Exam-2

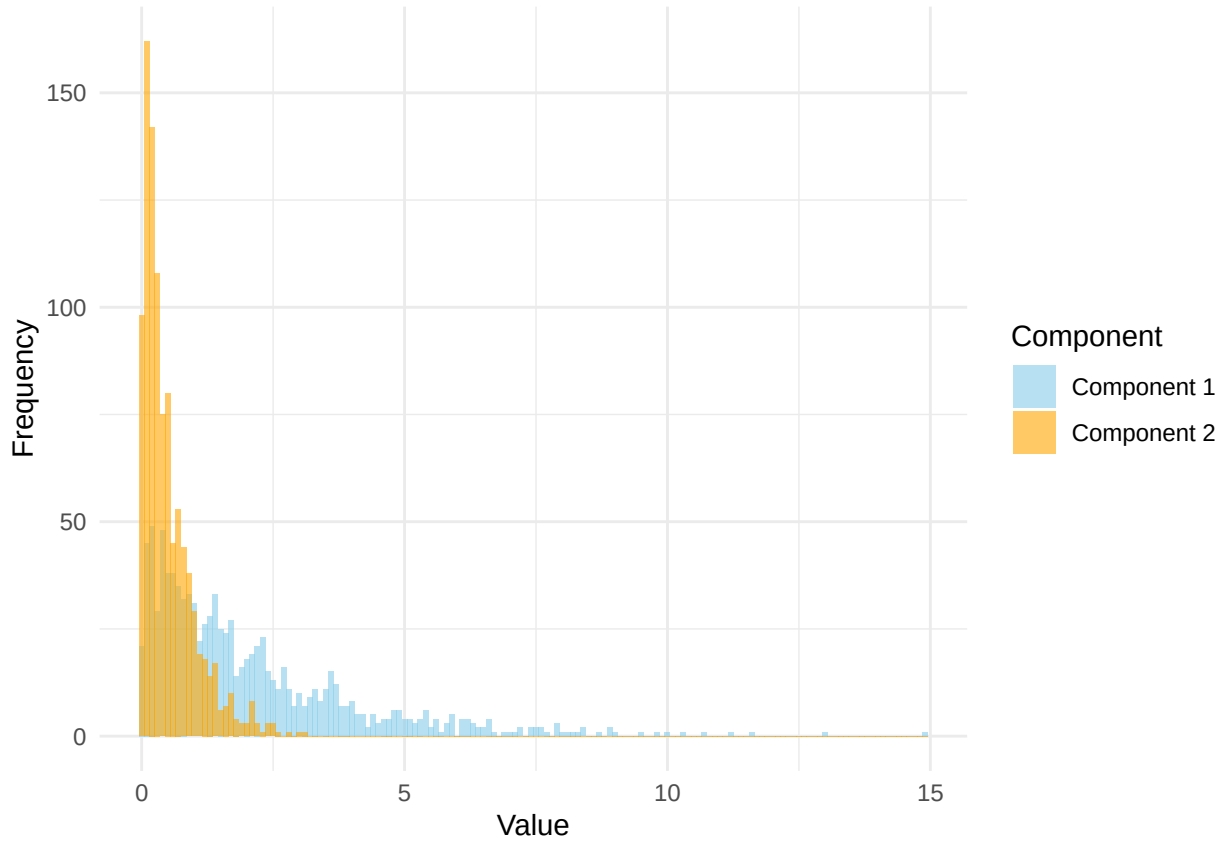
2024-11-16

Part I 1.a) What occurs in the E-step: Updates the probability that each data points belong to a specific cluster 1.b) What occurs in the M-Step: Updates the estimates of the parameters - both mixing coefficients and rate parameters. It leverages the probabilities calculated from E-step to get the parameter estimates by maximizing the likelihood.

1.c) Clusters can overlap in soft clustering because the M-step requires mixture coefficients to sum to one, allowing data points to belong to multiple clusters.

