ST509 Computational Statistics

Lecture 12: Gibbs Samplers

Seung Jun Shin

Department of Statistics
Korea University

E-mail: sjshin@korea.ac.kr

Gibbs Sampler I

- We extend the idea of MCMC by breaking the large complex problems into a sequence of simpler ones.
- Gibbs sampler may take a long time to converge, but is an interesting candidate.

Gibbs Sampler II

- Suppose we like to sample $(X, Y) \sim f(x, y)$, the two-stage Gibbs sampler generates a Markov chain (X_t, Y_t) whose stationary distribution is $f(x, y)$ as follows.
  1. Take $X_0 = x_0$
  2. For $t = 1, 2, \cdots$, generate
     - 2.1 $Y_t \sim f_{Y|X}(\cdot \mid x_{t-1})$
     - 2.2 $X_t \sim f_{X|Y}(\cdot \mid y_t)$
- Straightforward to implement as long as simulating from both conditionals are feasible.

Gibbs Sampler III

ex1 Consider

$$(X, Y) \sim N_2\left(0, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}\right)$$

- Given x_t, generate

$$Y_{t+1} \mid x_t \sim N(\rho x_t, 1-\rho^2)$$
$$X_{t+1} \mid y_{t+1} \sim N(\rho y_{t+1}, 1-\rho^2)$$

- Please implement the Gibbs sampler to generate the samples.

Gibbs Sampler IV

ex2 Consider the pair of distributions

$$X \mid \theta \sim \text{Binomial}(n, \theta), \qquad \theta \sim \text{Beta}(a, b)$$

which leads to the joint distribution:

$$f(x, \theta) = \binom{n}{x} \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \theta^{x+a-1} (1-\theta)^{n-x+b-1}$$

- Notice that the distribution of $X \mid \theta$ is given and

$$\theta \mid x \sim \text{Beta}(x+a, n-x+b)$$

- Please implement the Gibbs sampler to generate the samples.

Gibbs Sampler V

- Extension to multi-stage Gibbs sampler is straightforward.
  1. At iteration $t = 1, 2, \cdots$, given $\mathbf{x}^{(t)} = (x_1^{(t)}, \cdots, x_p^{(t)})$ generate
     1.1 $X_1^{(t+1)} \sim f_1(x_1 \mid x_2^{(t)}, \cdots, x_p^{(t)})$
     1.2 $X_2^{(t+1)} \sim f_2(x_2 \mid x_1^{(t+1)}, x_3^{(t)}, \cdots, x_p^{(t)})$
     $\vdots$
     1.3 $X_p^{(t+1)} \sim f_p(x_p \mid y_t) x_1^{(t+1)}, \cdots, x_{p-1}^{(t+1)})$
- We call $f_1, \cdots, f_p$ the full conditionals, the only densities used for simulation.

Gibbs Sampler VI

- Bayesian Linear Model:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \qquad N(0, \sigma^2)$$

- We consider a conjugate priors:

$$\beta_j \sim N(\mu_j, \tau_j^2), \quad j = 0, 1 \qquad \text{and} \qquad \sigma^2 \sim 1/\text{Gamma}(a, b)$$

- Gibbs sampler can be used to generate sample from

$$f(\beta_0, \beta_1, \sigma^2 \mid \mathbf{y}) \propto \pi_1(\beta_0)\pi_1(\beta_1)\pi_2(\sigma^2) f(y \mid \beta_0, \beta_1, \sigma^2)$$

Gibbs Sampler VII

- It is not difficult to show that the full conditionals are given by

$$\beta_0 \mid \beta_1, \sigma^2, \mathbf{y} \sim N\left(\frac{\sum_{i=1}^n (y_i - \beta_1 x_i)^2/\sigma^2 + \mu_0/\tau_0^2}{n/\sigma^2 + 1/\tau_0^2}, \left(n/\sigma^2 + 1/\tau_0^2\right)^{-1}\right)$$

$$\beta_1 \mid \beta_0, \sigma^2, \mathbf{y} \sim N\left(\frac{\sum_{i=1}^n x_i(y_i - \beta_0)/\sigma^2 + \mu_1/\tau_1^2}{\sum_{i=1}^n x_i^2/\sigma^2 + 1/\tau_1^2}, \left(\sum_{i=1}^n x_i^2/\sigma^2 + 1/\tau_1^2\right)^{-1}\right)$$

$$\sigma^2 \mid \beta_0, \beta_1, \mathbf{y} \sim 1/\text{Gamma}\left(\frac{n}{2} + a, \frac{1}{2}\sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2 + b\right)$$

