Bayesian Lab2 Codes

Manu Jain & Varun Gurupurandar

2025-05-03

QUESTION 1 CODE

```
###### Bayesian Learning Lab 2 ######
############ Question 1 #############

# Libraries Used
library(MASS)
library(mvtnorm)

#### PART A)

# loading dataset
data <- read.csv("C:/Users/Dell/OneDrive - Linköpings universitet/732A73 Bayesian Learning/Labs/Lab2/te

data$time2 <- data$time^2
x <- as.matrix(cbind(1, data$time, data$time2))
y <- data$temp
n <- length(y)

# Given Prior Hyperparamters
# mu0 <- c(0, 100, -100)
#omega0 <- 0.01* diag(3)
mu0 <- c(20, 0, -20) # modified
omega0 <- diag(c(0.1, 1, 1)) # modified
nu0 <- 1
sigma02 <- 1

# Evaluate time points
time_seq <- seq(0, 1, length.out = 100)
X_plot <- cbind(1, time_seq, time_seq^2)

# Number of prior draws
S <- 50

# Simulating prior draws, beta and plot
beta_prior_draws <- matrix(0, S, 3)
sigma2_prior_draws <- (nu0*sigma02)/rchisq(S,df = nu0)

plot(data$time, data$temp, xlab = "Time", ylab = "Temp", ylim = c(-30,40),
     main = "Regression Curves from Prior Draws")
```

```r
for(s in 1:S){

  cov_beta <- sigma2_prior_draws[s] * solve(omega0)
  beta_prior <- rmvnorm(1, mu0, cov_beta)
  beta_prior_draws[s,] <- beta_prior

  y_pred <- X_plot %*% t(beta_prior)
  lines(time_seq, y_pred, col = rgb(0, 0, 1, alpha = 0.2))
}

## Observation -
# The given dataset is related to temperature in Linkoping city at different
# point of time. Since, the temperature varies seasonally so the prior might
# have \mu0 = c(-20, 0, 20) and moderate uncertainty that is, \omega0 =
# c(0.1, 1, 1). The regression curves look reasonable because they fall within
# the range of plausable temperature value of [-20, 20]

#### PART B)

xtx <- t(x) %*% x
xty <- t(x) %*% y

# Posterior Parameters
mu_n <- solve(xtx + omega0) %*% (xty + omega0 %*% mu0)
omega_n <- xtx + omega0
nu_n <- nu0 + n
sigma_n2 <- (nu0*sigma02 + sum(y^2) + t(mu0) %*% omega0 %*% mu0 - t(mu_n) %*%
               omega_n %*% mu_n)/ nu_n

# Simulate Posterior Samples
N <- 500
sigma2_post_draws <- (nu_n*sigma_n2)/rchisq(N,df = nu_n)
beta_post_draws <- matrix(0, N, 3)

for (i in 1:N){
  beta_post_draws[i,] <- rmvnorm(1, mu_n, sigma2_post_draws[i] * solve(omega_n))
}

# Plot Posterior Histogram
hist(beta_post_draws[,1], main = expression("Histogram of " * beta[0]),
     breaks = 30, col = "lightblue")
hist(beta_post_draws[,2], main = expression("Histogram of " * beta[1]),
     breaks = 30, col = "lightgreen")
hist(beta_post_draws[,3], main = expression("Histogram of " * beta[2]),
     breaks = 30, col = "lightpink")

# For each time, compute posterior predictive curve
f_post <- apply(beta_post_draws, 1, function(b) X_plot %*% b)

# Compute pointwise statistics
f_median <- apply(f_post, 1, median)
f_lower <- apply(f_post, 1, quantile, probs = 0.05)
f_upper <- apply(f_post, 1, quantile, probs = 0.95)
```

```r
# Plot
plot(data$time, data$temp, xlab = "Time", ylab = "Temp",
     main = "Regression Curves from Posterior Draws")
lines(time_seq, f_lower, col = "blue", lty = 2)
lines(time_seq, f_upper, col = "darkgreen", lty = 2)
lines(time_seq, f_median, col = "red", lty = 2)

