Exponential Mixture Models

Rebecca C. Steorts, STA 325

Introduction

Often, we assume that data points come from a distribution, such as a Gaussian, and we one way to estimate the unknown parameters, such as the mean or variance using maximum likelihood estimation. However, this assumption always does not hold in practice.

As we saw in the introduction to mixture models, data points may not be unimodal and can be multi-modal. This means that assuming the sample occurs from a unimodal distribution, such as the Gaussian is too restrictive. One way to handle multi modal data is using mixture models, which provides a probabilistic way to model the data (and it also has a nice interpretation regarding clustering).

Goals and Objectives

1. Understand mixture models via Ghojogh et. al (2020), which provides a tutorial on these topics and derivations. (https://arxiv.org/pdf/1901.06708)
2. Using the tutorial, derive the parameter estimates for the exponential mixture model (using the EM algorithm).
3. Unfortunately, there is no function for this in R, so we will use this as practice to create one and test it out on two simulation studies that exhibit different behavior.
4. This is an ungraded exercise meant to provide in-class interaction and support along the way and encourage students to work with others, and provide more of a real case scenario of how machine learning problems occur in practice.

Learning Objectives

By the end of the exercise, you should be able to:

- explain mixture models and the EM algorithm
- derive the parameter estimates (using the tutorial or class materials)
- understand their importance
- be able to code up the EM algorithm and parameter estimates on your own
- be able to narrate and explain your findings in writing or verbally regarding what you found in the simulation studies

Model Setup

Consider a data set $\{x_1, x_2, \ldots, x_n\}$ generated from a mixture of K exponential distributions. The mixture model is given by:

$$p(x|\theta) = \sum_{k=1}^{K} \pi_k \lambda_k e^{-\lambda_k x}$$

where:

- π_k is the mixing coefficient for the k-th component (with $\sum_{k=1}^{K} \pi_k = 1$),
- λ_k is the rate parameter of the k-th exponential distribution,
- $\theta = \{\pi_k, \lambda_k\}$ are the parameters we want to estimate.

Part I

For this exercise, please read the tutorial (https://arxiv.org/pdf/1901.06708) and you may use parts of it to help you work through the exercise and derivations.

1. Write the mixture model using an unknown (or latent) variable/component or two-stage approach so that we can estimate any unknown parameters. Assume that Z_i indicates the unknown (or latent) component that x_i is drawn from.

Remark: There are a few different ways that are fine and equally equivalent to write this that we talked about in class. Feel free to write this in that way that makes the most sense to you!

2. Write the marginal distribution of $P(x_i)$.
3. Derive the expectation (E-step) and maximization (M-step) steps of the EM algorithm.

a. Specifically for the E-step, using Bayes' Theorem, show that

$$\gamma_{ik} = P(Z_i = k|x_i) = \frac{\pi_k \lambda_k e^{-\lambda_k x_i}}{\sum_{j=1}^{K} \pi_j \lambda_j e^{-\lambda_j x_i}}.$$

For details on why the Q function can be written as it is, please refer to the tutorial as it works this out in detail and provides a similar exercise for Gaussian and Poisson mixture models.

Also, recall in the E-step, we need to calculate the expected value of the complete log-likelihood given the observed data x and the current estimates of the parameters $\theta^{(t)}$:

$$Q(\theta|\theta^{(t)}) = \mathbb{E}_{z|x,\theta^{(t)}} \left[\log L(\theta|x, z)\right]$$

For this problem, we need to work with the following function (where all these details can be found in the tutorial linked above)

$$Q(\theta|\theta^{(t)}) = \sum_{i=1}^{n} \mathbb{E}_{z_i|x_i,\theta^{(t)}} \left[\log \left(\sum_{k=1}^{K} \pi_k \lambda_k e^{-\lambda_k x_i}\right)\right]$$

Recall that at some iteration t

$$\gamma_{ik}^{(t)} = \mathbb{P}(z_i = k|x_i, \theta^{(t)}) = \frac{\pi_k^{(t)} \lambda_k^{(t)} e^{-\lambda_k^{(t)} x_i}}{\sum_{j=1}^{K} \pi_j^{(t)} \lambda_j^{(t)} e^{-\lambda_j^{(t)} x_i}}.$$

We can substitute these as follows (see the tutorial regarding why this can be done):

$$Q(\theta|\theta^{(t)}) = \sum_{i=1}^{n}\sum_{k=1}^{K} \gamma_{ik} \log\left(\pi_k \lambda_k e^{-\lambda_k x_i}\right) \tag{1}$$

$$= \sum_{i=1}^{n}\sum_{k=1}^{K} \gamma_{ik} \left(\log \pi_k + \log \lambda_k - \lambda_k x_i\right) \tag{2}$$

$$= \sum_{k=1}^{K} \left(\sum_{i=1}^{n} \gamma_{ik} \log \pi_k + \sum_{i=1}^{n} \gamma_{ik} \log \lambda_k - \sum_{i=1}^{n} \gamma_{ik} \lambda_k x_i \right) \tag{3}$$

This function $Q(\theta|\theta^{(t)})$ will be maximized in the M-step to update the parameters.

b. Show that the M-step is as follows (given the E-step):

The mixing coefficient for each component is updated as follows:

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

Show that the rate parameters are updated as follows:

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

To find π_k, we use the constraint that the mixing weights must sum to 1 use a Langrange multiplier approach, which will handle the constraint. To find, λ_k, proceed as you typically would (as there is no constraint).

