Exam Review

Rebecca C. Steorts

STA 325

Announcements

Exam 2: Released Friday, November 15th, 5:00 PM. Due Monday November at 25th, 10:00 AM.

Exam 3: Released Monday, December 2. Due Friday December 6.

Questions for the Class

- Would you like the exam released earlier to have more time to work on it and to ask questions to me on Monday, Wednesday, and Friday (as well as the teaching assistants)?
- What would be the most helpful?
- What else can we provide to provide support?
- I will also post some optional lecture slides in case you wish to learn about some other machine learning methods in case you finish the materials early.

What is clustering?

Clustering is an unsupervised method that divides up data into groups (clusters), so that points in any one group are more similar to each other than to points outside the group.

When might we want to use clustering?

One practical application of clustering is recommender systems, where one clusters users with similar viewing patterns on Netflix/Hulu, etc.

What are other applications we have seen in class or you have encountered?

Machine Learning Algorithms for Clustering

- Mixture Models and the EM Algorithm
- k-means
- hierarchical clustering
- how to choose the number of clusters

Agenda

- In this review, I will focus on mixture models
- Review Exponential Mixture Models
- Review Derivation of Estimates
- How to Code These Models Up

Mixture models can be viewed as probabilistic clustering

- Mixture models put similar data points into "clusters".
- This is appealing as we can potentially compare different probabilistic clustering methods by how well they predict (under cross-validation).
- This contrasts other methods such as k-means and hierarchical clustering as they produce clusters (and not predictions), so it's difficult to test if they are correct/incorrect.

Resources

One resource that is a tutorial on mixture models is the following: https://arxiv.org/pdf/1901.06708

The derivations are adapted from the authors' materials.

Mixture Model

Assume data points $x_1, \ldots, x_n$.

Assume that we have a mixture of K exponential distributions.

Following the notation in the article, assume that

$$g_k(x, \lambda_k) = \lambda_k e^{-\lambda_k x}, x > 0$$

From equation (1) in the article, it follows that

$$f(x, \lambda_1, \ldots, \lambda_K) = \sum_{k=1}^{K} \pi_k \lambda_k e^{-\lambda_k x},$$

which is a K-component exponential distribution with rate parameter λ_k and mixing proportions (or weights) π_k.

EM Algorithm

Because the likelihood and log-likelihood is complex for mixture models, we often utilize an algorithm called the expectation-maximization (EM) algorithm to estimate any unknown parameter values.

Properties of the EM Algorithm

1. It's quite general (see the tutorial), so we can apply it to any type of mixture model.
2. It converges to a local minimum but not a global one.
3. It can get stuck depending on the starting value, so this is something to watch out for in practice.
4. We typically monitor convergence of the algorithm using the log-likelihood.

Derivation of the E-Step

From equation (38) in the tutorial, we can update γ_{ik} as follows:

$$\gamma_{ik} = \frac{\pi_k \lambda_k e^{-\lambda_k x_i}}{\sum_{j=1}^{K} \pi_j \lambda_j e^{-\lambda_j x_i}}$$

where $i = 1, \ldots, n$ and $k = 1, \ldots, K$.

What does this step do in words?

Derivation of the M-Step

Using section 3 of the tutorial, we can work with a function $Q()$ that takes advantage of the latent components/variables.

$$Q(\lambda_1, \ldots, \lambda_K) = \sum_{i=1}^{n} \sum_{k=1}^{K} \gamma_{ik} \log[\pi_k g_k(x_i, \lambda_k)] \tag{1}$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{K} \gamma_{ik} \log[\pi_k \lambda_k e^{-\lambda_k x_i}] \tag{2}$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{K} \gamma_{ik} [\log \pi_k + \log \lambda_k - \lambda_k x_i] \tag{3}$$

Note that the following constraint must be satisfied $\sum_{k=1}^{K} \pi_k = 1$.