## Observation-
# The posterior probability interval does not contain most of the data points.
# The actual data points also contain noise \epsilon ~ N(0, \sigma^2). So, to
# get the full dataset in the given range, variance(\sigma^2) is to be included
# in the prediction interval.

#### PART C)

# Computing time with lowest temperature -
# f(t) = \beta0 + \beta1*t + \beta2*t^2
# f'(t) = \beta1 + 2*\beta2*t
# putting f'(t) = 0 will give the minimum value for t. Because \beta2 > 0
# Thus,    t = - \beta1 / (2*\beta2)
x_min <- -beta_post_draws[,2] / 2* beta_post_draws[,3]

# Plot the posterior distribution
hist(x_min,  breaks = 30, col = "gray",
     main = "Posterior of time with lowest temp",
     xlab = "Time (x̃)" )


#### PART D)

# Build X matrix with polynomial terms up to degree 10
X10 <- sapply(0:10, function(i) data$time^i)
X10 <- as.matrix(X10)

# Prior mean and precision
mu0_10 <- rep(0, 11)
lambda <- 5
omega0_10 <- diag(lambda^(0:10))

# For prediction
# time_seq <- seq(0, 1, length.out = 100)
X_plot_10 <- sapply(0:10, function(j) time_seq^j)
X_plot_10 <- as.matrix(X_plot_10)

# Prior predictive draws
# S <- 50
beta_prior_draws_10 <- matrix(0, S, 10 + 1)
# sigma2_prior_draws <- (nu0 * sigma02) / rchisq(S, df = nu0)

# Plot prior predictive regression curves
plot(data$time, data$temp, xlab = "Time", ylab = "Temp",
     main = "Regression Curves from Prior Draws (Degree 10)")
```

```r
for(s in 1:S){

  cov_beta_10 <- sigma2_prior_draws[s] * solve(omega0_10)
  beta_prior_10 <- rmvnorm(1, mu0_10, cov_beta_10)
  beta_prior_draws_10[s,] <- beta_prior_10

  y_pred_10 <- X_plot_10 %*% t(beta_prior_10)
  lines(time_seq, y_pred_10, col = rgb(0, 0, 1, alpha = 0.2))
}

## Observation -
# The suitable prior mean should be \mu0 = 0 as it expresses no strong belief
# in any specific shape for the curve. The suitable prior precision could be a
# diagonal matrix with increasing values for higher degree as higher degree
# terms would be heavily shrunk towards zero preventing overfitting unless data
# strongly supports them. Thus, $ \mu0 = rep(0, 11) $ and
# $ \omega0 = diag(\lambda^(0:10)) where \lambda = 5 $
```

QUESTION 2 CODE

```r
set.seed(12345)
standardize <- function(x, mean_val = NULL, sd_val = NULL) {
  if (is.null(mean_val)) mean_val <- mean(x, na.rm = TRUE)
  if (is.null(sd_val)) sd_val <- sd(x, na.rm = TRUE)

  (x - mean_val) / sd_val
}



# Prepare the response variable (using 'class_of_diagnosis' as outcome)
y <- disease_data$class_of_diagnosis  # Assuming 1 = has disease, 0 = healthy
# Calculate standardization parameters (must do this FIRST)
age_mean <- mean(disease_data$age)
age_sd <- sd(disease_data$age)
symptoms_mean <- mean(disease_data$duration_of_symptoms)
symptoms_sd <- sd(disease_data$duration_of_symptoms)
whiteblood_mean <- mean(disease_data$white_blood)
whiteblood_sd <- sd(disease_data$white_blood)

set.seed(12345)
means <- list(
  age = age_mean,
  duration = symptoms_mean,
  white_blood = whiteblood_mean
)
set.seed(12345)
sds <- list(
  age = age_sd,
  duration = symptoms_sd,
  white_blood = whiteblood_sd
```

```r
)
# Prepare predictors with standardization where needed
X <- cbind(# C binder means column-wise into a matrix or data frame.
  Intercept = 1,
  Age = (disease_data$age - age_mean)/age_sd,
  Gender = disease_data$gender,
  Symptoms = (disease_data$duration_of_symptoms - symptoms_mean)/symptoms_sd,
  Breathing = disease_data$dyspnoea,
  WhiteCells = (disease_data$white_blood - whiteblood_mean)/whiteblood_sd
)

# Convert to X is stored as a numeric matrix
X <- as.matrix(X)

#(A)Calculate both beta tilda and J(beta_tilda) by using the optim function in
# R.Use the prior beta~ N(0, tau^2I), where tau=2.

# Define log-posterior function with N(0, tau²I) prior (tau=2)
log_posterior <- function(beta, X, y) {
  eta <- X %*% beta #here it computes linear predictor
  prob <- plogis(eta)  # Logistic function or equal to 1/(1+exp(eta)) which is same as logistic express
  log_lik <- sum(y * log(prob) + (1-y) * log(1-prob)) # log-likelihood is been applies on the above log
  log_prior <- sum(dnorm(beta, mean=0, sd=2, log=TRUE))  # Prior N(0,4)# here we calculate log-prior fo
  log_lik + log_prior
}
set.seed(12345)
# Find posterior mode (beta_tilde) and Hessian (J)
optim_result <- optim(#optim function here is general optimization function
  par = rep(0, ncol(X)),#It is the starting point for optimization,,it createsa vector of zeros,in logi
  fn = log_posterior,# this is the function that needs to be maximized
  X = X, y = y,
  method = "BFGS",
  control = list(fnscale=-1),  # Maximize
  hessian = TRUE  # it calculates the Hessian matrix(ie the second derivative )at the solution
)# why second derivative , the first derivate tells the direction of the
#gradient steep increase
#the second derivative or the hessian tells about the curvature of the slope
#(ie how fast the slope is changing)

# Extract results
beta_tilde <- optim_result$par  # Posterior mode
J_beta_tilde <- -optim_result$hessian  # Negative Hessian

