Estadística Bayesiana

Clase 8: Algoritmos de Simulación

Isabel Cristina Ramírez Guevara

Escuela de Estadística
Universidad Nacional de Colombia, Sede Medellín

Medellín, 26 de agosto de 2020

Algoritmos de Simulación

Algunas veces es necesario solucionar integrales de este tipo:

$$I = \int_x g(x)dx.$$

Una alternativa está basada en la generación de muestras aleatorias y luego obtener la solución de la integral a partir de una estimador estadístico insesgado, la media muestral. Suponga que se tiene la distribución de probabilidad $p(x)$, se puede entonces expresar:

$$I = \int_x \left[\frac{g(x)}{p(x)}\right] p(x)dx = \int_x g^*(x)p(x)dx.$$

donde $g^*(x) = \frac{g(x)}{p(x)}$.

Algoritmos de Simulación

Por lo tanto la integral I se puede estimar de la siguiente manera:

- Genere $x^{(1)}, x^{(2)}, \cdots, x^{(T)}$ de la distribución de probabilidad $p(x)$.
- Calcule la media muestral

$$\hat{I} = \frac{1}{T} \sum_{t=1}^{T} \left[\frac{g(x^{(t)})}{p(x^{(t)})} \right]$$

Este método se puede aplicar a algunos problemas de inferencia Bayesiana, así para cada función del parámetro de interés $G(\theta)$, se puede calcular su media y varianza posterior de la siguiente manera:

- Genere $\theta^{(1)}, \theta^{(2)}, \cdots, \theta^{(T)}$ de la distribución de probabilidad $p(\theta|x)$.
- Calcule la media muestral de $G(\theta)$ calculando la cantidad

$$\hat{I} = \frac{1}{T} \sum_{t=1}^{T} G(\theta^{(t)}).$$

Algoritmos de Simulación

Ejemplo

En estudios de epidemiología se estudian tablas de contigencia de 2x2 donde:

- *Y=1 si la persona está enferma.*
- *Y=0 si la persona no está enferma.*
- *X=1 la persona está expuesta a un factor de riesgo.*
- *X=0 la persona no está expuesta a un factor de riesgo.*

En general los investigadores están interesados en estimar el riesgo relativo (RR) y la razón de odds (OR), los cuales están dados por:

$$RR = \frac{p(Y=1|X=1)}{p(Y=1|X=0)} = \frac{\pi_1}{\pi_0}$$

$$OR = \frac{\frac{p(Y=1|X=1)}{p(Y=1|X=0)}}{\frac{p(Y=0|X=1)}{p(Y=0|X=0)}}$$

Algoritmos de Simulación

Ejemplo

$$OR = \frac{p(Y=1|X=1)p(Y=0|X=0)}{p(Y=0|X=1)p(Y=1|X=0)} = \frac{\pi_1(1-\pi_0)}{(1-\pi_1)\pi_0}$$

Para este problema la verosimilitud está dada por:

$$Y_0 \sim Binomial(\pi_0, n_0) \quad y \quad Y_1 \sim Binomial(\pi_1, n_1)$$

donde n_0 es el número de personas que no están expuestas al factor de riesgo y n_1 el número de personas expuestas al factor de riesgo. Los parámetros de interés son $\boldsymbol{\theta} = (\pi_0, \pi_1)$. Si $\pi_0 \sim Beta(a_0, b_0)$ y $\pi_1 \sim Beta(a_1, b_1)$ las distribuciones posteriores son:

$$\pi_0|\mathbf{y}_0 \sim Beta(y_0 + a_0, n_0 - y_0 + b_0)$$

$$\pi_1|\mathbf{y}_1 \sim Beta(y_1 + a_1, n_1 - y_1 + b_1)$$

Algoritmos de Simulación

Ejemplo

Las distribuciones posteriores de RR y OR no se pueden obtener de forma analítica. Simulando directamente de las distribuciones posteriores $\pi_0|\mathbf{y}_0$ *y* $\pi_1|\mathbf{y}_1$ *se pueden obtener las distribuciones de RR y OR.*

