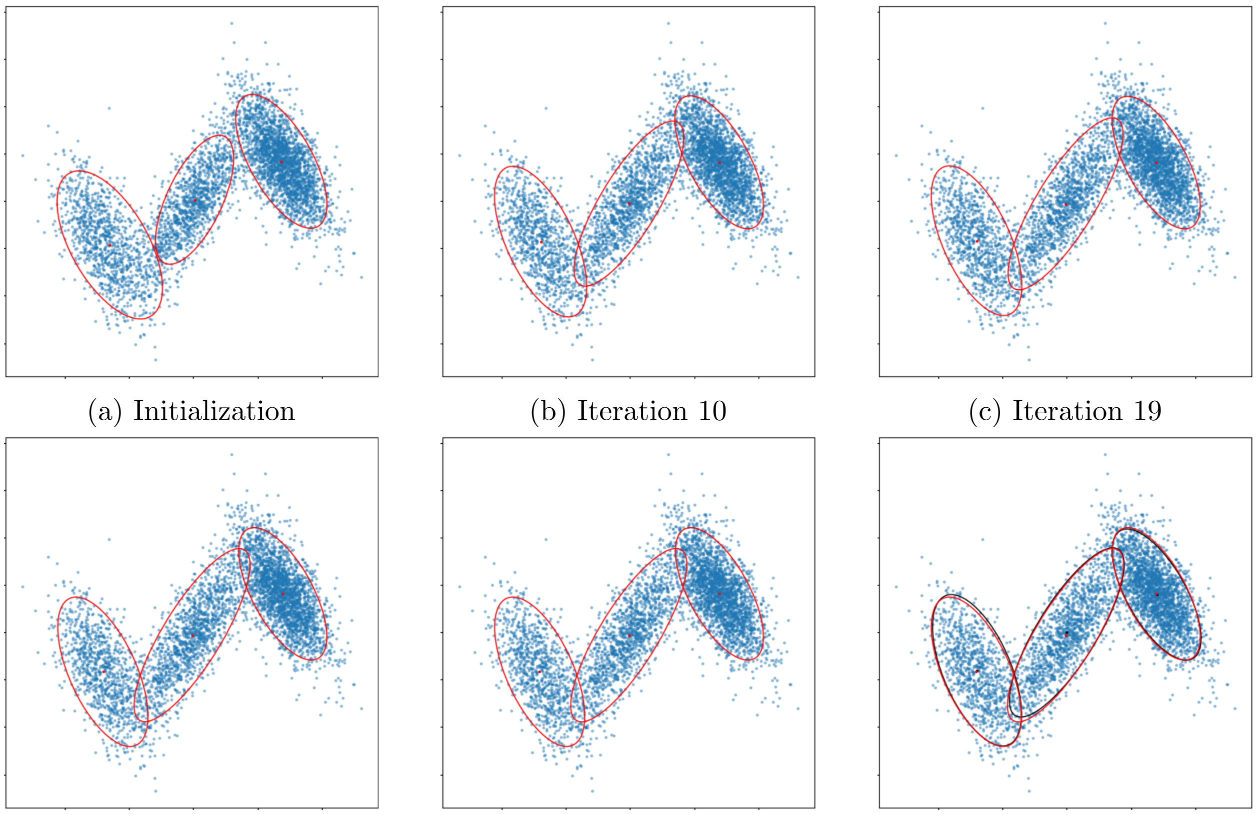


Figure 11: Example 4 data, overlayed with the mean and 2σ contour lines

Figure 12 and Figure 13, we show for both initialization method, the initialized parameters, and then the updated parameters after some iterations, along with the final iteration. Again, each red ellipse represents the 2σ contour of each component Gaussian at the current parameters.

As we expected, even for the k-means initialization, the number of iterations to convergence still takes 46 iterations, nonetheless still stopping early. We inspect Figure 12 however, and we find that starting from the initialization, the parameters at each iteration create Gaussians that resemble the true Gaussian. We believe thus, the longer runtime is due to the boundary points, where each iteration, these points 'switch' their label, and hence create the discrepancy in the log-likelihood, thus the changes are above the threshold.



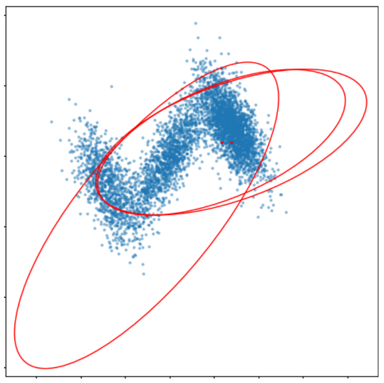
(a) Initialization (b) Iteration 10 (c) Iteration 19

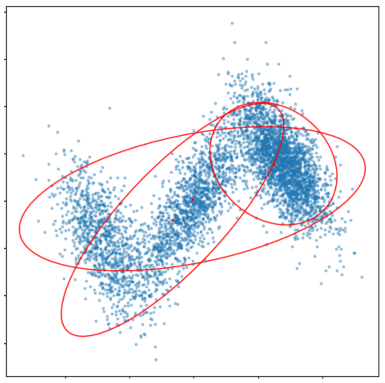
(d) Iteration 28 (e) Iteration 37 (f) Iteration 46

