Exponential Mixture Models applied to Simulation Study (Part II)

Rebecca C. Steorts

This document contains the solution to Part II of the simulation study, where there is a function below called `exponentialMixture` that implements our derivations from Part I. Furthermore, you will see that the estimates are quite similar to the true values. The function also monitors the log-loglikelihood, and we provide a plot for this over the number of EM iterations.

```r
# Load necessary libraries
library(ggplot2)
exponentialMixture <- function(data, K, max_iter = 1000, tol = 1e-5) {
  n <- length(data)

  # Initialize mixing proportions (pi) and rate parameters (lambda)
  pi <- rep(1/K, K)  # Mixing proportions
  lambda <- runif(K, 0.1, 1)  # Rate parameters

  log_likelihoods <- numeric(max_iter)  # To store log-likelihood values

  for (iter in 1:max_iter) {
    # E-step: Compute responsibilities or rather the gamma values
    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)
    gamma <- gamma / row_sums  # Normalize probabilities

    # M-step: Update mixing proportions and rate parameters
    pi_old <- pi
    lambda_old <- lambda

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

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

    # Check for convergence
    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))
}
```

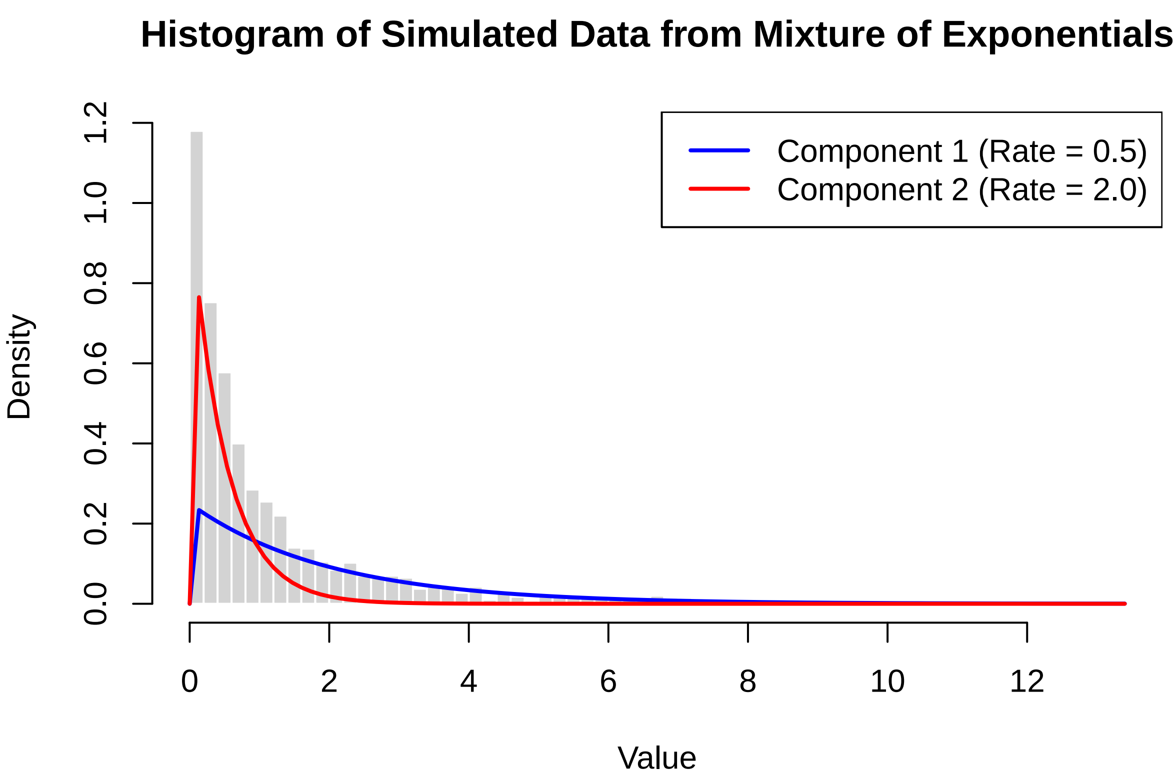
Below, we simulate some data, so that we can sanity check our code and our results before applying it to a real data set. The simulated data set is set up such that we hopefully will not run into convergence issues.

```
# Simulation parameters
n_samples <- 2000  # Increased sample size for better separation
true_rates <- c(0.5, 2.0)  # More distinctly separated rates
true_proportions <- c(0.5, 0.5)  # Balanced mixing proportions

# Generate synthetic data from a mixture of two exponential distributions
data <- c(rexp(n_samples * true_proportions[1], rate = true_rates[1]),
          rexp(n_samples * true_proportions[2], rate = true_rates[2]))

# Plot the generated data
hist(data, breaks = 50, probability = TRUE, col = 'lightgray',
     main = 'Histogram of Simulated Data from Mixture of Exponentials',
     xlab = 'Value', border = 'white')

# Overlay the true density for visualization
curve(0.5 * dexp(x, rate = true_rates[1]), add = TRUE, col = 'blue', lwd = 2)
curve(0.5 * dexp(x, rate = true_rates[2]), add = TRUE, col = 'red', lwd = 2)
legend("topright", legend = c("Component 1 (Rate = 0.5)", "Component 2 (Rate = 2.0)"),
       col = c("blue", "red"), lwd = 2)
```




```
# Fit the mixture model
result <- exponentialMixture(data, K = 2)
```

```
## Convergence reached at iteration 169 with log-likelihood: -2263.103
```

```r
# Print results
print(paste("Estimated Mixing Proportions:", round(result$pi, 4)))
```