Suponga que de 300 personas expuestas al factor de riesgo 25 están enfermas y de 900 personas que no están expuestas al factor de riesgo 30 están enfermas. Por lo tanto, $n_1 = 300$, $y_1 = 25$, $n_0 = 900$ *y* $y_0 = 30$. *Suponga que* $a_0 = b_0 = a_1 = b_1 = 1$, *entonces:*

$$\pi_0|\mathbf{y}_0 \sim Beta(30 + 1, 900 - 30 + 1)$$

$$\pi_1|\mathbf{y}_1 \sim Beta(25 + 1, 300 - 25 + 1)$$

Algoritmos de Simulación

Ejemplo

Para estimar RR y OR se procede de la siguiente manera, para $t = 1, \cdots, T$

1. *Genere* $\pi_0^{(t)} \sim$*Beta(31,871).*
2. *Genere* $\pi_1^{(t)} \sim$*Beta(26,276).*
2. *Calcule* $RR^{(t)}$ *y* $OR^{(t)}$ *utilizando las expresiones:*

$$RR^{(t)} = \frac{\pi_1^{(t)}}{\pi_0^{(t)}}$$

$$OR^{(t)} = \frac{\pi_1^{(t)}(1-\pi_0^{(t)})}{\pi_0^{(t)}(1-\pi_1^{(t)})}$$

Métodos de Simulación Monte Carlo de Cadenas de Markov (MCMC)

En modelos complicados no es posible realizar la integral análitica de la distribución conjunta posterior de los parámetros, ni establecer la constante de normalización. Los algoritmos MCMC son métodos computacionales que se utilizan para obtener muestras de distribuciones posteriores.

Métodos de Simulación Monte Carlo de Cadenas de Markov (MCMC)

En modelos complicados no es posible realizar la integral análitica de la distribución conjunta posterior de los parámetros, ni establecer la constante de normalización. Los algoritmos MCMC son métodos computacionales que se utilizan para obtener muestras de distribuciones posteriores.

Cadenas de Markov

Una cadena de Markov es una secuencia de variables $\theta^{(1)}, \theta^{(2)}$,... tal que la distribución de $\theta^{(t)}$ dados los valores previos $\theta^{(1)}, \theta^{(2)}, \cdots, \theta^{(t-1)}$ solo depende de $\theta^{(t-1)}$, esto es

$$p\left(\theta^{(t)}|\theta^{(1)}, \theta^{(2)}, \cdots, \theta^{(t-1)}\right) = p\left(\theta^{(t)}|\theta^{(t-1)}\right)$$

Métodos de Simulación Monte Carlo de Cadenas de Markov (MCMC)

En modelos complicados no es posible realizar la integral análitica de la distribución conjunta posterior de los parámetros, ni establecer la constante de normalización. Los algoritmos MCMC son métodos computacionales que se utilizan para obtener muestras de distribuciones posteriores.

Cadenas de Markov

Una cadena de Markov es una secuencia de variables $\theta^{(1)}, \theta^{(2)}$,... tal que la distribución de $\theta^{(t)}$ dados los valores previos $\theta^{(1)}, \theta^{(2)}, \cdots, \theta^{(t-1)}$ solo depende de $\theta^{(t-1)}$, esto es

$$p\left(\theta^{(t)}|\theta^{(1)}, \theta^{(2)}, \cdots, \theta^{(t-1)}\right) = p\left(\theta^{(t)}|\theta^{(t-1)}\right)$$

Bajo ciertas condiciones, una cadena de Markov converge a su distribución estacionaria:

$$\lim_{t\to\infty} p\left(\theta^{(t)}|\theta^{(t-1)}\right) = p(\theta)$$

Ideas básicas sobre los métodos MCMC

Supongamos que queremos simular valores de una distribución posterior $p(\theta|y)$

Ideas básicas sobre los métodos MCMC

Supongamos que queremos simular valores de una distribución posterior $p(\theta|y)$

- La idea de los métodos MCMC consiste en simular una cadena de Markov $\theta^{(1)}, \theta^{(2)}, ..., \theta^{(t)}$ cuya distribución estacionaria sea $p(\theta|\mathbf{y})$.

