Homework3

Gifty Osei

2024-10-04

Remark: If you would like to insert images for your handwritten part into this file, please refer to this article.

Problem 1. Simulated annealing

The following is a typical implementation of simulated annealing.

- Simulate ζ from a distribution with density $g(\zeta)$.
- Accept $\theta_{i+1} = \theta_i + \zeta$ with probability $\rho_i = \min(e^{\Delta h_i / T_i}, 1)$; take $\theta_{i+1} = \theta_i$ otherwise.
- Update T_i to T_{i+1}.

Write an R program to implement this algorithm to find the mode of the two-component mixture distribution

$$\frac{1}{4}N(\mu_1, 1) + \frac{3}{4}N(\mu_2, 1)$$

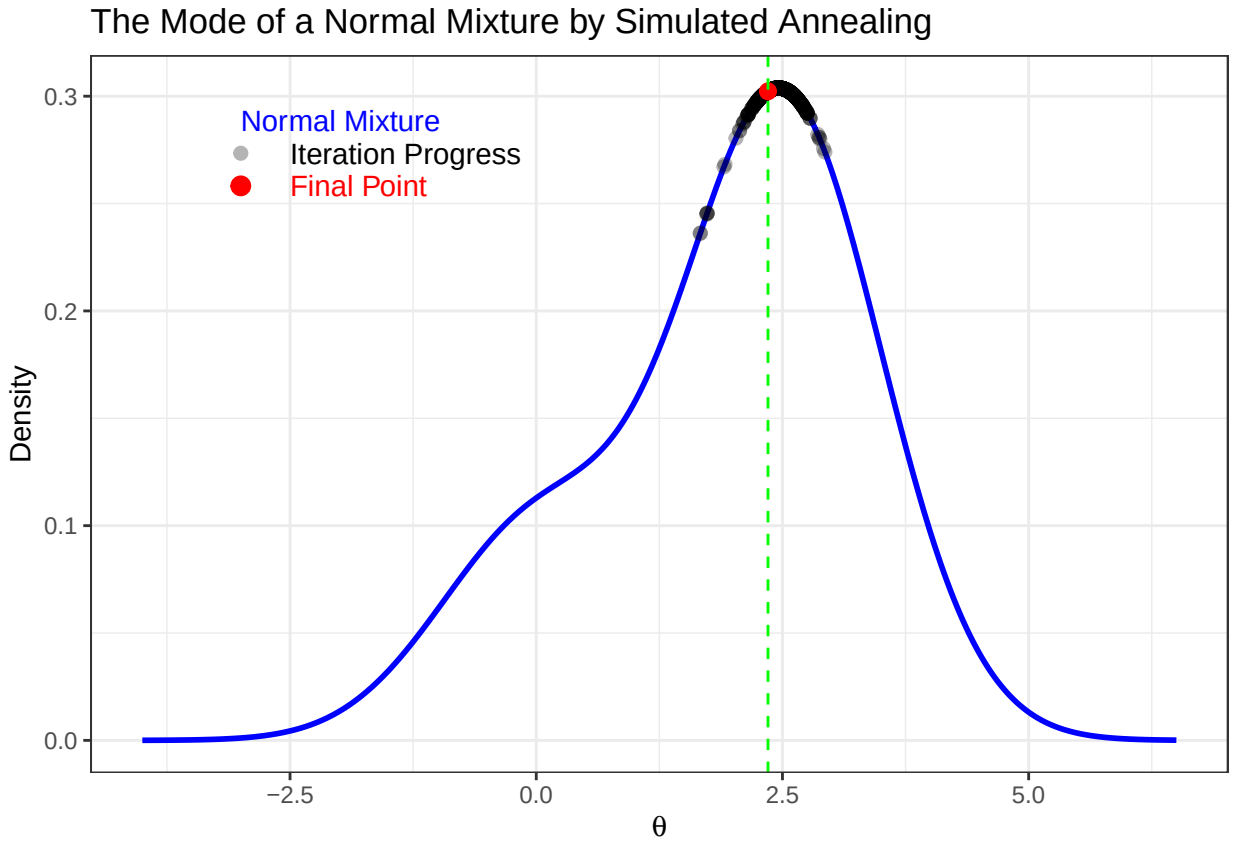
with $\mu_1 = 0$ and $\mu_2 = 2.5$. Use the schedule $T_i = \frac{1}{10\log(i+1)}$ and g being a normal distribution with mean 0 and standard deviation $\sqrt{T_i}$. Provide a plot to demonstrate how the iteration progresses.

