Lab 6 - Computational Statistics (732A89)

Helena Llorens Lluís (hllor282), Yi Yang (yiyan338)

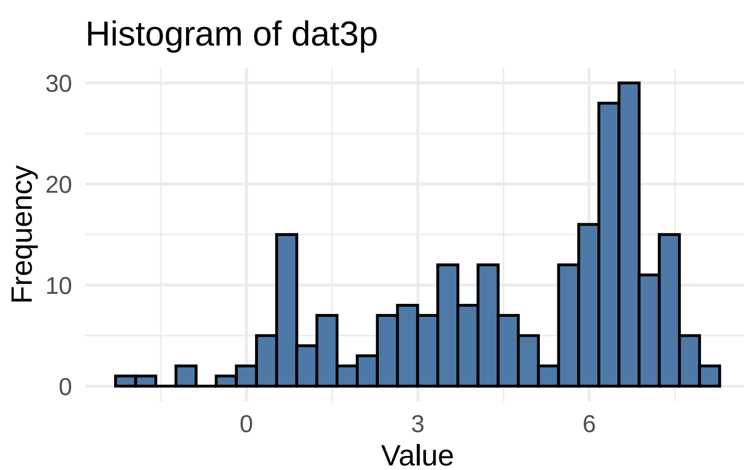
QUESTION 1: EM algorithm

Expectation-Maximization (EM) is an iterative optimization algorithm used for parameter estimation in probabilistic models with latent variables. In this report, we modify the provided EM algorithm to handle a mixture of three normal distributions.

Starting from the provided two-component EM algorithm, we introduced the following changes:

- Increased the number of mixture components from two to three.
- Adjusted the initialization of parameters accordingly.
- Modified the Expectation (E) step to compute probabilities for three components.
- Updated the Maximization (M) step to re-estimate parameters based on three components.
- Revised the stopping criterion to ensure scale-invariance by normalizing the convergence measure.

After loading the `dat3p` dataset, we plot a histogram to visualize its distribution.



The data exhibits multimodal behavior, suggesting the presence of multiple underlying distributions.

Then, we applied the EM algorithm with a stopping threshold of eps = 0.000001. The final estimated parameters of the normal mixture model are showed on the following table.

Table 1: Final estimated parameters of the normal mixture model

	p	mu	sigma
Component 1	0.3287399	2.337124	1.9199766
Component 2	0.2143430	3.979460	1.7707219
Component 3	0.4569171	6.627087	0.5635485

These values suggest that the algorithm successfully identified three distinct normal distributions underlying the dataset.

To assess convergence, we plotted the estimated μ and σ values across iterations.

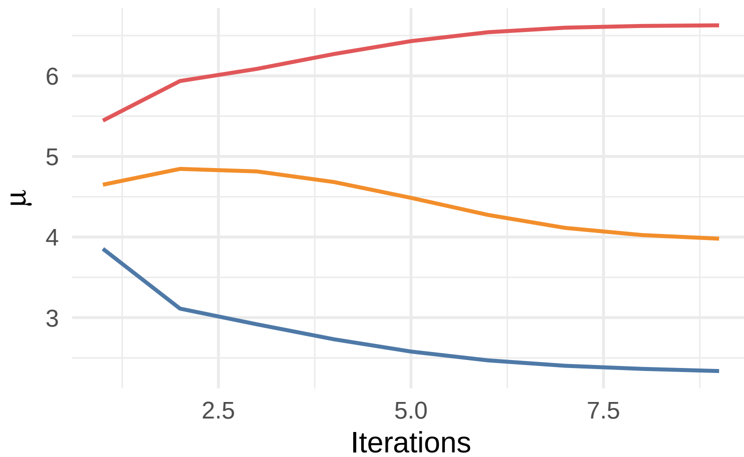


Figure 1: Plot of mu vs iteration number

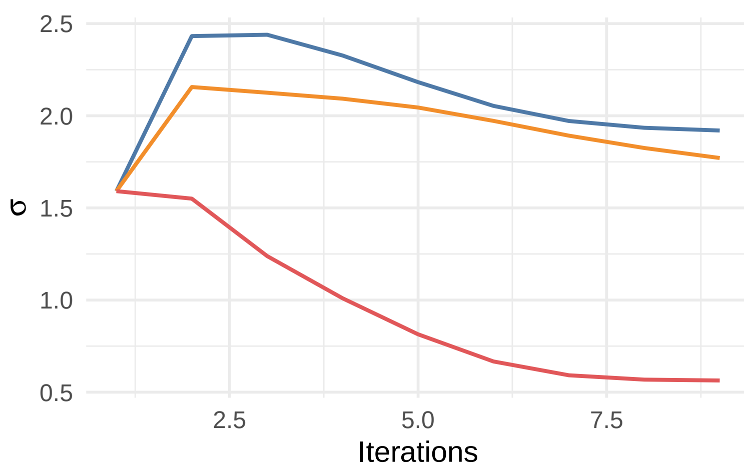


Figure 2: Plot of sigma vs iteration number

Both estimates stabilize after a certain number of iterations, supporting convergence.

