hw3

Jiahao Tian

2023-02-27

Question 1

a) Reversible Jumps MCMC algorithm

$$
\begin{aligned}
p(y|\beta,\sigma^2,\gamma)p(\beta|\sigma^2,\gamma) &= N(y; X_\gamma\beta, \sigma^2 I)N(\beta; \frac{\sigma^2}{g}(X_\gamma^\top X_\gamma)^{-1}) \\
&= (2\pi\sigma^2)^{-n/2}\exp\big(-\frac{1}{2\sigma^2}(y-X_\gamma\beta)^\top(y-X_\gamma\beta)\big)\exp\big(-\frac{g}{2\sigma^2}\beta^\top X_\gamma^\top X_\gamma\beta\big)
\end{aligned}
$$

$$
\begin{aligned}
&-\frac{1}{2\sigma^2}(y-X_\gamma\beta)^\top(y-X_\gamma\beta)-\frac{g}{2\sigma^2}\beta^\top X_\gamma^\top X_\gamma\beta \\
&= -\frac{1}{2}\big[y^\top y/\sigma^2 - 2\beta^\top Xy/\sigma^2 + \beta^\top X_\gamma^\top X_\gamma\beta/\sigma^2 + g\beta^\top X_\gamma^\top X_\gamma\beta/\sigma^2\big] \\
&= -\frac{1}{2}\big[y^\top y/\sigma^2 + (\beta-m)M^{-1}(\beta-m) + m^\top M^{-1}m\big]
\end{aligned}
$$

$$
where\ M = \sigma^2(X_\gamma^\top X_\gamma)^{-1}/(1+g)\ and\ m = MX^\top y/\sigma^2
$$

$$
So\ p(y|\beta,\sigma^2,\gamma)p(\beta|\sigma^2,\gamma) = (2\pi\sigma^2)^{-n/2}\exp\big(-\frac{1}{2}[y^\top y/\sigma^2 + (\beta-m)M^{-1}(\beta-m) + mM^{-1}m]\big)
$$

- Now need to integrate it out.

$$
\begin{aligned}
\int p(y|\beta,\sigma^2,\gamma)p(\beta|\sigma^2,\gamma)d\beta &= (2\pi\sigma^2)^{-n/2}\exp(\frac{1}{2\sigma^2}y^\top y)\exp(\frac{1}{2}m^\top M^{-1}m)|2\pi M|^{1/2} \\
&= (2\pi)^{(-n+q_\gamma)/2}(\sigma^2)^{(-n+q_\gamma)/2}|(X_\gamma^\top X_\gamma)|^{-1/2}(\frac{1}{g+1})^{q_\gamma/2}\exp(-\frac{1}{2\sigma^2}y^\top y)\exp(\frac{1}{2}m^\top M^{-1}m)
\end{aligned}
$$

$$
\begin{aligned}
\exp(-\frac{1}{2\sigma^2}y^\top y)\exp(\frac{1}{2}m^\top M^{-1}m) &= \exp\big(-\frac{1}{2\sigma^2}(y^\top y - y^\top X_\gamma(X_\gamma^\top X_\gamma)^{-1}X_\gamma^\top y/(1+g)\big) \\
&= \exp\big(-\frac{1}{2\sigma^2}y^\top(I-\frac{1}{1+g}X_\gamma(X_\gamma^\top X_\gamma)^{-1}X_\gamma^\top)y\big) \\
&= \exp\big(-\frac{1}{2\sigma^2}RSS\big)
\end{aligned}
$$

$$
\begin{aligned}
&= (2\pi)^{-(n+q_\gamma)/2}(\frac{1}{g+1})^{q_\gamma/2}|(X_\gamma^\top X_\gamma)|^{-1/2}\int(\sigma^2)^{(-n+q_\gamma)/2}\exp(-\frac{1}{2\sigma^2}RSS)p(\sigma^2)d\sigma^2 \\
&= (2\pi)^{(-n+q_\gamma)/2}(\frac{1}{g+1})^{q_\gamma/2}|(X_\gamma^\top X_\gamma)|^{-1/2}\frac{\Gamma(\frac{n}{2}-1)}{(RSS/2)^{n/2}} \\
which\ follows &\sim \Gamma^{-1}(n/2-1, RSS/2)
\end{aligned}
$$

- sample from $p(\gamma|Y)$.

$$
\begin{aligned}
p(\gamma|y) &\propto p(y|\gamma)p(\gamma) \\
&= (2\pi)^{-(n+q_\gamma)/2}(\frac{1}{g+1})^{q_\gamma/2}|(X_\gamma^\top X_\gamma)|^{-1/2}\frac{\Gamma(\frac{n}{2}-1)}{(RSS/2)^{n/2}}p(\gamma)
\end{aligned}
$$

$$
\begin{aligned}
p(\sigma^2,\beta|y,\gamma) &\propto p(y|\gamma,\beta_\gamma,\sigma^2)p(\beta_\gamma|,\sigma^2,\gamma)p(\sigma^2|\gamma) \\
&= N(m,M)\times IG(n/2-1, RSS/2)
\end{aligned}
$$