Ideas básicas sobre los métodos MCMC

Supongamos que queremos simular valores de una distribución posterior $p(\theta|y)$

- La idea de los métodos MCMC consiste en simular una cadena de Markov $\theta^{(1)}, \theta^{(2)}, ..., \theta^{(t)}$ cuya distribución estacionaria sea $p(\theta|\mathbf{y})$.
- Cada valor simulado, $\theta^{(t)}$, depende únicamente de su predecesor, $\theta^{(t-1)}$.

Ideas básicas sobre los métodos MCMC

Supongamos que queremos simular valores de una distribución posterior $p(\theta|y)$

- La idea de los métodos MCMC consiste en simular una cadena de Markov $\theta^{(1)}, \theta^{(2)}, ..., \theta^{(t)}$ cuya distribución estacionaria sea $p(\theta|\mathbf{y})$.
- Cada valor simulado, $\theta^{(t)}$, depende únicamente de su predecesor, $\theta^{(t-1)}$.
- Si el algoritmo se implementa correctamente, la convergencia de la cadena está garantizada independientemente de cuáles sean los valores iniciales.

Ideas básicas sobre los métodos MCMC

Supongamos que queremos simular valores de una distribución posterior $p(\theta|y)$

- La idea de los métodos MCMC consiste en simular una cadena de Markov $\theta^{(1)}, \theta^{(2)}, ..., \theta^{(t)}$ cuya distribución estacionaria sea $p(\theta|\mathbf{y})$.
- Cada valor simulado, $\theta^{(t)}$, depende únicamente de su predecesor, $\theta^{(t-1)}$.
- Si el algoritmo se implementa correctamente, la convergencia de la cadena está garantizada independientemente de cuáles sean los valores iniciales.
- Es necesario simular la cadena para un número "elevado" de iteraciones para aproximarse a la distribución estacionaria.

Ideas básicas sobre los métodos MCMC

Supongamos que queremos simular valores de una distribución posterior $p(\theta|y)$

- La idea de los métodos MCMC consiste en simular una cadena de Markov $\theta^{(1)}, \theta^{(2)}, ..., \theta^{(t)}$ cuya distribución estacionaria sea $p(\theta|\mathbf{y})$.
- Cada valor simulado, $\theta^{(t)}$, depende únicamente de su predecesor, $\theta^{(t-1)}$.
- Si el algoritmo se implementa correctamente, la convergencia de la cadena está garantizada independientemente de cuáles sean los valores iniciales.
- Es necesario simular la cadena para un número "elevado" de iteraciones para aproximarse a la distribución estacionaria.
- Los primeros valores simulados, iteraciones burn-in, se eliminan porque no están en el estado estacionario.

Algoritmo Metropolis-Hastings

Suponga que $\boldsymbol{\theta}$ es un vector de parámetros de dimensión q. El primer paso del algoritmo Metropolis-Hastings es generar un candidato, el cual vamos a denotar por $\boldsymbol{\theta}^*$. A menudo, el candidato difiere del valor actual de los parámetros en uno o dos componentes; por ejemplo, en la distribución normal se podría alternar entre actualizar el valor de μ y el valor de σ^2. Un método común para generar el valor candidato $\boldsymbol{\theta}^*$ es sumar una normal estándar a un único componente de $\boldsymbol{\theta}^{(j-1)}$, por ejemplo a $\theta_i^{(j-1)}$. Esto significa que se puede expresar el valor candidato $\boldsymbol{\theta}^*$ como el vector con las componentes:

$$\theta_i^* = \theta_i^{(j-1)} + sZ$$

$$\theta_k^* = \theta_k^{(j-1)} \quad \text{para} \quad k \neq i,$$

donde Z es una normal estándar y s una constante arbitraria.