Solution:

```
# 1. write an R program to implement the algorithm
# 2. produce a plot of the pdf of the mixture model
# 3. then impose the iteration history onto the pdf plot
# and mark the last iteration by a red dot (use the function point())


# Set seed
set.seed(232)

# Define mixture distribution parameters
mu1 <- 0
mu2 <- 2.5
sigma <- 1

# mixture density function
```

```r
mix_den_fun <- function(x) {
  (1/4) * dnorm(x, mean = mu1, sd = sigma) +
    (3/4) * dnorm(x, mean = mu2, sd = sigma)
}

# log-density function
h <- function(x) {
  log(mix_den_fun(x))
}

# Initial parameters
n_iter <- 1000        # Number of iterations
theta <- numeric(n_iter + 1)  # Store theta values
theta[1] <- 1                 # Starting point

# Perform Simulated Annealing
for (i in 1:n_iter) {
  # temperature schedule
  T_i <- 1 / (10 * log(i + 1))

  # Sample perturbation from N(0, sqrt(T_i))
  zeta <- rnorm(1, mean = 0, sd = sqrt(T_i))

  # Propose new theta
  theta_cand <- theta[i] + zeta

  # Compute change in log-density
  delta_h <- h(theta_cand) - h(theta[i])

  # Acceptance probability
  rho <- min(exp(delta_h / T_i), 1)

  # Decide whether to accept the new theta
  if (runif(1) < rho) {
    theta[i + 1] <- theta_cand
  } else {
    theta[i + 1] <- theta[i]
  }
}

# Remove the initial theta for plotting
theta_history <- theta[-1]

# data frame for Ploting the Mixture density
x_vals <- seq(mu1 - 4*sigma, mu2 + 4*sigma, length.out = 1000)

density_df <- data.frame(x = x_vals, y = mix_den_fun(x_vals))

# Create a data frame for the iteration history
history_df <- data.frame(theta = theta_history, iteration = 1:n_iter)
history_df$y <- mix_den_fun(history_df$theta)

# Extract the last iteration
```

```r
final_point <- history_df[n_iter, ]


ggplot() +
  # mixture density
  geom_line(data = density_df, aes(x = x, y = y), color = "blue", size = 1) +

  # Overlay the iteration history
  geom_point(data = history_df, aes(x = theta, y = y),
             color = "black", alpha = 0.3, size = 2) +

  # Mark the final iteration with a red dot
  geom_point(data = final_point, aes(x = theta, y = y),
             color = "red", size = 2.5) +
  geom_vline(xintercept = final_point$theta,
             linetype = "dashed", color = "green", size = 0.5)+

  ggtitle("The Mode of a Normal Mixture by Simulated Annealing") +
  xlab(expression(theta)) +
  ylab("Density") +

  # Density legend
  scale_color_manual(name = "Legend",
                     values = c("Normal Mixture" = "blue",
                                "Iteration Progress" = "black",
                                "Final Point" = "red")) +

  theme_bw() +

  # legend color
  guides(color = guide_legend(override.aes = list(
    linetype = c("solid", "blank", "blank"),
    shape = c(NA, 16, 16),
    size = c(1, 2, 2),
    alpha = c(1, 0.3, 1)
  ))) +

# legend
  annotate("text", x = mu1 - 3*sigma, y = max(density_df$y)*0.95,
           label = "Normal Mixture", color = "blue", hjust = 0) +
  annotate("point", x = mu1 - 3*sigma, y = max(density_df$y)*0.90,
           color = "black", alpha = 0.3, size = 2) +
  annotate("text", x = mu1 - 2.5*sigma, y = max(density_df$y)*0.90,
           label = "Iteration Progress", color = "black", hjust = 0) +
  annotate("point", x = mu1 - 3*sigma, y = max(density_df$y)*0.85,
           color = "red", size = 3) +
  annotate("text", x = mu1 - 2.5*sigma, y = max(density_df$y)*0.85,
           label = "Final Point", color = "red", hjust = 0)
```



Problem 2. Laplace approximation for marginal likelihood

Consider Bayesian estimation for $\mathbf{x} = (x_1, \ldots, x_n)$, a random sample from $f(x|\theta)$. Let the prior distribution of θ be $\pi(\theta)$. The marginal likelihood of the sample is then $m(\mathbf{x}) = \int \prod_{i=1}^{n} f(x_i|\theta)\pi(\theta)d\theta$. Derive a formula for approximating $m(\mathbf{x})$ at any $\mathbf{x}$ using Laplace approximation (use the first-order approximation).

Next, apply the approximation to the following scenario.

- $\theta \sim N(0, 3^2)$
- $x|\theta \sim N(\theta, 1)$

```
# sample data
x = c(1.2241, 0.3598, 0.4008, 0.1107, -0.5558, 1.7869, 0.4979, -1.9666, 0.7014, -0.4728)
# use the Laplace approximation to evaluate the marginal likelihood
# Your R code
```

Solution:

Deriving the Laplace Approximation for any x:

```
knitr::include_graphics("D:/WashU/First Year/Sem1/SDS5531_StatsComputing/Homework/HW3/Hw32a.pdf")
```

Problem 2

Deriving the formula to approximate $m(x)$ with Laplace approximation as;

Marginal likelihood:

$$M(x) = \int \prod_{i=1}^{n} f(x_i|\theta)\,\pi(\theta)\,d\theta$$

by using log-likelihood;

Let

$$\prod_{i=1}^{n} f(x_i|\theta) = e^{\sum_{i=1}^{n} \log f(x_i|\theta)}$$

$$\pi(\theta) = e^{\log \pi(\theta)}$$

So;

$$M(x) = \int e^{\sum_{i=1}^{n} \log f(x_i|\theta) + \log \pi(\theta)}\, d\theta$$

Now; Let

$$h(\theta) = \sum_{i=1}^{n} \log f(x_i|\theta) + \log \pi(\theta)$$

$$M(x) = \int e^{nh(\theta)}\, d(\theta)$$

find the maximizer of $h(\theta)$;

Figure 1: Problem 2, Deriving

```
knitr::include_graphics("D:/WashU/First Year/Sem1/SDS5531_StatsComputing/Homework/HW3/Hw32b.pdf")
```

$$\hat{\theta} = \arg\max_{\theta} h(\theta)$$

$h'(\theta) = 0$

By Taylor expansion of $h(\theta)$ around $\hat{\theta}$:

$$h(\theta) \approx h(\hat{\theta}) + h'(\hat{\theta})(\theta - \hat{\theta}) + h''(\theta)\frac{(\theta-\hat{\theta})^2}{2}$$