Derivation of the M-Step

The Lagrangian[1] becomes

$$\mathcal{L}(\lambda, \pi, \alpha) = \sum_{i=1}^{n} \sum_{k=1}^{K} [\gamma_{ik} \log \pi_k + \gamma_{ik} \log \lambda_k - \gamma_{ik} \lambda_k x_i] - \alpha(\sum_{k=1}^{K} \pi_k - 1). \tag{4}$$

This implies that

$$\frac{\partial \mathcal{L}}{\partial \lambda_k} = \sum_{i=1}^{n} \left[\frac{\gamma_{ik}}{\lambda_k} - \gamma_{ik} x_i \right] = 0 \implies \tag{5}$$

$$\frac{\sum_{i=1}^{n} \gamma_{ik}}{\lambda_k} = \sum_{i=1}^{n} \gamma_{ik} x_i \implies \tag{6}$$

$$\lambda_k = \frac{\sum_{i=1}^{n} \gamma_{ik}}{\sum_{i=1}^{n} \gamma_{ik} x_i} \tag{7}$$

[1] This is a constrained optimization problem.

Derivation of the M-Step

There are two approaches we can use to find π_k.

1. Using the tutorial, pg. 7 and equation (41), it follows that

$$\pi_k = \frac{\sum_{i=1}^{n} \gamma_{ik}}{\sum_{i=1}^{n} \sum_{j=1}^{K} \gamma_{ij}} = \frac{1}{n} \sum_{i=1}^{n} \gamma_{ik},$$

where $\sum_{i=1}^{n} \sum_{j=1}^{K} \gamma_{ij} = n$.

2. You can also derive the estimate using the Lagrangian.

Derivation of the M-Step

$$\mathcal{L}(\lambda, \pi, \alpha) = \sum_{i=1}^{n} \sum_{k=1}^{K} [\gamma_{ik} \log \pi_k + \gamma_{ik} \log \lambda_k - \gamma_{ik} \lambda_k x_i] - \alpha(\sum_{k=1}^{K} \pi_k - 1).$$

$$\frac{\partial \mathcal{L}}{\partial \pi_k} = \frac{\sum_{i=1}^{n} \gamma_{ik}}{\pi_k} - \alpha = 0 \implies \tag{8}$$

$$\pi_k = \frac{1}{\alpha} \sum_{i=1}^{n} \gamma_{ik} \implies \tag{9}$$

$$\sum_{k=1}^{K} \pi_k = \frac{1}{\alpha} \sum_{k=1}^{K} \sum_{i=1}^{n} \gamma_{ik} \implies \tag{10}$$

$$1 = \frac{1}{\alpha} \sum_{k=1}^{K} \sum_{i=1}^{n} \gamma_{ik} \implies \tag{11}$$

$$\alpha = \sum_{k=1}^{K} \sum_{i=1}^{n} \gamma_{ik} = n \tag{12}$$

Derivation of the M-Step

Putting this together, we find that

$$\pi_k = \frac{\sum_{i=1}^{n} \gamma_{ik}}{\sum_{i=1}^{n} \sum_{j=1}^{K} \gamma_{ij}} = \frac{1}{n} \sum_{i=1}^{n} \gamma_{ik}.$$

Monitor the log-likelihood

Recall that the log-likelihood is computed as follows:

$$L(\pi, \lambda) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \pi_k \cdot \text{dexp}(x_i \mid \lambda_k) \right),$$

where dexp() represents the Exponential density.

Specifically for an iteration $t+1$ and t we can monitor

$$\Delta L^{(t+1)} = |\Delta L^{(t+1)} - \Delta L^{(t)}| < \epsilon$$

Why can we monitor this? We estimate the parameters using the EM algorithm! (See the code below for a connection to the mathematical formulation and implementation).

Monitor the log-likelihood

Again, recall that

$$Q(\lambda_1, \ldots, \lambda_K) = \sum_{i=1}^{n} \sum_{k=1}^{K} \gamma_{ik} [\log \pi_k + \log \lambda_k - \lambda_k x_i]$$

Formally, this is the expected log-likelihood, which can be used an approximation to monitor convergence of the EM algorithm.