Algoritmo Metropolis-Hastings

Sea $f(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(j-1)})$ la *densidad propuesta* para generar $\boldsymbol{\theta}^*$ de $\boldsymbol{\theta}^{(j-1)}$. La probabilidad de moverse desde el valor candidato al original, de una manera similar, está dada por $f(\boldsymbol{\theta}^{(j-1)}|\boldsymbol{\theta}^*)$. La densidad propuesta debe satisfacer:

i. permitir moverse desde cualquier subconjunto del espacio parametral o otro subconjunto del espacio parametral en un número finito de movimientos,

Algoritmo Metropolis-Hastings

Sea $f(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(j-1)})$ la *densidad propuesta* para generar $\boldsymbol{\theta}^*$ de $\boldsymbol{\theta}^{(j-1)}$. La probabilidad de moverse desde el valor candidato al original, de una manera similar, está dada por $f(\boldsymbol{\theta}^{(j-1)}|\boldsymbol{\theta}^*)$. La densidad propuesta debe satisfacer:

i. permitir moverse desde cualquier subconjunto del espacio parametral o otro subconjunto del espacio parametral en un número finito de movimientos,

ii. no puede ser periódica, lo cual significa que en una corrida larga movimientos a cualquier subconjunto del espacio parametral puede ocurrir en cualquier momento,

Algoritmo Metropolis-Hastings

Sea $f(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(j-1)})$ la *densidad propuesta* para generar $\boldsymbol{\theta}^*$ de $\boldsymbol{\theta}^{(j-1)}$. La probabilidad de moverse desde el valor candidato al original, de una manera similar, está dada por $f(\boldsymbol{\theta}^{(j-1)}|\boldsymbol{\theta}^*)$. La densidad propuesta debe satisfacer:

i. permitir moverse desde cualquier subconjunto del espacio parametral o otro subconjunto del espacio parametral en un número finito de movimientos,

ii. no puede ser periódica, lo cual significa que en una corrida larga movimientos a cualquier subconjunto del espacio parametral puede ocurrir en cualquier momento,

iii. que cumpla la siguiente regla:

$$0 < \frac{f(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(j-1)})}{f(\boldsymbol{\theta}^{(j-1)}|\boldsymbol{\theta}^*)} < \infty,$$

para todos los valores de $\boldsymbol{\theta}^{(j-1)}$ y $\boldsymbol{\theta}^*$.

Algoritmo Metropolis-Hastings

Luego de tener el valor candidato $\boldsymbol{\theta}^*$ se calcula la probabilidad de que este sea aceptado como el próximo valor simulado de la secuencia. Esto lo vamos a llamar *probabilidad de aceptación*, r, y está definido como:

$$r = \min\left(1, \frac{p(\boldsymbol{\theta}^*|\text{datos})}{p(\boldsymbol{\theta}^{(j-1)}|\text{datos})} \frac{f(\boldsymbol{\theta}^{(j-1)}|\boldsymbol{\theta}^*)}{f(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(j-1)})}\right)$$

La primera razón hace que el algoritmo se mueva a los valores de los parámetros que tengan una alta probabilidad posterior, la segunda razón tiene en cuenta que la densidad propuesta podría favorecer algunos valores de los parámetros sobre otros. Si la distribución propuesta es simétrica, $f(\boldsymbol{\theta}^{(j-1)}|\boldsymbol{\theta}^*) = f(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(j-1)})$, la segunda razón es 1 y se puede omitir de la formula de la probabilidad de aceptación.

Algoritmo Metropolis-Hastings

Después de calcular la probabilidad de aceptación, se lleva a cabo el tercer paso del algoritmo Metropolis-Hastings. Se simula un valor de una Uniforme(0,1), digamos u, y se compara con r. Si $u \leq r$, entonces se acepta el valor candidato y $\boldsymbol{\theta}^{(j)} = \boldsymbol{\theta}^*$. Si $u > r$, se rechaza el candidato y $\boldsymbol{\theta}^{(j)} = \boldsymbol{\theta}^{(j-1)}$. Este proceso se repite para cada componente de $\boldsymbol{\theta}$.

Algoritmo Metropolis-Hastings

Después de calcular la probabilidad de aceptación, se lleva a cabo el tercer paso del algoritmo Metropolis-Hastings. Se simula un valor de una Uniforme(0,1), digamos u, y se compara con r. Si $u \leq r$, entonces se acepta el valor candidato y $\boldsymbol{\theta}^{(j)} = \boldsymbol{\theta}^*$. Si $u > r$, se rechaza el candidato y $\boldsymbol{\theta}^{(j)} = \boldsymbol{\theta}^{(j-1)}$. Este proceso se repite para cada componente de $\boldsymbol{\theta}$.