The EM algorithm successfully fitted a three-component normal mixture model. The convergence plots validate that the algorithm converged for each parameter. Additionally, modifying the stopping criterion ensured robustness against data scaling.

APPENDIX

```r
library(ggplot2)
library(knitr)

# EM algorithm
em <- function(data, eps = 0.0001){
  n      <- length(data)
  pi     <- matrix(NA, ncol = 3, nrow = n)

  # Random initialization of the parameters
  p <- rep(1/3, 3)
  sigma <- rep(sd(data)*2/3, 3)
  mu <- c(mean(data) - sigma[1]/2, mean(data), mean(data) + sigma[1]/2)
  pv <- c(p, mu, sigma)
  cc  <- eps + 100
  mu_list <- c(mu)
  sigma_list <- c(sigma)

  while (cc > eps){
    # save previous parameter vector
    pv1  <- pv

    # E step
    for (j in 1:n){
      pi1 <- p[1]*dnorm(data[j], mean=mu[1], sd=sigma[1])
      pi2 <- p[2]*dnorm(data[j], mean=mu[2], sd=sigma[2])
      pi3 <- p[3]*dnorm(data[j], mean=mu[3], sd=sigma[3])

      total <- pi1+pi2+pi3
      pi[j,] <- c(pi1/total, pi2/total, pi3/total)
    }

    # M step
    p <- colSums(pi)/n
    mu[1] <- sum(pi[, 1]*data)/(p[1]*n)
    mu[2] <- sum(pi[, 2]*data)/(p[2]*n)
    mu[3] <- sum(pi[, 3]*data)/(p[3]*n)
    sigma[1] <- sqrt(sum(pi[, 1]*((data-mu[1])^2))/(p[1]*n))
    sigma[2] <- sqrt(sum(pi[, 2]*((data-mu[2])^2))/(p[2]*n))
    sigma[3] <- sqrt(sum(pi[, 3]*((data-mu[3])^2))/(p[3]*n))

    # stopping criteria
    pv <- c(p, mu, sigma)
    cc     <- sum((pv - pv1)^2) / sum(pv1^2)  # a convergence criterion, maybe not the best one
    mu_list <- rbind(mu_list, mu)
    sigma_list <- rbind(sigma_list, sigma)

  }
  res <- list(final_probs = pv[1:3],
              final_mus = pv[4:6],
              final_sigmas = pv[7:9],
              mu = mu_list,
```

```r
              sigma = sigma_list)
  return(res)
}

# load data
load(file = "D:\\liu\\CompStat\\Computational-Statistics\\lab6\\threepops.Rdata")
data <- dat3p

# Histogram of the data
df <- data.frame(data)
ggplot(df, aes(x = df[, 1])) +
  geom_histogram(col = "black", fill = "#4E79A7") +
  labs(title = "Histogram of dat3p", x = "Value", y = "Frequency") +
  theme_minimal()

# Run algorithm on the data
res <- em(data)
final_probs <- res$final_probs
final_mus <- res$final_mus
final_sigmas <- res$final_sigmas

# results table
result_table <- data.frame(final_probs, final_mus, final_sigmas)
colnames(result_table) <- c("p", "mu", "sigma")
rownames(result_table) <- c("Component 1", "Component 2", "Component 3")
kable(result_table,
      caption = "Final estimated parameters of the normal mixture model")

#  plots of current estimates for each model parameter versus the iteration-number
# mu plot
mus <- data.frame(res$mu)
ggplot(data = mus, aes(x = 1:nrow(mus))) +
  geom_line(aes(y = mus[, 1]), col = "#4E79A7", size = 0.7) +
  geom_line(aes(y = mus[, 2]), col = "#F28E2B", size = 0.7) +
  geom_line(aes(y = mus[, 3]), col = "#E15759", size = 0.7) +
  theme_minimal() +
  labs(x = "Iterations", y = expression(mu))

# sigma plot
sigmas <- data.frame(res$sigma)
ggplot(data = sigmas, aes(x = 1:nrow(sigmas))) +
  geom_line(aes(y = sigmas[, 1]), col = "#4E79A7", size = 0.7) +
  geom_line(aes(y = sigmas[, 2]), col = "#F28E2B", size = 0.7) +
  geom_line(aes(y = sigmas[, 3]), col = "#E15759", size = 0.7) +
  theme_minimal() +
  labs(x = "Iterations", y = expression(sigma))
```