Monitor the expected log-likelihood

Specifically for an iteration $t+1$ and t we can monitor

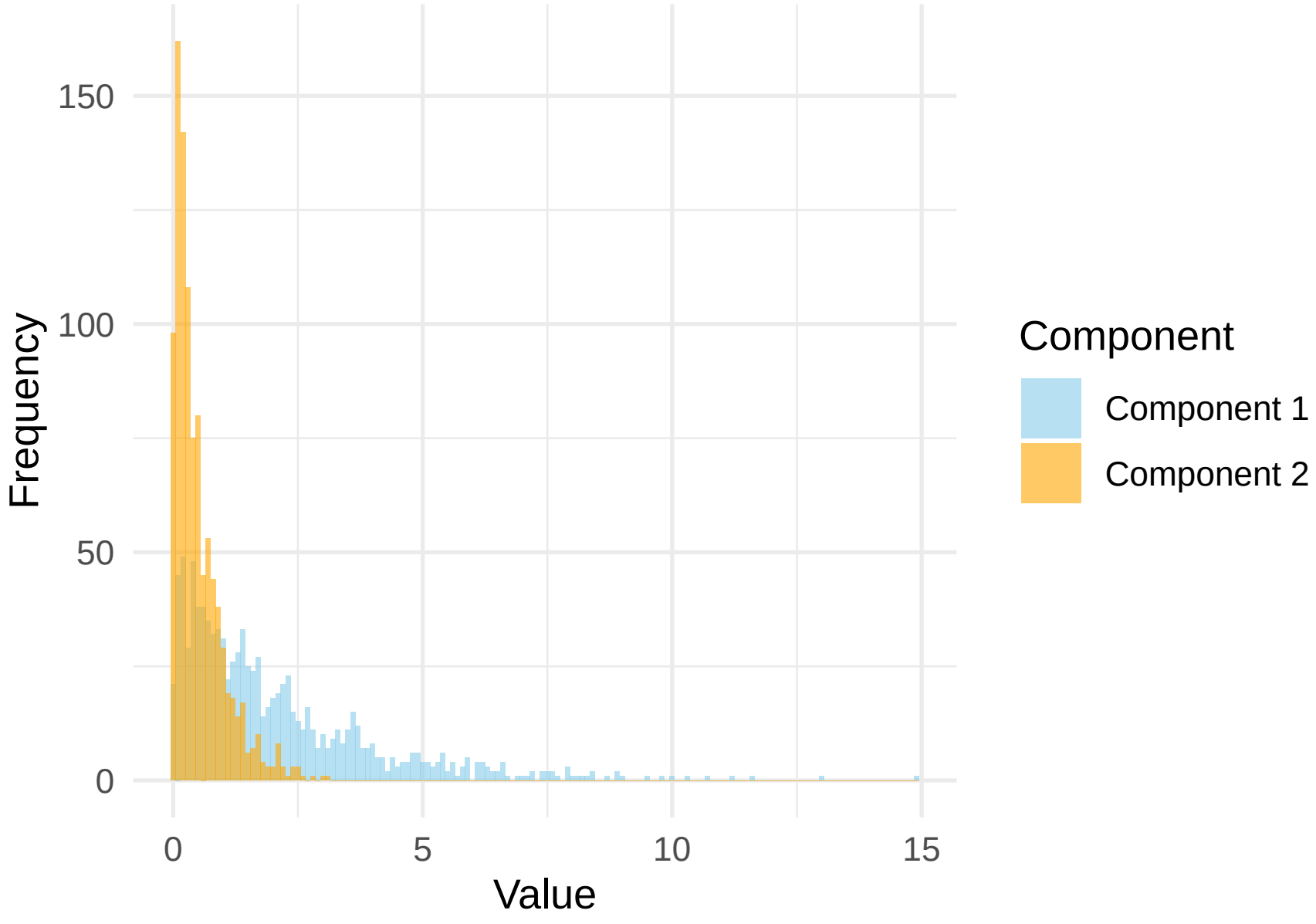
$$\Delta Q^{(t+1)} = |\Delta Q^{(t+1)} - \Delta Q^{(t)}| < \epsilon$$

We could alternatively plot the expected log-likelihood versus the number iterations, however, this is not done in my code.

Simulation Study

Simulate a two-component exponential mixture model with 2,000 data points. Let the true rates be 0.5 and 2.0. Let the mixing proportions be 0.5 and 0.5.