4. Write the log-likelihood, which is used to to evaluate convergence of the EM algorithm. Explain how convergence can be checked using the log-likelihood.

Part II

Implement a two-component exponential mixture model (using Part I), estimating the unknown parameters using the EM algorithm. You will need to write your own code to do this.

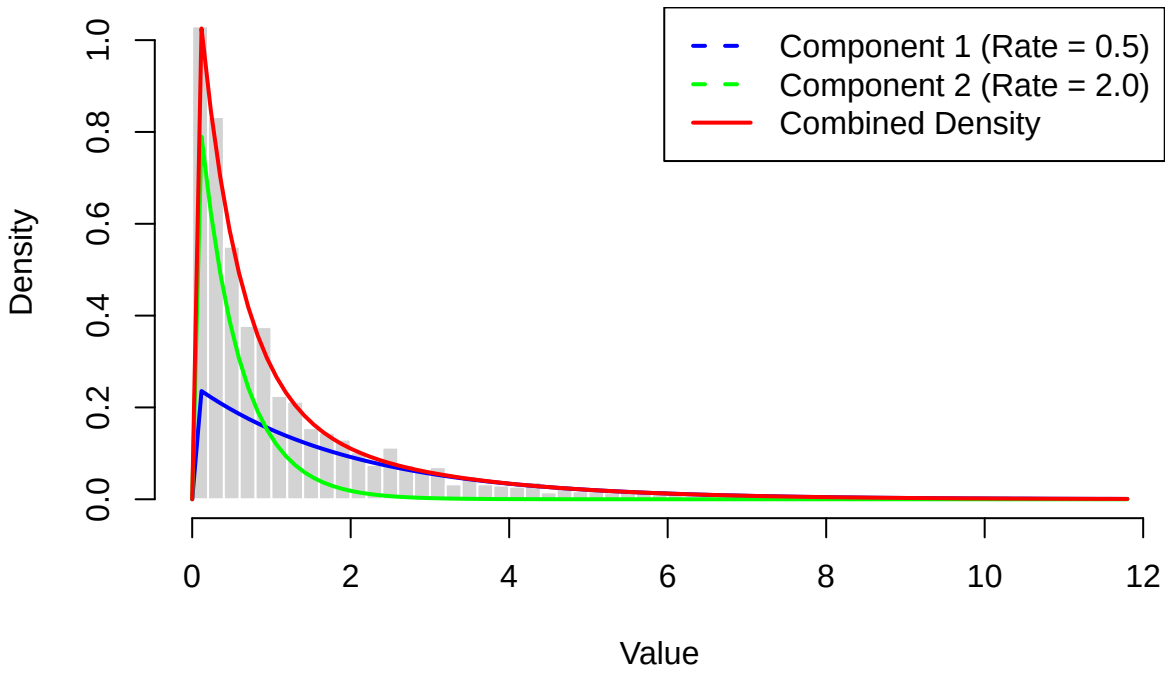
Test your code on a simulated data from an exponential mixture model, where we know what the estimated parameters should be close to, assuming the algorithm does not get stuck in a local mode. You will find such an example below.

```r
library(ggplot2)
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 = 'green', lwd = 2)
curve(0.5 * dexp(x, rate = true_rates[1]) + 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)",
                              "Combined Density"),
       col = c("blue", "green", 'red'), lty = c(2, 2, 1), lwd = 2)
```


Histogram of Simulated Data from Mixture of Exponentials
Component 1 (Rate = 0.5)
Component 2 (Rate = 2.0)
Combined Density
Density
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
10
12
Value

Hint: You could utilize the skeleton code below if you find it helpful or start with your own.

```r
# Function to fit a k-component exponential mixture model
mixtureExponetials <- 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)  # To store log-likelihood values

  # E-step: find the gamma parameter
  # (this is the result from Bayes Theorem in part I)

  # M-step: update the mixing proportions, update the rate parameters
  # (These are from the M-step in part I)


  # Calculate the log-likelihood for the current iteration

  # Check for convergence. This point can be a bit tricky
  # and you may run into errors here potentially.

  return(list(pi = pi, lambda = lambda, log_likelihood = log_likelihoods))
}
```

Remark: I often find it helpful to write and de-bug code in base R (instead of RStudio personally), however, this is all personal preference. Why?

1. I find it personally easier to debug and go through.
2. I find the debugging functions a bit easier to understand when I have an error, but this is all personal preference.
3. Find what works the best for you as it's all about finding the best workflow, and it's different for everyone.
4. If you have questions about workflow, please do ask. I love talking about it and it's a very important part of machine learning.

Remark: I often find markdown/quarto helpful for visualization or tidying up data to showcase, whereas, I find base R most effective for writing code for papers/projects. If you look into how packages are written in R, they are in base R, however vignettes are showcased in markdown. Think of why this would be the case.