($h'(\hat{\theta}) \to 0$)

Since $\hat{\theta}$ is a maximizer,

$$h(\theta) \approx h(\hat{\theta}) + h''(\theta)\frac{(\theta-\hat{\theta})^2}{2}$$

$\therefore$

$$M(x) \approx \int e^{nh(\hat{\theta}) + \frac{1}{2}h''(\hat{\theta})(\theta-\hat{\theta})^2 n} d\theta$$

$$= e^{nh(\hat{\theta})} \int e^{\frac{nh''(\theta)(\theta-\hat{\theta})^2}{2}} d\theta$$

the integral is a Gaussian integral b/c $h''(\hat{\theta}) < 0$

$$\int_{-\infty}^{\infty} e^{\frac{n(\theta-\hat{\theta})^2 h''(\hat{\theta})}{2}} d\theta = \int_{-\infty}^{\infty} e^{-\frac{1}{2}(\theta-\hat{\theta})^2 \cdot h''(\hat{\theta})n} \cdot \frac{1}{\sqrt{2\pi/nh''(\hat{\theta})}}$$

$$= \sqrt{\frac{2\pi}{-nh''(\hat{\theta})}} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi \cdot -h''(\theta)n}} e^{\frac{-\frac{1}{2}(\theta-\hat{\theta})^2}{nh''(\theta)}} d\theta \to 1$$

pdf of $N(\hat{\theta}, -nh''(\theta))$

Figure 2: Problem 2, Deriving

```
knitr::include_graphics("D:/WashU/First Year/Sem1/SDS5531_StatsComputing/Homework/HW3/Hw32c.pdf")
```

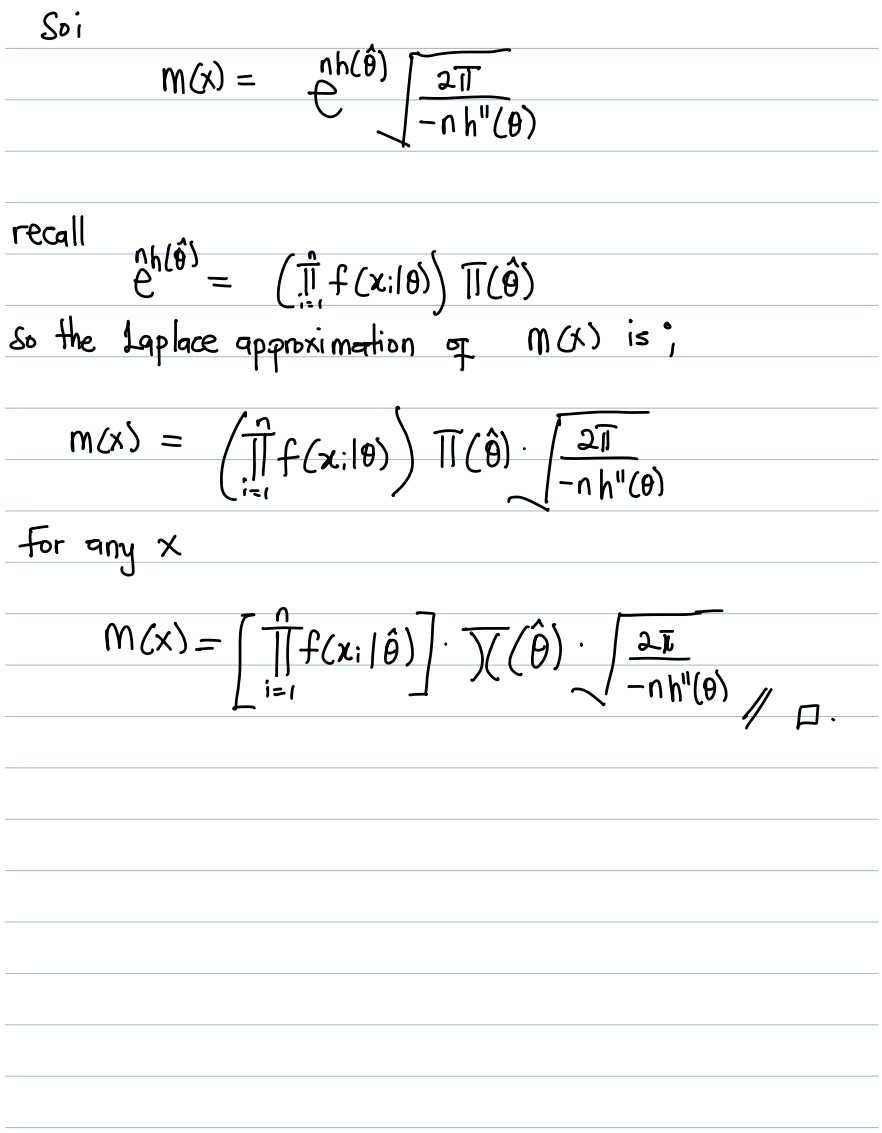


Figure 3: Problem 2, Deriving

Apply the approximation to the following scenario:

- $\theta \sim N(0, 3^2)$
- $x|\theta \sim N(\theta, 1)$

Recall that

$$f(\theta \mid x_i) = \frac{f(x \mid \theta)\pi(\theta)}{m(X)}$$

where $f(\theta \mid x_i) \propto f(x \mid \theta)\pi(\theta)$. Then this problem can look like this below;

```r
# use the Laplace approximation to evaluate the marginal likelihood
# Your R code

# Given data
x <- c(1.2241, 0.3598, 0.4008, 0.1107, -0.5558, 1.7869, 0.4979, -1.9666, 0.7014, -0.4728)
n <- length(x)          # Sample size
S <- sum(x)             # Sum of the data
mean_x <- mean(x)       # Sample mean

# Prior parameters
mu_0 <- 0               # Prior mean
tau_0_sq <- 9           # Prior variance (3^2)
sigma_sq <- 1           # Likelihood variance

# Compute posterior parameters
tau_n_sq_inv <- (1 / tau_0_sq) + n / sigma_sq  # Inverse of posterior variance
tau_n_sq <- 1 / tau_n_sq_inv                   # Posterior variance
mu_n <- tau_n_sq * ((mu_0 / tau_0_sq) + (n * mean_x / sigma_sq))  # Posterior mean

# Compute theta_hat (mode of the posterior)
theta_hat <- mu_n

# Compute the second derivative h''(theta_hat)
h_dd <- - (n / sigma_sq + 1 / tau_0_sq)

# Compute h(theta_hat) = log-likelihood + log-prior
log_likelihood <- sum(dnorm(x, mean = theta_hat, sd = sqrt(sigma_sq), log = TRUE))

log_prior <- dnorm(theta_hat, mean = mu_0, sd = sqrt(tau_0_sq), log = TRUE)

h_theta_hat <- log_likelihood + log_prior

# Compute the Gaussian integral term
gaussian_term <- sqrt(2 * pi / (-h_dd))

# Compute the marginal likelihood approximation
m_x <- exp(h_theta_hat) * gaussian_term

# Output the result
```

The marginal likelihood approximation $m(X) \approx 8.3671859 \times 10^{-8}$ represents the approximate probability of observing the data X under the model, integrating over all possible values of θ weighted by the prior $\pi(\theta)$.

In this special case, one can actually analytically solve the integral in $m(\mathbf{x})$. Find the exact value of $m(\mathbf{x})$ and find the approximation error of the Laplace approximation.

Analytical Solution:

```r
knitr::include_graphics("D:/WashU/First Year/Sem1/SDS5531_StatsComputing/Homework/HW3/Hw32d1.pdf")
```

Analytical Solution – Problem 2

Given

$$m(x) = \int \prod_{i=1}^{n} f(x_i|\theta)\, \pi(\theta)\, d\theta$$

likelihood: $f(x_i|\theta) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x_i-\theta)^2}{2}\right)$, $x_i|\theta \sim N(\theta, 1)$

Prior: $\pi(\theta) = \frac{1}{\sqrt{2\pi \cdot 3^2}} \exp\left(-\frac{\theta^2}{2(3^2)}\right)$, $\theta \sim N(0, 3^2)$

$$h(\theta) = \sum_{i=1}^{n} \log f(x_i|\theta) + \log \pi(\theta)$$

plug in $f(x_i|\theta)$ and $\pi(\theta)$

$$h(\theta) = \frac{-n}{2}\log(2\pi) - \frac{1}{2}\sum(x_i-\theta)^2 - \frac{1}{2}\log(2\pi \cdot 9) - \frac{\theta^2}{18}$$

to find the mode, set $h'(\theta) = 0$

$$h'(\theta) = \sum(x_i - \theta) - \frac{\theta}{9}$$

$$0 = \sum x_i - n\theta - \frac{\theta}{9}$$

$$\hat{\theta} = \frac{9\sum x_i}{9n+1} \quad //$$

Figure 4: Problem 2, Analytical

```
knitr::include_graphics("D:/WashU/First Year/Sem1/SDS5531_StatsComputing/Homework/HW3/Hw32d2.pdf")
```

$$h''(\theta) = -\left[\frac{1}{n}\left(\frac{9n+1}{9}\right)\right]$$

because prior $\sim N(0, 3^2)$, likelihood $\sim N(\theta, 1)$, we can deduce that the $m(x) \sim N(\mu_n, \sigma_n^2)$ where;

$$\mu_n = \frac{\frac{\mu_0}{\tau_0^2} + \frac{n\bar{x}}{\sigma^2}}{\frac{1}{\tau_0^2} + \frac{n}{\sigma^2}}$$

$\tau_0^2 \sim$ prior precision
$\mu_0 \sim$ Prior mean
$n =$ Sample size

$$\tilde{\tau}_n^2 = \left(\frac{1}{\tau_0^2} + \frac{n}{\sigma^2}\right)^{-1}$$

$$m(x) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n \exp\left(\frac{-\sum(x_i-\bar{x})^2}{2\sigma^2}\right)\sqrt{\frac{\tau^2}{\tau^2+n\sigma^2}}\exp\left(\frac{-n(\bar{x}-\mu_0)^2}{2(\tau^2+n\sigma^2)}\right)$$

after plugging in the values from $\Pi(\theta)$ and $f(x|\theta)$

$$m(x) = \exp\left[-\frac{1}{2}\sum(x_i-\hat{\theta})^2 - \frac{\theta^2}{18} + \log(2\pi)^{-n/2} + \log\frac{1}{\sqrt{18\pi}}\right]\cdot\sqrt{\frac{2\pi}{n+1/9}}$$