Simulation Study



Code

```r
exponentialMixture <- function(data, K, max_iter = 1000, tol = 1e-5) {
  n <- length(data)
  pi <- rep(1/K, K)  # mixing proportions
  lambda <- runif(K, 0.1, 1)  # rate parameters
  log_likelihoods <- numeric(max_iter)  # store log-likelihood values

  for (iter in 1:max_iter) {   # E-step: numerator of the gammas
    gamma <- matrix(NA, nrow = n, ncol = K)
    for (k in 1:K) {
      gamma[, k] <- pi[k] * dexp(data, rate = lambda[k])
    }
    row_sums <- rowSums(gamma) # denominator of the gammas
    gamma <- gamma / row_sums  # normalize probabilities

    pi_old <- pi; lambda_old <- lambda
    # M-step: Update mixing proportions and rate parameters
    pi <- colMeans(gamma)  # update mixing proportions
    for (k in 1:K) {
      lambda[k] <- sum(gamma[, k]) / sum(gamma[, k] * data)  # update rate parameters
    }

    log_likelihoods[iter] <- sum(log(row_sums)) # calculate log-likelihood

    if (max(abs(pi - pi_old)) < tol && max(abs(lambda - lambda_old)) < tol) {
      log_likelihoods <- log_likelihoods[1:iter]  # trim to the number of iterations
      cat("Convergence reached at iteration", iter,
          "with log-likelihood:", log_likelihoods[iter], "\n")
      break
    }
  }
  return(list(pi = pi, lambda = lambda, log_likelihood = log_likelihoods))