Gibbs Sampler VIII

```
# Gibs_sampler
for(i in 1:n.samples){
  # update sigma2:
  SSE <- sum((y - beta[1] - x * beta[2])^2)
  sigma2 <- 1/rgamma(1, n/2 + a, SSE/2 + b)

  # update beta1:
  v <- n/sigma2 + 1/tau[1]^2
  m <- sum(y - x * beta[2])/sigma2 + mu[1]/tau[1]^2
  beta[1] <- rnorm(1, m/v, 1/sqrt(v))

  # update beta2:
  v <- sum(x^2)/sigma2 + 1/tau[2]^2
  m <- sum(x*(y-beta[1]))/sigma2 + mu[2]/tau[2]^2
  beta[2] <- rnorm(1,m/v,1/sqrt(v))

  samples[i,] <- c(beta, sigma2)
}
```

MH Algorithm within Gibbs Sampler I

- Full conditionals are not always available
- One can use MH algorithm to get a sample from a targeted conditional density.

MH Algorithm within Gibbs Sampler II

- Bayesian Logistic Regression

$$y_i \mid \mathbf{x}_i \sim \text{Bern}(p(\mathbf{x}_i; \boldsymbol{\beta}_0))$$

where $\mathbf{x}_i = (1, x_{i1}, \cdots, x_{ip})^T$ and $\boldsymbol{\beta} = (\beta_0, \beta_1, \cdots, \beta_p)^T$.

- Independent normal priors on $\boldsymbol{\beta} = (\beta_0, \beta_1, \cdots, \beta_p)^T$:

$$\beta_j \sim N(\mu_j, \tau_j^2), \qquad j = 0, 1, \cdots, p$$

- The full conditional density of β_j given all others is

$$f(\beta_j \mid \boldsymbol{\beta}_{-j}, \mathbf{y}) \propto \underbrace{\prod_{i=1}^{n} \{p(\mathbf{x}_i; \boldsymbol{\beta})\}_i^y \{1 - p(\mathbf{x}_i; \boldsymbol{\beta})\}^{1-y_i}}_{\text{Likelihood}} \times \underbrace{\exp\left(-\frac{1}{2\tau_j^2}(\beta_j - \mu_j)^2\right)}_{\text{Prior}}.$$

- Apply MH to get samples form the full conditional.

MH Algorithm within Gibbs Sampler III

```
mcmc.logit2 <- function(x, y, init, n.sample = 10000, step = rep(0.3, p)){
  n <- nrow(x)
  p <- ncol(x)

  post.beta <- matrix(0, n.sample, p)
  ac.ratio <- matrix(0, n.sample, p)

  prior.m <- 10
  prior.s <- 1000 # for vague prior

  # intialize
  post.beta[1,] <- beta <- init
  eta <- x %*% beta
  pi <- exp(eta)/(1 + exp(eta))

  log.like  <- sum(y * log(pi) +  (1 - y) * log(1 - pi))

  for (iter in 1:n.sample){
    beta.new <- beta
    # candidate
    for (j in 1:p)
    {
      beta.new[j] <- beta[j] + rnorm(1, 0, step[j])
      eta.new <- x %*% beta.new
      pi.new  <- exp(eta.new)/(1 + exp(eta.new))
```