# (B) Present the numerical values of beta tilda and J^-1y(beta tilda )for the
# Disease data.
# Name the coefficients
names(beta_tilde) <- colnames(X)
rownames(J_beta_tilde) <- colnames(X)
colnames(J_beta_tilde) <- colnames(X)

# Print results
cat("(B) Posterior Mode (beta_tilde):\n")
print(beta_tilde)
```

```r
cat("\n(B) Inverse Negative Hessian (J^-1(beta_tilde)):\n")
print(solve(J_beta_tilde))  # Posterior covariance matrix

post_cov <- solve(J_beta_tilde)
# (C) Compute an approximate 95% equal tail posterior probability interval for
# the regression coeffcient to the variable Age.

# Get Age coefficient index
age_index <- which(names(beta_tilde) == "Age")#IT IS MENTIONS IN THE QUESTION that we find the postirio

# Calculate posterior standard deviation for Age
age_sd <- sqrt(solve(J_beta_tilde)[age_index, age_index])

# 95% equal-tailed credible interval
age_ci <- beta_tilde[age_index] + c(-1.96, 1.96) * age_sd

cat("\n(C) 95% Posterior Interval for Age coefficient:\n")
cat(sprintf("[%.4f, %.4f]\n", age_ci[1], age_ci[2]))

# Interpretation
if (prod(age_ci) > 0) {  # Both limits have same sign
  cat("Age is statistically significant (95% CI doesn't contain 0)\n")
  if (age_ci[1] > 0) {
    cat("Older age increases disease probability\n")
  } else {
    cat("Older age decreases disease probability\n")
  }
} else {
  cat("Age is not statistically significant (95% CI contains 0)\n")
}
#(D) verify that estimation results reasonable by comparing the posterior means #tomaximum likelihood e

# Optional: Compare with frequentist approach
cat("\nComparison with standard glm() results:\n")
glm_fit <- glm(y ~ Age + Gender + Symptoms + Breathing + WhiteCells,
               data = as.data.frame(X[,-1]),  # Remove intercept
               family = binomial)
print(summary(glm_fit))
################################################################################
#2ND PART
set.seed(12345)
# 1. Define the simulation function (improved version)
simulate_predictive <- function(age, gender, symptoms, breathing, whiteblood,
                                n_sim = 10000) {
  # Standardize the continuous variables using the same scaling as before
  age_std <- (age - age_mean) / age_sd
  symptoms_std <- (symptoms - symptoms_mean) / symptoms_sd
  whiteblood_std <- (whiteblood - whiteblood_mean) / whiteblood_sd

  # Create the feature vector for this patient
  x_patient <- c(1, age_std, gender, symptoms_std, breathing, whiteblood_std)

  # Simulate beta draws from posterior normal approximation
```

```r
  beta_draws <- rmvnorm(n_sim, mean = beta_tilde, sigma = post_cov)

  # Calculate Pr(y=1|x) for each draw
  prob_draws <- plogis(beta_draws %*% x_patient)

  return(prob_draws)
}

# 2. Simulate for our specific patient:
# 38-year-old woman (gender=1), 10 days symptoms, no labored breathing (0),
# 11000 white blood
library(ggplot2)
# 1. Define core functions
sigmoid <- function(z) 1 / (1 + exp(-z))
set.seed(12345)
predict_prob <- function(beta_mean, beta_cov, new_x, n_samples = 1000) {
  beta_samples <- rmvnorm(n_samples, beta_mean, beta_cov)
  apply(beta_samples, 1, function(beta) sigmoid(sum(new_x * beta)))
}

new_patient <- c(
  1,  # Intercept
  standardize(38, means$age, sds$age),  # Age
  standardize(10, means$duration, sds$duration),  # Symptoms
  standardize(11000, means$white_blood, sds$white_blood),  # WhiteBlood
  1,  # Gender (1 = female)
  0   # Dyspnoea (0 = no)
)
set.seed(12345)
probs <- predict_prob(beta_tilde, post_cov, new_patient)

# 5. Visualization (clean and simple)
par(mfrow = c(1, 2))

# Density plot
plot(density(probs), main = "Disease Probability Distribution",
     xlab = "P(Disease = 1)", col = "blue", lwd = 2)
abline(v = mean(probs), col = "red", lty = 2)
legend("topright", legend = c("Density", "Mean"),
       col = c("blue", "red"), lty = c(1, 2), cex = 0.5)

# Histogram
hist(probs, breaks = 30, col = "skyblue", border = "white",
     main = "Posterior Predictive Distribution",
     xlab = "Probability of Disease")

par(mfrow = c(1, 1))

# 6. Print results
cat("\nPrediction Summary for 38yo Female Patient:\n")
cat(sprintf("Mean probability: %.3f\n", mean(probs)))
cat(sprintf("95%% Credible Interval: [%.3f, %.3f]\n",
            quantile(probs, 0.025), quantile(probs, 0.975)))
```