where $\bar{x} = \frac{\sum x_i}{n}$. □

Figure 5: Problem 2, Analytical

Implementation:

```
## Exact Solution

tau_sq <- 0.12
sigma_sq_n <- sigma_sq * n
V <- tau_sq + sigma_sq_n

first_term <-  sqrt(tau_sq / (tau_sq + sigma_sq_n))

exp_term1 <- exp(- n*(mean_x - mu_0)^2 / (2 * V))

second_term <- (1/sqrt(2 * pi * sigma_sq))^(n)

SSE <- sum((x - mean_x)^2)

exp_term2 <- exp(- SSE / (2 * sigma_sq))

exact_mx <- first_term * exp_term1 * second_term * exp_term2

## Laplace Approximation

## derived above


absolute_error = abs(m_x - exact_mx)
relative_error = absolute_error / exact_mx


# Output the results
 data_result <- data.frame("Exact_value" = exact_mx,
                           "Laplace_approx" = m_x,
                           "Absolute_error" =absolute_error,
                           "Relative_error"= relative_error)

kable(data_result, digits = 8, caption = "Analytical Solution Values")%>%
  kable_styling(position = "center",
              latex_options = "HOLD_position")
```

Table 1: Analytical Solution Values

Exact_value	Laplace_approx	Absolute_error	Relative_error
9e-08	8e-08	0	0.01874774

Problem 3. EM algorithm for normal mixtures

Let $\mathbf{x} = (x_1, \ldots, x_n)$ be a random sample from a two component normal mixture

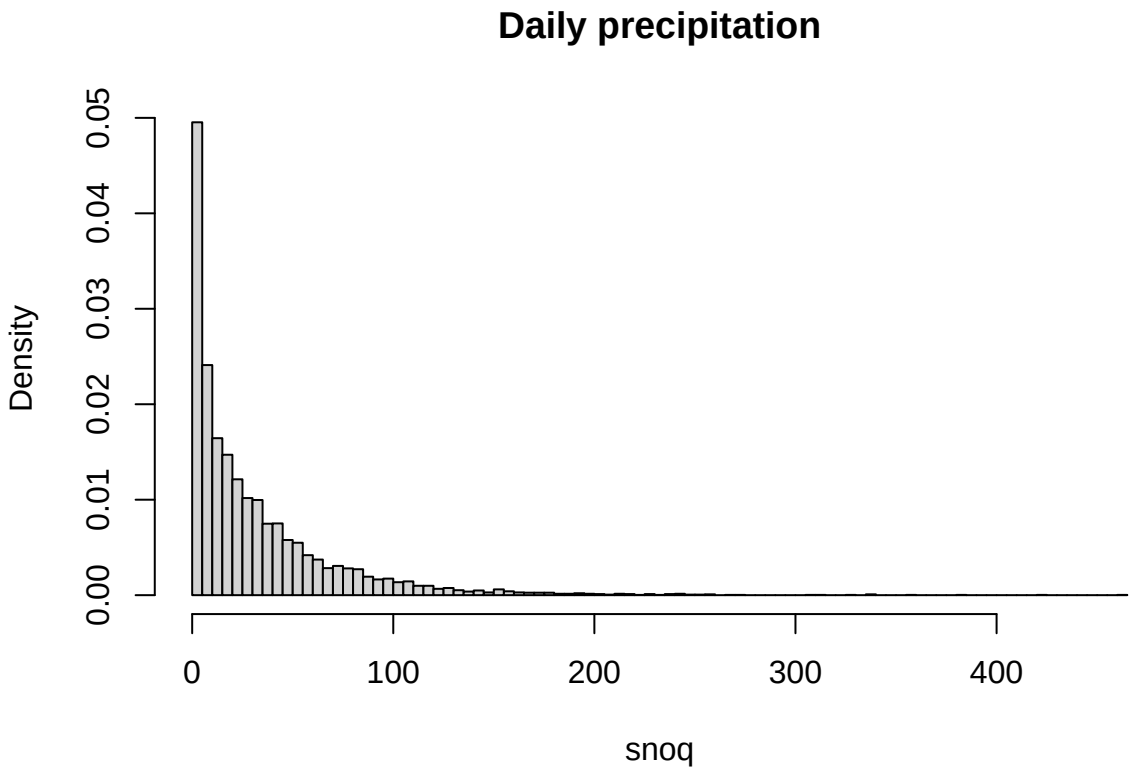
$$pN(\mu_1, \sigma_1^2) + (1-p)N(\mu_2, \sigma_2^2).$$

Derive the iterative updates in the EM algorithm for finding the MLE of $\theta = (p, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2)^T$.

Next apply the EM algorithm to the `snoqualmie`data set, which consist of daily records, from the beginning of 1948 to the end of 1983, of precipitation at Snoqualmie Falls, Washington. Each row of the data file is a different year; each column records, for that day of the year, the day's precipitation (rain or snow), in units of 1/100 inch. Because of leap-days, there are 366 columns, with the last column having an `NA` value for three out of four years.