}
```

Code

Why does `sum(log(row_sums))` correspond to the log-likelihood?

In the E-step, we calculate the gamma values using Bayes Theorem for each data point belonging to each component.

```
# https://www.geeksforgeeks.org/apply-lapply-sapply-and-ta
gamma <- sapply(1:K,
                function(k) pi[k] * dexp(data,
                    rate = lambda[k]))
row_sums <- rowSums(gamma)
gamma <- gamma / row_sums
```

- gamma is the matrix of weighted probabilities
- row_sums are the total probabilities for each data point under the current parameters

Mathematically, row_sums represents $\sum_k \pi_k \text{dexp}(x_i \mid \lambda_k)$ So taking the log and then summing gives the entire log-likelihood

Results

```r
set.seed(1234)
library(knitr)
# Fit the mixture model
result <- exponentialMixture(data, K = 2)
```

```
## Convergence reached at iteration 229 with log-likelihood: -2390.612
```

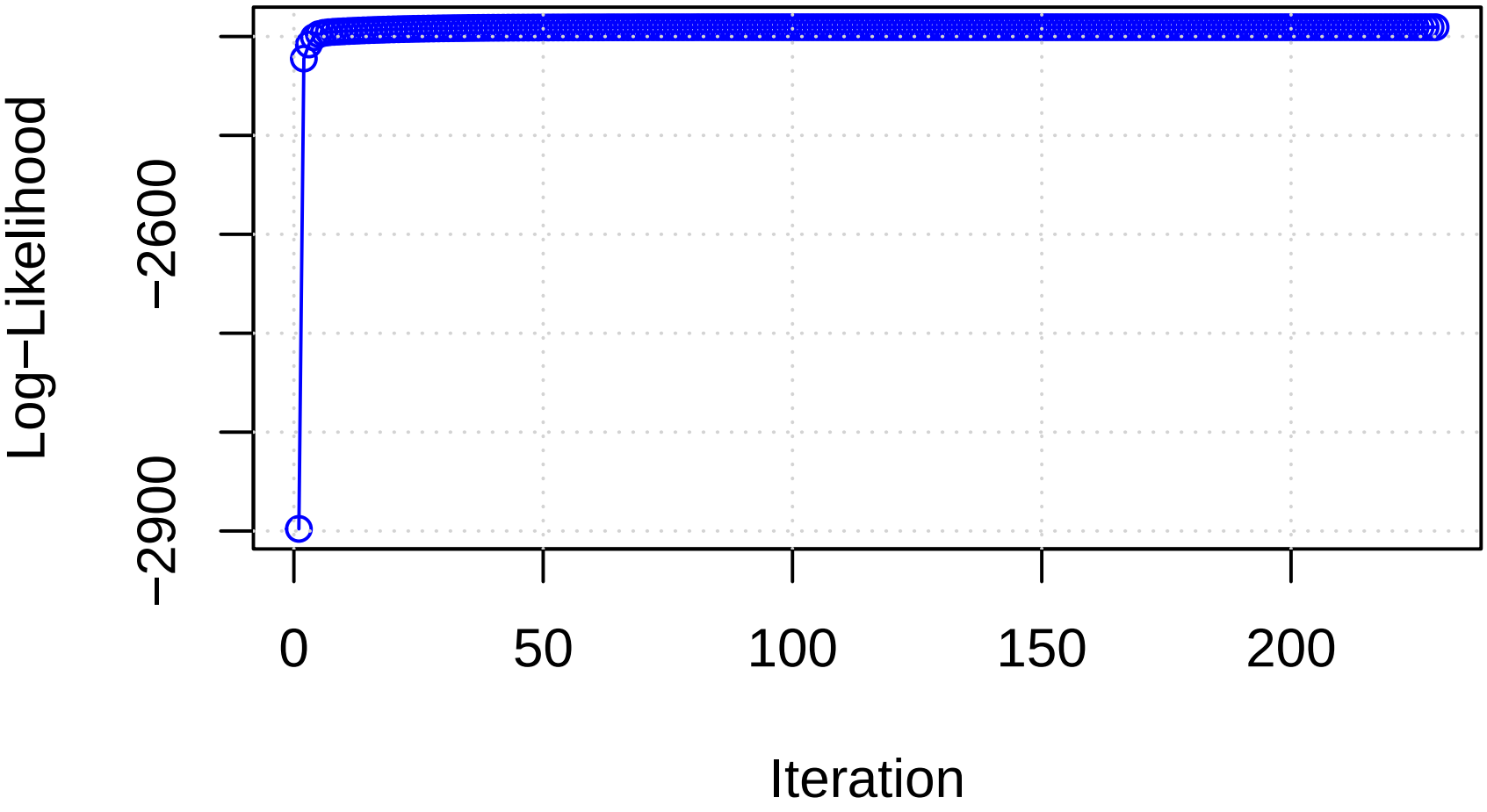
```r
# Create a data frame to display the results in a table
results_table <- data.frame(
  "Component" = 1:2,
  "Estimated Mixing Proportion" = round(result$pi, 4),
  "Estimated Rate Parameter (lambda)" = round(result$lambda, 4)
)

# move results to a table
kable(results_table, caption = "Estimated Parameters for the Mixture Model", format = "pipe")
```