Part II

```
library(ggplot2)
set.seed(1234)
# sample
n_samples <- 2000
true_rates <- c(0.5, 2.0)
true_proportions <- c(0.5, 0.5)
# Sample group assignments (1 or 2) based on mixing proportions
component_labels <- sample(1:2, n_samples, replace = TRUE, prob = true_proportions)
# Stage 2: Generate data from the corresponding exponential
# distribution based on the group assignment
data <- numeric(n_samples) # Initialize the data vector
# Generate data for group 1 (rate = 0.5)
data[component_labels == 1] <- rexp(sum(component_labels == 1), rate = true_rates[1])
# Generate data for group 2 (rate = 2.0)
data[component_labels == 2] <- rexp(sum(component_labels == 2), rate = true_rates[2])
# Create a data frame for plotting
plot_data <- data.frame(
Value = data, # Data points
Component = factor(component_labels, labels = c("Component 1", "Component 2")) # Component labels
)
# Plot the data using ggplot2
ggplot(plot_data, aes(x = Value, fill = Component)) +
geom_histogram(binwidth = 0.1, position = "identity", alpha = 0.6) +
scale_fill_manual(values = c("skyblue", "orange")) + # Set colors for the components
labs(
x = "Value",
y = "Frequency",
fill = "Component"
) +
theme_minimal() +
theme(plot.title = element_blank()) # Remove the title
```




```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 # M-step: Update mixing proportions and rate parameters
  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
  }
  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))
}
```

2.a) Cite: Exam Review Slide 26

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

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

```
# 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

The true rates for cluster 1 and cluster 2 are 0.5 and 2.0 respectively, while the mixing proportions are 0.5 and 0.5. The estimates closely resemble the true rates (0.4998 for cluster 1 and 1.9721 for cluster 2; both with error smaller than 0.02) and the mixing proportion (0.5135 for cluster 1 and 0.4865 for cluster; both with error less than 0.02).

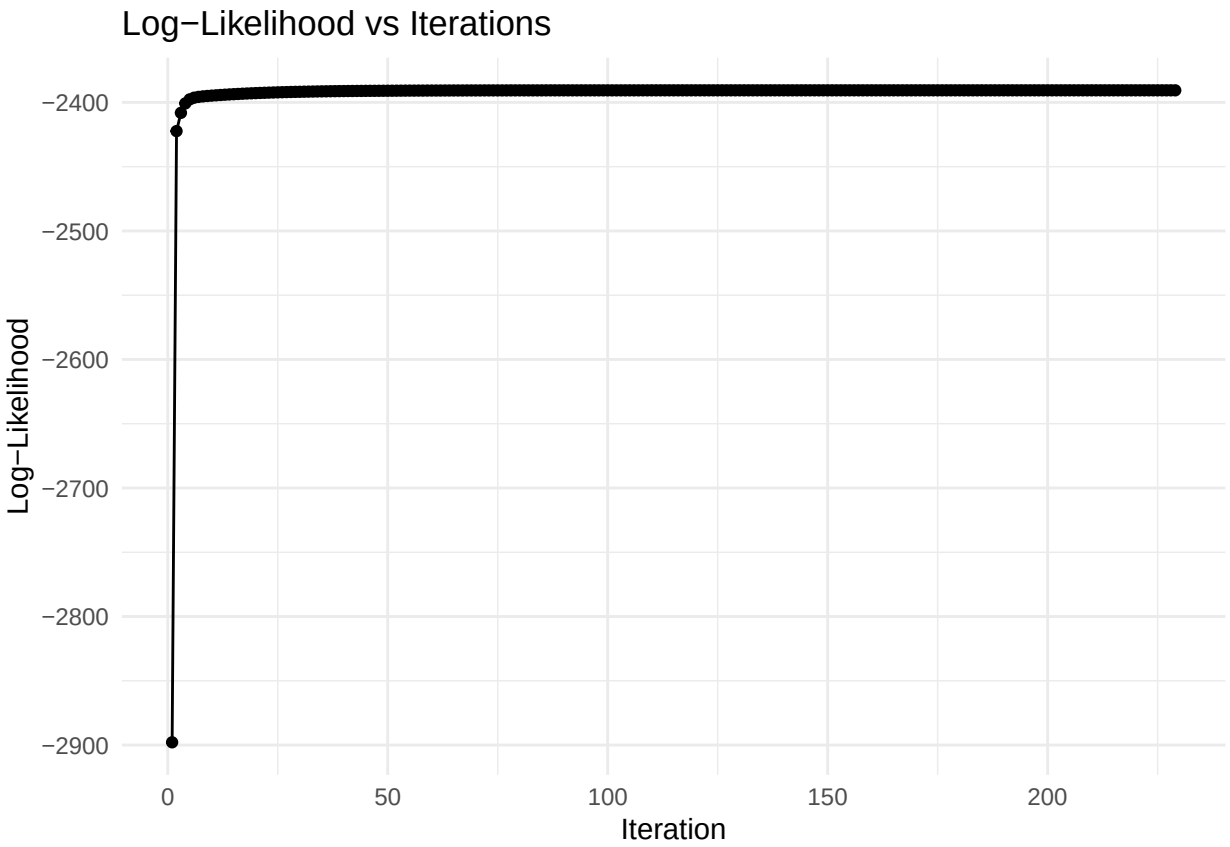
2.b) The parameter estimates do appear close to the true estimates. Cluster 1's true rate is 0.5 and its estimated rate is 0.4998, while cluster 2's true rate is 2.0 and its estimated rate is 1.9721. The true mixing proportions are 0.5 and the estimated proportion are 0.5135 and 0.4865. All the errors are less than 0.002.

2.c)