Suponga que se tiene un experimento binomial con 3 éxitos en 11 intentos, por lo tanto la verosimilitud es proporcional a $\pi^3(1-\pi)^8$. Si se asigna como distribución a priori una uniforme en el intervalo (0.1,0.9) se tiene que:

$$p(\pi|datos) \propto \pi^3(1-\pi)^8 \quad \text{si} \quad 0{,}1 < \pi < 0{,}9$$

Esta distribución no tiene la forma estándar de una distribución beta dado que no está definida para $0 < \pi < 1$.

Algoritmo Metropolis-Hastings

El algoritmo Metropolis-Hastings para muestrear de esta distribución posterior opera de la siguiente forma:

0. Inicializar $j = 0$ y $\pi^{(j)} = 0.5$.

Algoritmo Metropolis-Hastings

El algoritmo Metropolis-Hastings para muestrear de esta distribución posterior opera de la siguiente forma:

0. Inicializar $j = 0$ y $\pi^{(j)} = 0.5$.
1. Genere π^* de una distribución Uniforme(0.1,0.9).

Algoritmo Metropolis-Hastings

El algoritmo Metropolis-Hastings para muestrear de esta distribución posterior opera de la siguiente forma:

0. Inicializar $j = 0$ y $\pi^{(j)} = 0.5$.
1. Genere π^* de una distribución Uniforme(0.1,0.9).
2. Calcule $r = \text{mín}\left(1, \frac{(\pi^*)^3(1-\pi^*)^8}{(\pi^{(j-1)})^3(1-\pi^{(j-1)})^8}\right)$.

Algoritmo Metropolis-Hastings

El algoritmo Metropolis-Hastings para muestrear de esta distribución posterior opera de la siguiente forma:

0. Inicializar $j = 0$ y $\pi^{(j)} = 0.5$.
1. Genere π^* de una distribución Uniforme(0.1,0.9).
2. Calcule $r = \text{mín}\left(1, \frac{(\pi^*)^3(1-\pi^*)^8}{(\pi^{(j-1)})^3(1-\pi^{(j-1)})^8}\right)$.
3. Genere u de una distribución Uniforme(0,1).

Algoritmo Metropolis-Hastings

El algoritmo Metropolis-Hastings para muestrear de esta distribución posterior opera de la siguiente forma:

0. Inicializar $j = 0$ y $\pi^{(j)} = 0.5$.
1. Genere π^* de una distribución Uniforme(0.1,0.9).
2. Calcule $r = \text{mín}\left(1, \frac{(\pi^*)^3(1-\pi^*)^8}{(\pi^{(j-1)})^3(1-\pi^{(j-1)})^8}\right)$.
3. Genere u de una distribución Uniforme(0,1).
4. Si $u \leq r$ entonces $\pi^{(j)} = \pi^*$. De lo contrario, $\pi^{(j)} = \pi^{(j-1)}$.

Algoritmo Metropolis-Hastings

El algoritmo Metropolis-Hastings para muestrear de esta distribución posterior opera de la siguiente forma:

0. Inicializar $j = 0$ y $\pi^{(j)} = 0.5$.
1. Genere π^* de una distribución Uniforme(0.1,0.9).
2. Calcule $r = \text{mín}\left(1, \frac{(\pi^*)^3(1-\pi^*)^8}{(\pi^{(j-1)})^3(1-\pi^{(j-1)})^8}\right)$.
3. Genere u de una distribución Uniforme(0,1).
4. Si $u \leq r$ entonces $\pi^{(j)} = \pi^*$. De lo contrario, $\pi^{(j)} = \pi^{(j-1)}$.
5. Incremente j y regrese al paso 1.