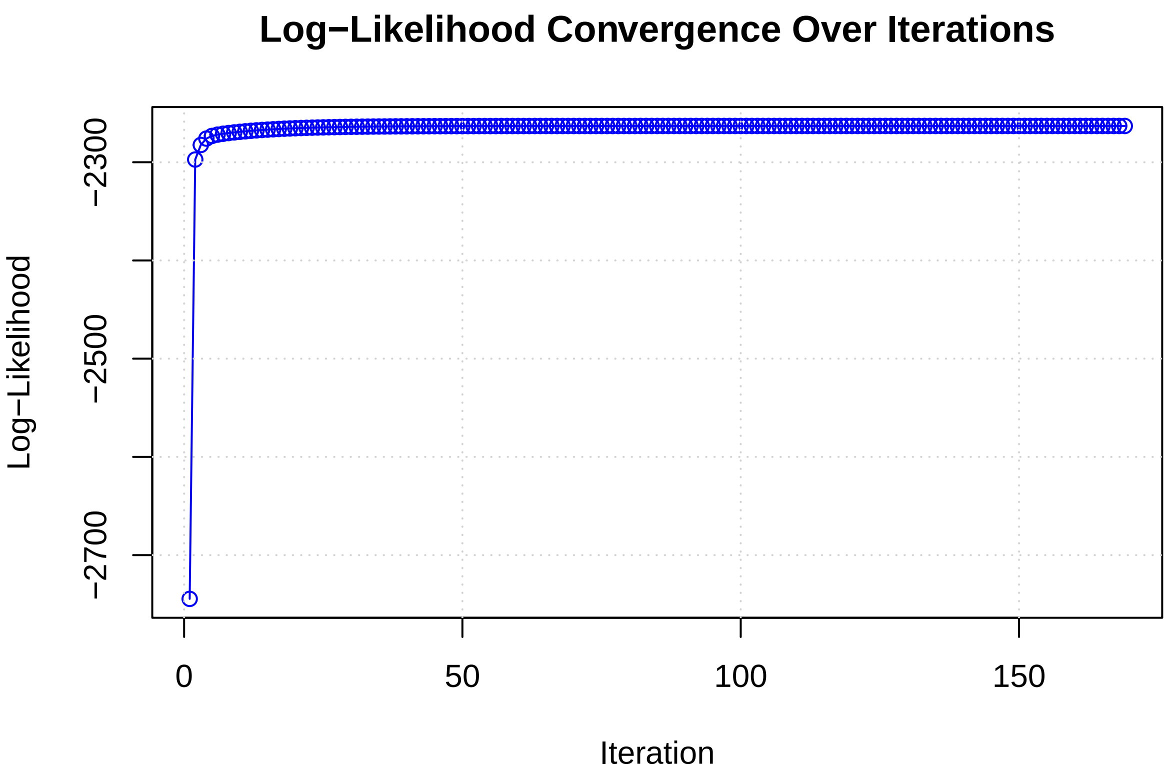
```
## [1] "Estimated Mixing Proportions: 0.4872"
## [2] "Estimated Mixing Proportions: 0.5128"
```

```r
print(paste("Estimated Rate Parameters:", round(result$lambda, 4)))
```

```
## [1] "Estimated Rate Parameters: 2.316"  "Estimated Rate Parameters: 0.5108"
```

The mixture model recovers the true parameters pretty well in this situation.

```r
log_likelihoods <- result$log_likelihood
plot(log_likelihoods, type = "o", col = "blue", xlab = "Iteration",
     ylab = "Log-Likelihood", main = "Log-Likelihood Convergence Over Iterations")
grid()
```



Hopefully, you have learned that coding up a function can be hard if you have not done this before. This of course is good practice for understanding the functions that you are utilizing in R and having a better understanding of machine learning methods in general and how they work if this is something that you might have an interest in moving forward. What would next steps look like? We would want to test our function out on many simulation studies and then we could reapply this example on the Snoq. Falls case study to see if it has better performance than a Gaussian mixture model. (Return to that module and see if you can tie together these concepts regarding mixture models in general).