Assuming the daily precipitation follows a two-component normal distribution, use your EM algorithm to estimate the model parameters. Set the stopping criterion as $||\theta^{(t+1)} - \theta^{(t)}||_2 < \epsilon$.

```
# read in the data
snoqualmie <- read.csv(url("https://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/data/snoqualmie.csv"),
  header=F)
# drop the missing values
snoqualmie.vec <- na.omit(unlist(snoqualmie))
# only keep the days that are wet
snoq <- snoqualmie.vec[snoqualmie.vec > 0]

# use your EM algorithm to estimate the model parameters
# Choose epsilon to be a small value, e.g. 0.00001.
# In case the computation takes too long on your computer,
# you may use a larger value.

# draw a histogram
hist(snoq,breaks=100,prob=T,main="Daily precipitation")
```




```
# impose your estimated density of the mixture model onto the histogram
# you may use the curve() function
```

Solution:

Derive the iterative updates in the EM algorithm for finding the MLE :

```
knitr::include_graphics("D:/WashU/First Year/Sem1/SDS5531_StatsComputing/Homework/HW3/Hw33a.pdf")
```

Problem 3

Deriving the EM algorithm for parameters

$$\theta = (p, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2)^T$$

Introduce a latent variable Z_i representing which component normal distribution is observed ;

$Z_i = 1$ if x_i comes from Component 1

$Z_i = 0$ if x_i comes from Component 2

$Z_i \sim \text{Bernulli}(Z_i)$

2. Complete data log-likelihood :

$$N(x, z ; \mu, \sigma^2)$$

$$\log L(\theta) = \sum_{i=1}^{n} [z_i (\log p + \log N(x_i ; \mu_1, \sigma_1^2)) + (1 - z_i)[\log(1-p) + \log N(x_i ; \mu_2, \sigma_2^2)]]$$

Figure 6: Problem 3, Deriving

```
knitr::include_graphics("D:/WashU/First Year/Sem1/SDS5531_StatsComputing/Homework/HW3/Hw33b.pdf")
```

E - Step:

$$Q(\theta|\theta^{(t)}) = E[z_i|x_i,\theta^{(t)}] = P[z=1|x_i,\theta^{(t)}]$$

Let

$$Q(\theta|\theta^{(t)}) = \gamma_i$$

$$\gamma_i = \frac{p^{(t)} N(x_i; \mu_1^{(t)}, \sigma_1^{2(t)})}{p^{(t)} N(x_i; \mu_1^{(t)}, \sigma_1^{2(t)}) + (1-p^{(t)}) N(x_i; \mu_2^{(t)}, \sigma_2^{2(t)})}$$

γ_i is the probability that datapoint x_i comes from component i.

M - Step:

Update for p:

$$\frac{\partial Q(\theta|\theta^{(t)})}{\partial p^{(t)}} = p^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n}\gamma_i$$

Update for μ_1 & σ_1^2:

$$\frac{\partial Q(\theta|\theta^{(t)})}{\partial \mu_1^{(t)}} = \mu_1^{(t+1)} = \frac{\sum_{i=1}^{n}\gamma_i x_i}{\sum_{i=1}^{n}\gamma_i}$$

$$\frac{\partial Q(\theta|\theta^{(t)})}{\partial \sigma_1^{2(t)}} = \sigma_1^{2(t+1)} = \frac{\sum_{i=1}^{n}\gamma_i (x_i - \mu_1^{(t+1)})^2}{\sum_{i=1}^{n}\gamma_i}$$

Figure 7: Problem 3, Deriving

```
knitr::include_graphics("D:/WashU/First Year/Sem1/SDS5531_StatsComputing/Homework/HW3/Hw33c.pdf")
```

Update for μ_2 & σ_2^2 :

$$\frac{\partial Q(\theta|\theta^{(t)})}{\partial \mu_2^{(t)}} = \mu_2^{(t+1)} = \frac{\sum_{i=1}^{n}(1-\gamma_i)x_i}{\sum_{i=1}^{n}(1-\gamma_i)}$$

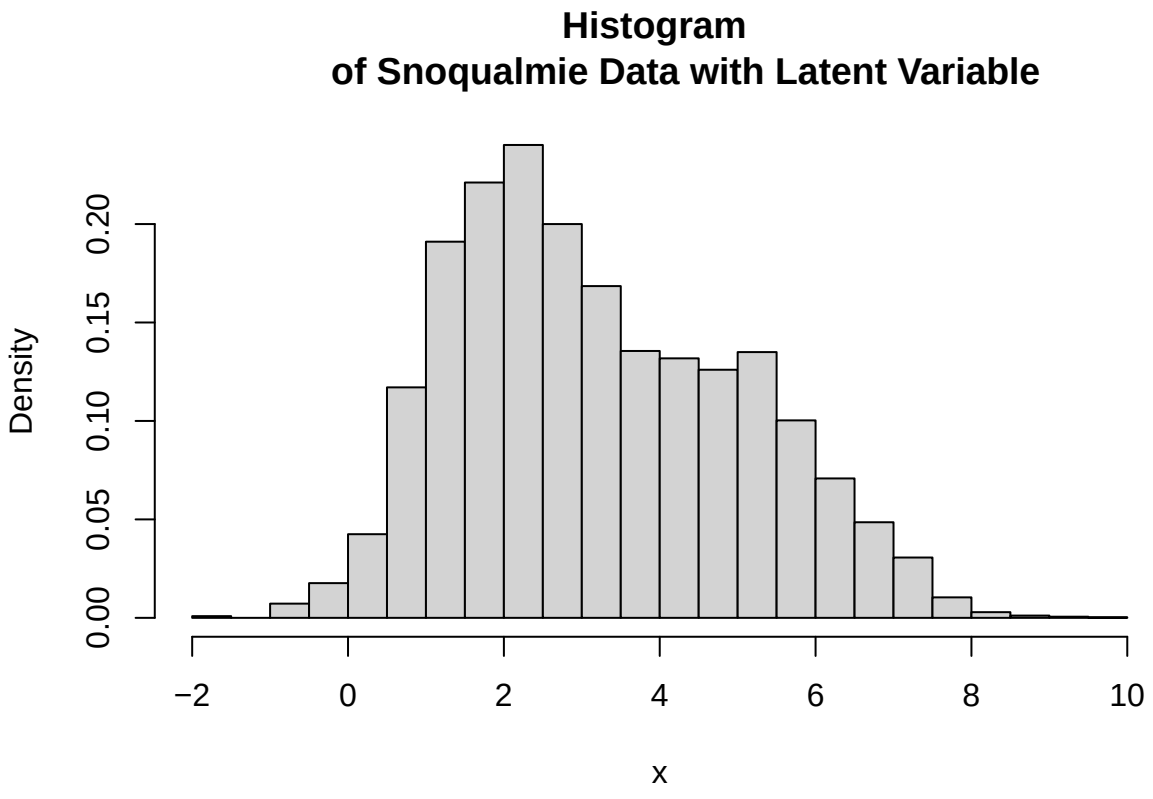
$$\frac{\partial Q(\theta|\theta^{(t)})}{\partial \sigma_2^{2(t)}} = \sigma_2^{2(t+1)} = \frac{\sum_{i=1}^{n}(1-\gamma_i)(x_i-\mu_2^{(t+1)})^2}{\sum_{i=1}^{n}(1-\gamma_i)}$$