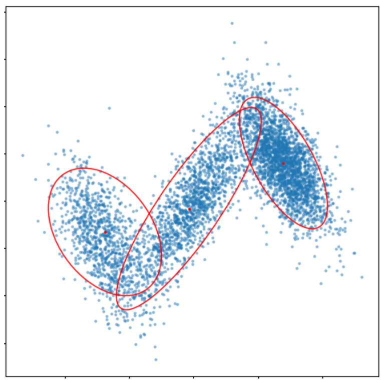
Figure 12: Example 4 — K-means initialization

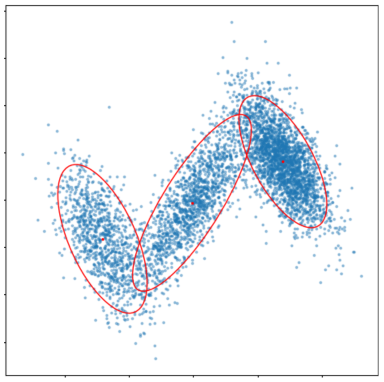
For the random from data initialization in Figure 13, we find a similar trend as with the k-means initialization. Albeit the initialization here is again, quite large, by iteration 30, we see that parameters have somewhat, 'settled' in near their true values. Again past iteration 30, it must be the small changes in these parameters that accumulate to the prolonged runtime.

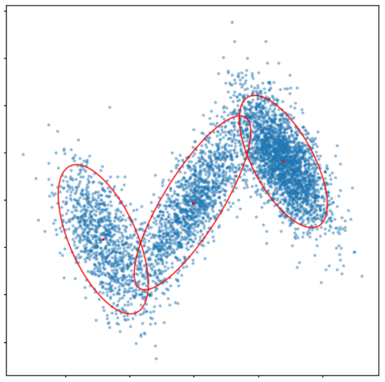
We can further investigate this in Figure 14 with the log-likelihoods. We see that for both initialization methods, the log-likelihoods begin to converge half-way through. In fact, both converge to a value of -34025.391.


(a) Initialization


(b) Iteration 10


(c) Iteration 20


(d) Iteration 30


(e) Iteration 40

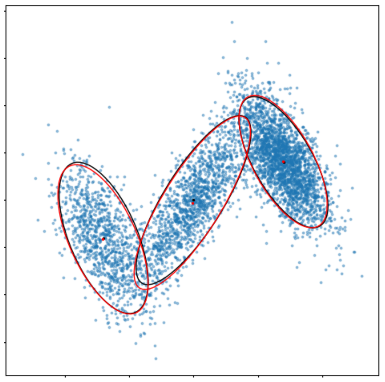
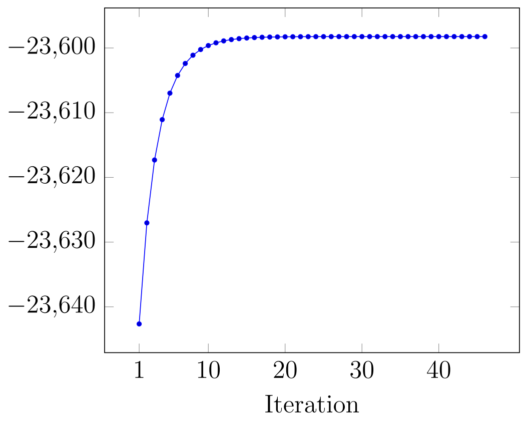
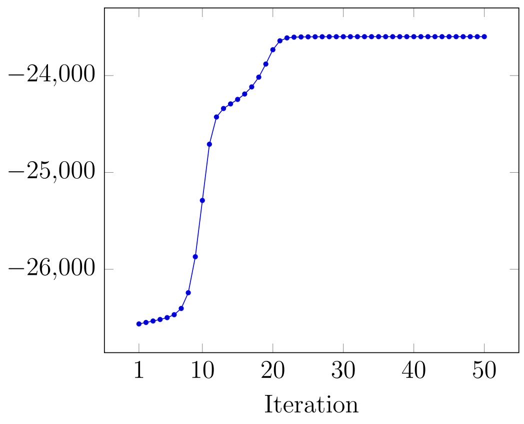

(f) Iteration 50

Figure 13: Example 4 — Random from data initialization



(a) K-means initialization



(b) Random from data initialization

Figure 14: Example 4 log-likelihoods

We now present the loss metrics in Table 4 We find for both initialization methods, the losses are overall low, and further, identical given the rounding. This indicates both methods have converged to the same model, further supporting that our algorithm has most likely converged to the true model, and the low loss values indicate that our estimates are indeed

	K-means	Random from data
MSE($\hat{\mu}$)	0.013498	0.0068473
avg. $\left\lvert \lvert\Sigma\rvert - \lvert\hat{\Sigma}\rvert \right\rvert$	0.20576	0.20576
avg. $\lvert\mathrm{tr}(\Sigma) - \mathrm{tr}(\Sigma)\rvert$	0.13814	0.13814
avg. $\lvert\lvert\Sigma - \hat{\Sigma}\rvert\rvert_F$	0.13814	0.13814
avg. $D_{KL}(C \mid \hat{C})$	0.0027424	0.0027424
avg. $D_{KL}(\boldsymbol{\pi} \mid \hat{\boldsymbol{\pi}})$	≈ 0	≈ 0
Log-likelihood	-23598.239	-23598.238

Table 4: Accuracy metrics for Example 4

close to the true parameters.