```
# Create a data frame for plotting
log_likelihoods <- result[["log_likelihood"]]

log_likelihood_data <- data.frame(
  Iteration = 1:length(log_likelihoods),
  LogLikelihood = log_likelihoods
)

# Generate the plot
ggplot(log_likelihood_data, aes(x = Iteration, y = LogLikelihood)) +
  geom_line() +
  geom_point() +
  labs(
    title = "Log-Likelihood vs Iterations",
    x = "Iteration",
    y = "Log-Likelihood"
  ) +
  theme_minimal()
```



The log-likelihood becomes stable after roughly 50 iterations, suggesting that the EM algorithm has converged to the optimal parameters (local maximum).

2.d) We typically monitor convergence of the algorithm using the log-likelihood.(Exam Review Slide 12) It is also evident by the code as we calculated the log likelihood (log_likelihoods[iter] <- sum(log(row_sums))) and didn't include the latent variable.

2.e)

```
n = 5

# an empty data frame to store results
results <- data.frame()
set.seed(1234)
for (run in 1:n) {


  # Fit the mixture model
  result <- exponentialMixture(data, K = 2)
  results <- rbind(
    results,
    data.frame(
      Run = run,
      Component = 1:2,  # Components 1 and 2
      Mixing_Proportion = result$pi,
      Rate_Parameter = result$lambda
    )
  )
```

```
}
```

```
## Convergence reached at iteration 229 with log-likelihood: -2390.612
## Convergence reached at iteration 203 with log-likelihood: -2390.612
## Convergence reached at iteration 198 with log-likelihood: -2390.612
## Convergence reached at iteration 237 with log-likelihood: -2390.612
## Convergence reached at iteration 203 with log-likelihood: -2390.612
```

```
kable(results, caption = "Estimates from Five Runs of the EM Algorithm", format = "pipe")
```

Table 2: Estimates from Five Runs of the EM Algorithm

Run	Component	Mixing_Proportion	Rate_Parameter
1	1	0.5135382	0.4998441
1	2	0.4864618	1.9721110
2	1	0.5135368	0.4998434
2	2	0.4864632	1.9721059
3	1	0.4864629	1.9721072
3	2	0.5135371	0.4998436
4	1	0.5135381	0.4998440
4	2	0.4864619	1.9721106
5	1	0.4864611	1.9721136
5	2	0.5135389	0.4998445

The cluster labeling switches sometimes (Run 1 cluster 1 is the true cluster 1 with rate of 0.5, but Run 2 Cluster 2 is the true cluster 1). The estimated cluster rates and proportion vary slightly across runs. For examples, Run 2 Cluster 1 and Run 4 Cluster 1 both correspond to true cluster 1 but have estimated proportion of 0.4864629 and 0.4864611 (error less than 0.00001).

2.f) I found the most difficult when trying to code the latent variable. The exam review and the tutorials were particularly helpful for clarifying questions. Having tutorials to follow and check answers with could make learning on material easier in the future.

Part III.

```
# Simulation parameters
set.seed(123) # For reproducibility
n_samples <- 1000 # Number of samples
true_rates <- c(0.5, 1.5) # True rate parameters for two components
true_proportions <- c(0.6, 0.4) # True mixing proportions
# Generate synthetic data
data <- c(rexp(n_samples * true_proportions[1], rate = true_rates[1]),
rexp(n_samples * true_proportions[2], rate = true_rates[2]))
```

3.a)

```
n = 5

# an empty data frame to store results
results2 <- data.frame()
set.seed(1234)
for (run in 1:n) {
  # Fit the mixture model
  result2 <- exponentialMixture(data, K = 2)
  results2 <- rbind(
```

```
    results2,
    data.frame(
      Run = run,
      Component = 1:2,  # Components 1 and 2
      Mixing_Proportion = result2$pi,
      Rate_Parameter = result2$lambda
    )
  )
}
```