MH Algorithm within Gibbs Sampler IV

```
# prior
      log.prior     <- dnorm(beta[j],     prior.m, prior.s, log = T)
      log.prior.new <- dnorm(beta.new[j], prior.m, prior.s, log = T)

      # liklihood
      log.like.new <- sum(y * log(pi.new) + (1 - y) * log(1 - pi.new))

      # ratio
      temp <- exp((log.like.new + log.prior.new) - (log.like + log.prior))
      rho <- min(1, temp)

      if (runif(1) < rho) {
        ac.ratio[iter,j] <- 1
        beta[j] <- beta.new[j]
        log.like  <- log.like.new
        eta <- x %*% beta
        pi <- exp(eta)/(1 + exp(eta))
      }
    }

    post.beta[iter,] <- beta
  }

  obj <- list(posterior = post.beta, acpt.ratio = apply(ac.ratio, 2, mean))
  return(obj)
}
```

Nonparametric Function Estimation I

- Consider
$$y_i = f(x_i) + \epsilon_i$$
with a random error ϵ_i and $E(\epsilon_i) = 0$ and $\mathrm{Var}(\epsilon_i) = \sigma^2$.
- Suppose $x \in [0,1]$. Bernstein ploynomials of f is given by
$$\begin{aligned} B_M(x; f) &= \sum_{m=0}^{M} f(m/M) \binom{M}{m} x^m (1-x)^{M-m} \\ &= \sum_{m=0}^{M} \beta_m b_M(x, m) \end{aligned}$$
- Weierstrass approximation theorem states that
$$\lim_{M \to \infty} \sup_x |f(x) - B_M(x; f)| \;=\; 0.$$
- That is, we can model $f(x)$ via Bernstein polynomials.

Nonparametric Function Estimation II

- Consider a re-parameterization of $\beta_m = f(m/M), m = 0, 1, \cdots, M$.

$$\gamma_0 = \beta_0, \gamma_m = \beta_m - \beta_{m-1}, \qquad j = 1, \cdots M - 1$$

which yields

$$\beta_m = \sum_{k=0}^{m} \gamma_k.$$

- Notice that

$$\begin{aligned} B_M(x; f) &= \sum_{m=0}^{M} \beta_m b_M(x, m) \\ &= \sum_{m=0}^{M} \sum_{k=0}^{m} \gamma_k b_M(x, m) \\ &= \sum_{k=0}^{M} \gamma_k \sum_{m=k}^{M} b_M(x, m) = \sum_{k=0}^{M} \gamma_k F_M(x, k) \end{aligned}$$

Nonparametric Function Estimation III

where

$$F_M(x,k) = \sum_{m=k}^{M} b_M(x,k) = \int_0^x \frac{u^{m-1}(1-u)^{M-k}}{Beta(k, M-k+1)} du$$

i.e., the distribution function of the Beta random variable with parameters m and $M-m+1$ for $m = 0, 1, 2, \cdots, M$.

- Bernstein polynomial model for $f(x)$ is given by

$$y_i \sim N\left(\sum_{k=0}^{M} \gamma_k F_M(x_i,k), \sigma^2\right)$$

- One can assume the priors on $\gamma_j, j = 0, \cdots, M$ and σ^2.

Nonparametric Function Estimation IV

- Bernstein polynomial is particularly useful under presence of shape constraints.
- We consider a nonparametric dose response model

$$y_i \mid x_i \sim \text{Bern}(p(x_i))$$

where

$$p(x) = P(Y = 1 \mid x) \in [0, 1]$$

- Here $p(x)$ is assumed to be an increasing function of x and $p(0) = 0$ and $p(1) \leq 1$, with $x \in [0, 1]$.

Nonparametric Function Estimation V

- The assumptions can be imposed by:
  1. $\gamma_0 = 0$,
  2. $\gamma_j \geq 0, \quad j = 1, \cdots, p$;
  3. $\beta_M = \sum_{m=1}^{M} \gamma_m \leq 1$

Nonparametric Function Estimation VI

- Applying the Bernstein approximation, we have

$$y_i \mid x_i \sim \mathrm{Bern}(p(x))$$

where

$$p(x) = \sum_{k=1}^{M} \gamma_k F_M(x_i, k)$$

- To impose the 2nd and 3rd constraints, we consider

$$\gamma_1 = \delta_1, \qquad \text{and} \qquad \gamma_j = \delta_j \prod_{k=1}^{j-1} (1 - \delta_k)$$

with the independent uniform prior:

$$\delta_j \overset{iid}{\sim} \mathrm{Uniform}(0, 1), \quad j = 1, 2, \cdots, p.$$