Table 1: Estimated Parameters for the Mixture Model

Component	Estimated.Mixing.Proportion	Estimated.Rate.Parameter..lambda.
1	0.5135	0.4998
2	0.4865	1.9721

Log-Likelihood Plot



Findings

- Observe that the parameter estimates are close to the true values.
- Running this multiple times, should result similar findings and similar estimates each time.
- Why would the results change even if we re-run things and have a seed?
- What does the expected log-likelihood plot indicate?

Second Simulation Study

In this part, I am giving you a second simulation study.

Results

The true rate parameters are 0.5 and 1.5. The true mixing proportions are 0.6 and 0.4.

```
## Convergence reached at iteration 720 with log-likelihood: -1383.459
```

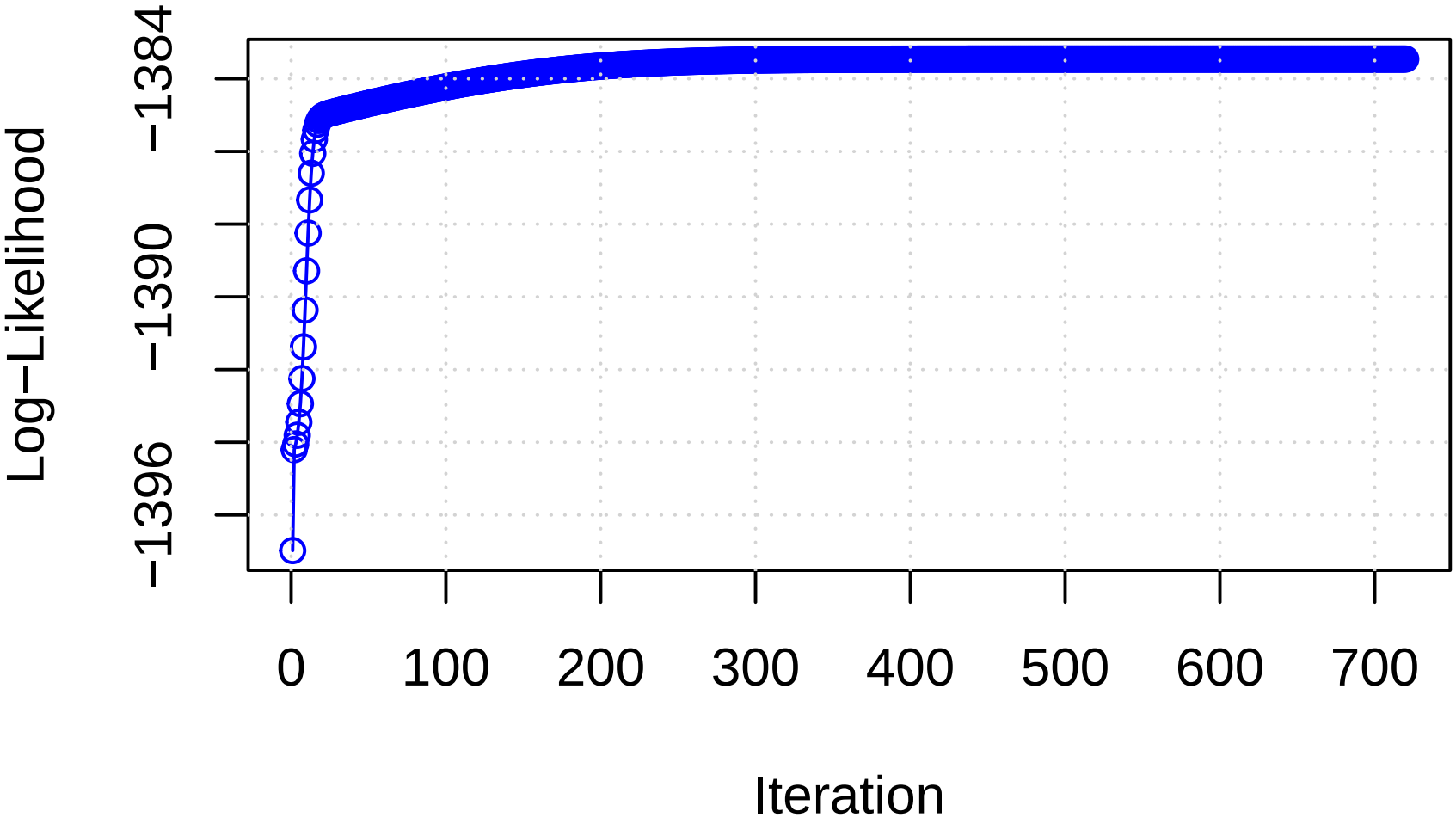
Table 2: Estimated Parameters for the Mixture Model

Component	Estimated.Mixing.Proportion	Estimated.Rate.Parameter..lambda.
1	0.2327	1.9504
2	0.7673	0.5624

What are your observations?

1. Are the estimates close to the truth?
2. What could be the cause of the issue?
3. How could we potentially remedy the issue?

Log-Likelihood Plot



Second Simulation Study

Go dig into what you think is going on here!

Next Steps

- What would you do next for data analysis to investigate this simulation study?
- How can you utilize this code for the second simulation study?
- My major recommendation to prepare for the second exam is to go through these review slides and fully understand this material as we have spend a large amount of time on it in and out of class.
- Exam II will test your knowledge of mixture models to allow you to dig deeply into one problem instead of testing your knowledge at the surface level on many problems.

Exam II

This will very similar to the exponential exercise, however, most of the code you will have available to make the task easier as this proved to be difficult for students to implement.

Supplementary Materials

Below I am providing some questions that students asked regarding some question that came up.

Why is there randomness in the EM algorithm?

The main randomness occurs from the parameter initialization.

The EM algorithm assumes initial parameter estimates for the model. If the parameters are chosen randomly, the starting values can affect the convergence of the algorithm and the final solution, especially in cases where the log-likelihood has multiple local maxima.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society: Series B (Methodological), 39(1), 1–38.

Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer. See Chapter 9.

Can other randomness occur?

It depends.

In the variant we have introduced, the EM is deterministic after the initialization, so there is no other randomness after this step.

There are other variants that include an extra random component.

- It is good to be aware that there are variants of the EM algorithm that include randomness, such as stochastic EM or Monte Carlo EM.

Neal, R. M., & Hinton, G. E. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. In Learning in Graphical Models (pp. 355-368). Springer, Dordrecht.