Part III

Now, consider a second simulation study. Test the EM algorithm out on this simulated data set and report your findings.

a. Report the parameter estimates for five different starting values of the EM algorithm. Explain your findings and behavior. It may be helpful to look at the log-likelihood plot.

b. You should find in part a) that your estimates are in accurate but do your best to try and explain why they are not accurate. Explain what is happening to the EM algorithm that causes the estimates to be inaccurate.

c. Do you have a suggestion for improving your estimates? Hint: Think about k-means. Make a simple modification to your code. Do you find improvements? Or do you still have similar problems that you ran into in part b.

d. Explain what this has taught you about mixture models and the EM algorithm from the two different simulation studies. Given this new knowledge, what precautions should you take when working with mixture models and the EM algorithm moving forward?

```r
# 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 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]))
```

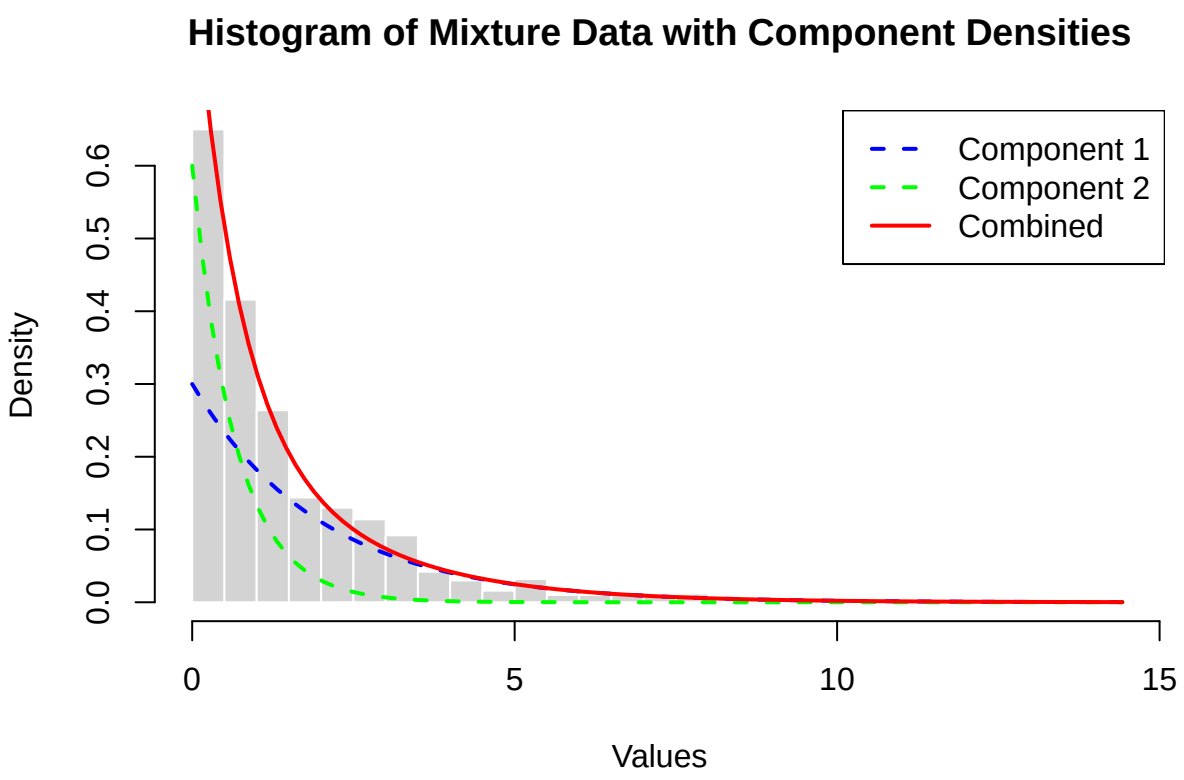
```r
# Set up the plot area
hist(data, breaks = 30, probability = TRUE,
     main = "Histogram of Mixture Data with Component Densities",
     xlab = "Values",
     col = "lightgray",
     border = "white")

# Define the x values for the density curves
x_values <- seq(0, max(data), length.out = 100)

# Calculate the densities for the components
density_component1 <- dexp(x_values, rate = true_rates[1]) * true_proportions[1]
density_component2 <- dexp(x_values, rate = true_rates[2]) * true_proportions[2]
combined_density <- density_component1 + density_component2

# Add the component densities to the plot
lines(x_values, density_component1, col = "blue", lwd = 2, lty = 2)  # Component 1
lines(x_values, density_component2, col = "green", lwd = 2, lty = 2)  # Component 2
lines(x_values, combined_density, col = "red", lwd = 2)                # Combined density

# Add a legend
legend("topright", legend = c("Component 1", "Component 2", "Combined"),
       col = c("blue", "green", "red"), lty = c(2, 2, 1), lwd = 2)
```


Histogram of Mixture Data with Component Densities
Component 1
Component 2
Combined
Density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
5
10
15
Values