Nonparametric Function Estimation VII

```
set.seed(1)
n <- 200
x <- runif(n)

beta <- 5
f <- function(x, beta) (exp(beta * x) - 1)/(exp(beta) - 1)
p <- f(x, beta)
y <- rbinom(n, 1, p)

n.sample <- 10000
step <- 0.1
M <- 10 # degress of bernstein polynomials

max.x <- 1
min.x <- 0
x.tilde <- (x - min.x)/(max.x - min.x)
W <- unlist(lapply(1:M, function(k) pbeta(x.tilde, k, M-k+1)))
Ft <- matrix(W, ncol = M)

# initialize
delta <- runif(M)

# compute gamma from delta
gamma <- delta * cumprod(1 - c(0, delta[-M]))

# p from gamma
p <- Ft %*% gamma
```

Nonparametric Function Estimation VIII

```
# likelihood
log.like <- sum(y * log(p) + (1-y) * log(1-p))

ac.ratio <- matrix(0, n.sample, M)
post.gamma <- matrix(0, n.sample, M)

for (iter in 1:n.sample) {
delta.new <- delta
gamma.new <- gamma
  for (m in 1:M)
  {
  eta <- log(delta[m]/(1 - delta[m]))
  eta.new <- eta + rnorm(1, 0, step)
  delta.new[m] <- exp(eta.new)/(1 + exp(eta.new))

  gamma.new <- delta.new * cumprod(1 - c(0, delta.new[-M]))

  # update p
  p.new <- Ft %*% gamma.new

  # likelihood
  log.like.new <- sum(y * log(p.new) + (1-y) * log(1-p.new))

  # update
  temp <- exp(log.like.new - log.like)
  rho <- min(1, temp)
  if (runif(1) < rho) {
     ac.ratio[iter,m] <- 1
```

Nonparametric Function Estimation IX

```
      delta[m] <- delta.new[m]
      gamma[m] <- gamma.new[m]
      log.like  <- log.like.new
      p <-Ft %*% gamma
      }
   }
 post.gamma[iter,] <- gamma
 }


 est <- apply(Ft %*% t(post.gamma), 1, quantile, prob = c(0.025, 0.5, 0.975))

 id <- order(x)

 plot(x[id], est[2, id],
      xlab = "x", ylab = "probability", col = 4, type = "l", lwd = 2,
      ylim = c(0,1))
 polygon(c(x[id], rev(x[id])), c(est[1, id], rev(est[3, id])),
         col = rgb(0,0,1, alpha = 0.22),
         border = F)
 t <- seq(0, 1, length = 1000)
 lines(t, f(t, beta), col = 2, lwd = 2, lty = 2)
```

Nonparametric Function Estimation X

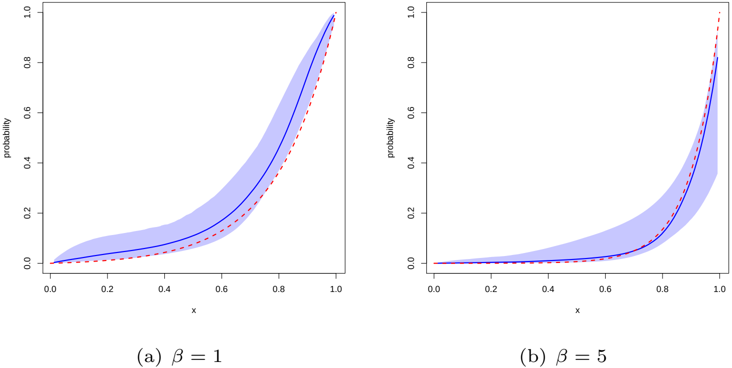

Figure: Estimated Curves with 95% credible bands. Dashed line is the true and solid line is the estimated.