$$
\begin{aligned}
p(\beta_\gamma,\gamma,\sigma^2|y) &\propto p(y|\beta_\gamma,\sigma^2,\gamma)p(\beta_\gamma,\sigma^2,\gamma) \\
&= p(y|\beta_\gamma,\sigma^2,\gamma)p(\beta_\gamma,\sigma^2|\gamma)p(\gamma) \\
&= N(m,M)\times IG(n/2-1, RSS/2)\times 2^{-q_\gamma}
\end{aligned}
$$

```r
# Define the function to calculate the posterior probability
posterior_prob = function(x, y, gamma, alpha, beta, sigma2_0, tau2_0) {
  n = length(y)
  p_gamma = length(gamma)
  if (p_gamma == 0) {
    return(0)
  }
  else {
    x_gamma = x[, gamma, drop = FALSE]
    tau2 = tau2_0[gamma]
    sigma2 = sigma2_0
    beta_gamma = MASS::ginv(x_gamma) %*% y
    log_lik = -0.5 * (n * log(sigma2)
                      + sum((y - x_gamma %*% beta_gamma)^2) / sigma2)
    log_prior = sum(dnorm(beta_gamma, mean = 0, sd = sqrt(tau2), log = TRUE)) +
                sum(dgamma(tau2, shape = alpha, rate = beta, log = TRUE))
```

```
    return(log_lik + log_prior)
  }
}
```

```
inclusion_prob = function(x, y, n_iter, burn, alpha, beta, sigma2_0, tau2_0) {

  # Initialize variables
  n = ncol(x)
  gamma = numeric(n)
  gamma_old = gamma
  posterior_prob_old = -Inf
  posterior_prob_vec = numeric(n_iter - burn)
  gamma_mat = matrix(0, ncol = n, nrow = n_iter - burn)

  # Run the MCMC algorithm
  for (i in 1:n_iter) {
    # Propose a move
    if (runif(1) < 0.5) {
      # Add a variable
      gamma_prop = sample(n, sum(gamma == 0), replace = FALSE)
      gamma_prop = sort(c(gamma[gamma != 0], gamma_prop))
      }
    else {
      # Delete a variable (only if gamma is not empty)
      if (sum(gamma != 0) > 0) {
        gamma_prop = gamma[-sample(which(gamma != 0), 1)]
        }
      else {
        gamma_prop = gamma
      }
      }

    # Calculate acceptance probability
    posterior_prob_prop = posterior_prob(x, y, gamma_prop, alpha, beta, sigma2_0, tau2_0)
    alpha_prop = min(1, exp(posterior_prob_prop - posterior_prob_old))
    # Accept or reject move
    if (runif(1) < alpha_prop) {
      gamma = gamma_prop
      posterior_prob_old = posterior_prob_prop
      }

    # Save current state if past burn-in period
    if (i > burn) {
      gamma_mat[i - burn, gamma] = 1
      posterior_prob_vec[i - burn] = posterior_prob_old
    }
    }
  # Calculate inclusion probabilities
  inclusion_probs = colMeans(gamma_mat)
  }
```

```
set.seed(888)
data(diabetes)
```

```
x = cbind(1, diabetes$x)
x = as.matrix(x)
y = diabetes$y

# Define prior parameters
alpha = 0.5
beta = 0.5
sigma2_0 = 100
tau2_0 = rep(10, ncol(x))

# Set number of iterations and burn
n_iter = 10000
burn = 5000

ip = inclusion_prob(x, y, n_iter, burn, alpha, beta, sigma2_0, tau2_0)
print(ip)
```

```
##  [1] 0.8236 0.0000 0.0000 0.8166 0.8224 0.0000 0.8192 0.8220 0.8188 0.8218
## [11] 0.8232
```

- I tried my best…

b)

- Compare results in (a) with results obtained sampling directly from $p(\gamma|Y)$. Implement this without the need for RJ Monte Carlo methods.

```
# Define the function
p_x = function(y, x, g = 0.001) {

  # Initialize variables
  n = nrow(x)
  p = ncol(x)
  if(is.null(p)) {
    X = as.matrix(x)
    n = dim(x)[1]
    p = dim(x)[2]
    }
  H = 0
  if(p > 1) {
    H = 1 / (g + 1) * x %*% solve(t(x) %*% x) %*% t(x)
    }
  RSS = t(y) %*% (diag(n) - H) %*% y
  return((-n + p) / 2 * log(2 * pi) -
            p / 2 * log(g + 1) - 1 / 2 * log(det(t(x) %*% x)) -
            lgamma(n / 2 - 1) - n / 2 * log(RSS / 2))
}
```

```
inclusion_prob2 = function(x, y, n_iter) {

  # Initialize variables
```

```r
  p = ncol(x)
  z = rep(1, p)
  p_proposed = p_x(y, x[, z == 1])
  p_current = p_proposed
  Z = matrix(0, n_iter, ncol(x))

  for(i in 1:n_iter) {
    for(j in sample(2:p)) {
      z.propose = z
      z.propose[j] = 1 - z.propose[j]

      # log-likelihood of proposed
      p_proposed = p_x(y, x[, z.propose == 1])

      r = (p_proposed - p_current) * (-1)^(z.propose[j] == 0)

      # sample from bernoulli
      z[j] = rbernoulli(1, 1 / (1 + exp(-r)))

      if(z[j] == z.propose[j]) {
        p_current = p_proposed
      }
      }
    Z[i, ] = z
    }
  inclusion_probs2 = colMeans(Z[1:(1 * n_iter), ])
}
```

```r
set.seed(999)
y = diabetes$y
x = cbind(1, diabetes$x)
x = as.matrix(x)

ip2 = inclusion_prob2(x = x, y = y, n_iter = 5000)
```

```
## Warning: `rbernoulli()` was deprecated in purrr 1.0.0.
```

```r
print(ip2)
```

```
##  [1] 1.0000 0.7324 0.9998 1.0000 1.0000 0.9884 0.9768 0.9104 0.9248 1.0000
## [11] 0.8374
```