repeat the E-Step and M-Step until Convergence

Figure 8: Problem 3, Deriving

Applying Algorithm:

```r
# Set seed for reproducibility
set.seed(778)

# Parameters for the true distributions
n <- 6920          # Total number of data points
p_true <- 0.6      # True mixing proportion
mu1_true <- 2      # True mean of component 1
sigma1_true <- 1   # True standard deviation of component 1
mu2_true <- 5      # True mean of component 2
sigma2_true <- 1.5 # True standard deviation of component 2

# Generate latent variable based on the true mixing proportion
z <- rbinom(n, size = 1, prob = p_true)


# Generate data from the two normal distributions given the latent var

snoq[z == 1] <- rnorm(sum(z == 1), mean = mu1_true, sd = sqrt(sigma1_true))
snoq[z == 0] <- rnorm(sum(z == 0), mean = mu2_true, sd = sqrt(sigma2_true))

# Plot the histogram of the data
hist(snoq , breaks = 30, probability = TRUE, main = "Histogram
     of Snoqualmie Data with Latent Variable",
     xlab = "x", ylab = "Density",
     col = "lightgray")
```



```r
# Set simple initial parameters
p <- 0.5
mu1 <- quantile(snoq , probs = 0.25)
mu2 <- quantile(snoq , probs = 0.75)
sigma1 <- var(snoq)
sigma2 <- var(snoq)
```

Implement the EM Algorithm:

```r
# Function to compute the normal density vectorized
dnorm_vectorized <- function(x, mean, sd) {
  dnorm(x, mean = mean, sd = sd)
}

# EM Algorithm parameters
max_iter <- 1000     # Maximum number of iterations
tol <- 1e-6          # Tolerance for convergence
log_likelihood <- numeric(max_iter)  # Store log-likelihoods

for (iter in 1:max_iter) {
  ### E-step
  # Compute responsibilities (gamma_i)
  tau1 <- p * dnorm_vectorized(snoq, mu1, sqrt(sigma1))        # Numerator for component 1
  tau2 <- (1 - p) * dnorm_vectorized(snoq, mu2, sqrt(sigma2))  # Numerator for component 2
  denom <- tau1 + tau2                                         # Denominator
  gamma <- tau1 / denom                                        # Responsibility for component 1

### M-step
  # Update parameters using the criteria
p_new <- mean(gamma)
mu1_new <- sum(gamma * snoq) / sum(gamma)
mu2_new <- sum((1 - gamma) * snoq) / sum(1 - gamma)
sigma1_new <- (sum(gamma * (snoq - mu1_new)^2) / sum(gamma))
sigma2_new <- (sum((1 - gamma) * (snoq - mu2_new)^2) / sum(1 - gamma))

  # Compute log-likelihood
  log_likelihood[iter] <- sum(log(denom))

  # Check for convergence with stopping criterion
  if (iter > 1 && abs(log_likelihood[iter] - log_likelihood[iter - 1]) < tol) {
    cat("Converged at iteration:", iter, "\n")
    break
  }

  # Update parameters for the next iteration
  p <- p_new
  mu1 <- mu1_new
  mu2 <- mu2_new
  sigma1 <- sigma1_new
  sigma2 <- sigma2_new
}
```

```
## Converged at iteration: 223
```

```
# Trim the log-likelihood vector
log_likelihood <- log_likelihood[1:iter]

# Output the estimated parameters
# Output result table
result_data <- data.frame("Mix_Prob" = p,
                          "Mu_1" = mu1,
                          "Var1" = sigma1,
                          "Mu_2" = mu2,
                          "Var2" = sigma2)


kable(result_data, digits = 4, caption = "Estimated Parameters")%>%
  kable_styling(position = "center",
              latex_options = "HOLD_position")
```