```
## Convergence reached at iteration 774 with log-likelihood: -1383.459
## Convergence reached at iteration 724 with log-likelihood: -1383.459
## Convergence reached at iteration 707 with log-likelihood: -1383.459
## Convergence reached at iteration 810 with log-likelihood: -1383.459
## Convergence reached at iteration 717 with log-likelihood: -1383.459
```

```
kable(results2, caption = "Estimates from Five Runs of the EM Algorithm", format = "pipe")
```

Table 3: Estimates from Five Runs of the EM Algorithm

Run	Component	Mixing_Proportion	Rate_Parameter
1	1	0.7673356	0.5624362
1	2	0.2326644	1.9503495
2	1	0.7673356	0.5624362
2	2	0.2326644	1.9503494
3	1	0.2326643	1.9503503
3	2	0.7673357	0.5624362
4	1	0.7673367	0.5624366
4	2	0.2326633	1.9503564
5	1	0.2326629	1.9503589
5	2	0.7673371	0.5624367

3.b) The cluster labeling switches sometimes (Run 1 cluster 1 is the true cluster 1 with rate of 0.5, but Run 3 Cluster 2 is the true cluster 1). The estimated parameters vary between runs and vary significantly from the true estimates. Est.proportions are around (0.7673,0.2327) contrasted with the true proportions (0.6,0.4), while the est.rates are around (0.5624,1.9503) contrasted with the true rates (0.5,1.5).

3.c) The root problem of significant difference between est.parameters and the true parameters can be a result that the EM algorithm converges to a local maximum, potentially due to the poor choice of initialization values.

3.d) K-means clustering can help with the root problem. We can group the data in clusters and estimate the mixing proportion using proportion of points in each cluster and rate using the inverse of the mean of data points in the cluster (since this is an exponential mixture model with a mean of 1/rate).

3.extra Credit)

```
#Cite: below from data-clean/06-more-clustering/code/kclust.R
set.seed(1234)
exponentialMixture_kmeans <- function(data, K, max_iter = 1000, tol = 1e-5) {
  n <- length(data)# Initialize parameters using k-means
  kmeans_result <- kmeans(data, centers = K)
  cluster_assignments <- kmeans_result$cluster

  # Mixing proportions
```

```r
  pi <- table(cluster_assignments) / n

  # Rate parameters
  lambda <- sapply(1:K, function(k) {
    1 / mean(data[cluster_assignments == k])
  })

  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 # M-step: Update mixing proportions and rate parameters
    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
    }
    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))
  }
```

```r
n = 5

# an empty data frame to store results
results3 <- data.frame()
#set.seed(1234)
for (run in 1:n) {
  # Fit the mixture model
  result3 <- exponentialMixture_kmeans(data, K = 2)
  results3 <- rbind(
    results3,
    data.frame(
      Run = run,
      Component = 1:2,  # Components 1 and 2
      Mixing_Proportion = result3$pi,
      Rate_Parameter = result3$lambda
    )
  )
}
```

```
## Convergence reached at iteration 856 with log-likelihood: -1383.459
## Convergence reached at iteration 856 with log-likelihood: -1383.459
```

```
## Convergence reached at iteration 856 with log-likelihood: -1383.459
## Convergence reached at iteration 856 with log-likelihood: -1383.459
## Convergence reached at iteration 856 with log-likelihood: -1383.459
```

```
kable(results3, caption = "Estimates from Five Runs of the EM Algorithm", format = "pipe")
```

Table 4: Estimates from Five Runs of the EM Algorithm

Run	Component	Mixing_Proportion	Rate_Parameter
1	1	0.7673365	0.5624365
1	2	0.2326635	1.9503554
2	1	0.2326635	1.9503554
2	2	0.7673365	0.5624365
3	1	0.7673365	0.5624365
3	2	0.2326635	1.9503554
4	1	0.2326635	1.9503554
4	2	0.7673365	0.5624365
5	1	0.7673365	0.5624365
5	2	0.2326635	1.9503554

K-means doesn't work as well in this case likely because k-means work best for cluster/spherical shape while exponential distribution's shape differs. It is also difficult to escape the local optima once the algorithm is stuck in there.

3.e) The first simulation provided a foundational understanding of mixture models and the EM algorithm, including coding it, how the E-step integrates with the M-step, and managing the algorithm's inherent randomness. The second simulation highlighted the limitations of the EM algorithm, particularly its tendency to get trapped in local maxima, and explored potential solutions. Moving forward, I will focus on selecting reasonable initial values, even in the absence of known truths, and applying techniques like clustering to enhance the accuracy and precision of the EM algorithm.