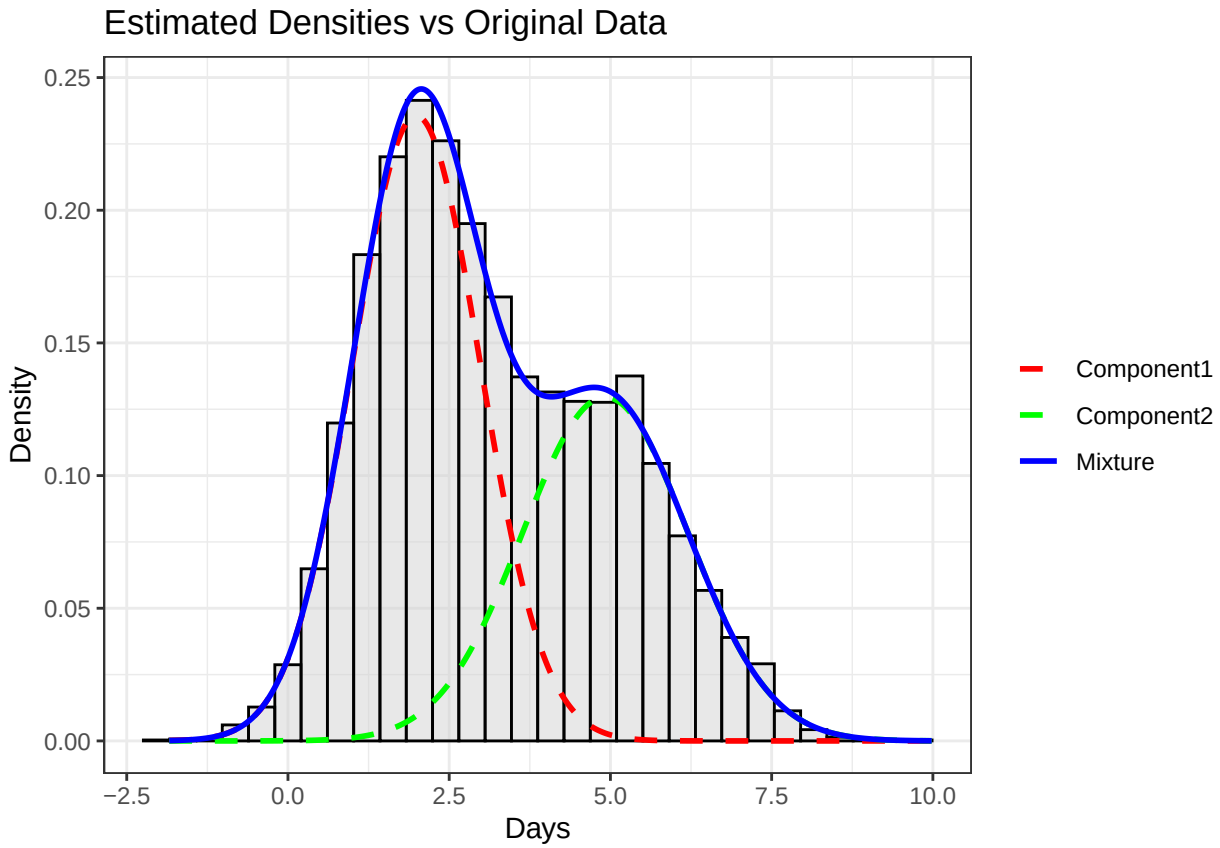
Table 2: Estimated Parameters

Mix_Prob	Mu_1	Var1	Mu_2	Var2
0.5839	1.9891	0.9778	4.92	1.6407

Plotting the problem

```
# Data frame for the histogram
data_plot <- data.frame(Snoq = snoq)

# Create a sequence of x values for plotting the densities
x_seq <- seq(min(snoq), max(snoq), length.out = 1000)

# Compute the estimated densities
estdensity_data <- data.frame(
  x = x_seq,
  Mixture = p * dnorm(x_seq, mean = mu1, sd = sqrt(sigma1)) +
    (1 - p) * dnorm(x_seq, mean = mu2, sd = sqrt(sigma2)),
  Component1 = p * dnorm(x_seq, mean = mu1, sd = sqrt(sigma1)),
  Component2 = (1 - p) * dnorm(x_seq, mean = mu2, sd = sqrt(sigma2))
)


# Reshape the data to long format
est_densitydata_long <- estdensity_data %>%
  pivot_longer(cols = c("Mixture", "Component1", "Component2"),
               names_to = "Density",
               values_to = "DensityValue")


# Create the plot
ggplot() +
```

```r
# Histogram of the data
geom_histogram(data = data_plot, aes(x = snoq, y = ..density..),
               bins = 30, fill = "lightgray",
               color = "black", alpha = 0.5) +
# Estimated densities
geom_line(data = est_densitydata_long ,
          aes(x = x, y = DensityValue,
              color = Density, linetype = Density),
          size = 1) +
# Manual adjustments for colors and line types
scale_color_manual(values = c("Mixture" = "blue",
                              "Component1" = "red",
                              "Component2" = "green")) +
scale_linetype_manual(values = c("Mixture" = "solid",
                                 "Component1" = "dashed", "Component2" = "dashed")) +
# Labels and theme
labs(title = "Estimated Densities vs Original Data",
     x = "Days", y = "Density") +
theme_bw() +
theme(legend.title = element_blank())
```



Convergence Plot:

```r
## convergence plot

# Data frame for log-likelihood
loglik_data <- data.frame(
  Iteration = 1:iter,
  LogLikelihood = log_likelihood
)

# Plot the log-likelihood convergence
ggplot(loglik_data, aes(x = Iteration, y = LogLikelihood)) +
  geom_line(color = "blue") +
  geom_point(color = "red") +
  labs(title = "Log-Likelihood Convergence", x = "Iteration", y = "Log-Likelihood") +
  theme_minimal()
```