Algoritmo Metropolis-Hastings

El algoritmo Metropolis-Hastings para muestrear de esta distribución posterior opera de la siguiente forma:

0. Inicializar $j = 0$ y $\pi^{(j)} = 0.5$.
1. Genere π^* de una distribución Uniforme(0.1,0.9).
2. Calcule $r = \text{mín}\left(1, \frac{(\pi^*)^3(1-\pi^*)^8}{(\pi^{(j-1)})^3(1-\pi^{(j-1)})^8}\right)$.
3. Genere u de una distribución Uniforme(0,1).
4. Si $u \leq r$ entonces $\pi^{(j)} = \pi^*$. De lo contrario, $\pi^{(j)} = \pi^{(j-1)}$.
5. Incremente j y regrese al paso 1.

En este ejemplo se selecciona como densidad propuesta a la distribución a priori. Como la densidad propuesta en este algoritmo no depende de $\pi^{(j-1)}$, a este algoritmo Metropolis-Hastings se le llama muestreador independiente.

Algoritmo Metropolis-Hastings

En general, los muestreadores independientes funcionan bien cuando la densidad propuesta representa una aproximación razonable de la densidad posterior. Trabajan pobremente cuando la densidad propuesta asigna muy poca masa cercana a la región del espacio parametral donde la densidad posterior está más concentrada.

Algoritmo Metropolis-Hastings

En general, los muestreadores independientes funcionan bien cuando la densidad propuesta representa una aproximación razonable de la densidad posterior. Trabajan pobremente cuando la densidad propuesta asigna muy poca masa cercana a la región del espacio parametral donde la densidad posterior está más concentrada.

En problemas de alta dimensión la propiedad de independencia de los muestreadores tiene dos implicaciones, primero al muestreador le podría tomar un tiempo muy largo para encontra valores cercanos a la masa de la distribución posterior, por lo tanto se necesitaría un tiempo de "burn-in"muy largo. Segundo, una vez se encuentra un punto cercano a la masa de la densidad posterior, muchas iteraciones tienen que suceder antes de que este punto sea reemplazado por otro con alta probabilidad posterior.

Muestreador de Gibbs

El éxito del algoritmo Metropolis-Hastings depende de elección razonable de una distribución propuesta, lo cual, en algunos casos, puede ser difícil. Cuando la selección de la densidad propuesta no es la adecuada se pueden tener dos problemas.

Muestreador de Gibbs

El éxito del algoritmo Metropolis-Hastings depende de elección razonable de una distribución propuesta, lo cual, en algunos casos, puede ser difícil. Cuando la selección de la densidad propuesta no es la adecuada se pueden tener dos problemas. Primero, si la densidad propuesta tiene una varianza muy pequeña (estrecha), el algoritmo Metropolis-Hastings podría permanecer mucho tiempo en una región limitada del espacio parametral. Adicionalmente, hace que se tenga una alta correlación de las cadenas, haciendo el tamaño de muestra efectivo muy pequeño.

Muestreador de Gibbs

El éxito del algoritmo Metropolis-Hastings depende de elección razonable de una distribución propuesta, lo cual, en algunos casos, puede ser difícil. Cuando la selección de la densidad propuesta no es la adecuada se pueden tener dos problemas. Primero, si la densidad propuesta tiene una varianza muy pequeña (estrecha), el algoritmo Metropolis-Hastings podría permanecer mucho tiempo en una región limitada del espacio parametral. Adicionalmente, hace que se tenga una alta correlación de las cadenas, haciendo el tamaño de muestra efectivo muy pequeño. Por otro lado, si la densidad propuesta tiene una varianza muy grande, la cadena podría permanecer en un único estado durante cientos o miles de iteraciones.

Muestreador de Gibbs

El éxito del algoritmo Metropolis-Hastings depende de elección razonable de una distribución propuesta, lo cual, en algunos casos, puede ser difícil. Cuando la selección de la densidad propuesta no es la adecuada se pueden tener dos problemas. Primero, si la densidad propuesta tiene una varianza muy pequeña (estrecha), el algoritmo Metropolis-Hastings podría permanecer mucho tiempo en una región limitada del espacio parametral. Adicionalmente, hace que se tenga una alta correlación de las cadenas, haciendo el tamaño de muestra efectivo muy pequeño. Por otro lado, si la densidad propuesta tiene una varianza muy grande, la cadena podría permanecer en un único estado durante cientos o miles de iteraciones.

Un mejor método MCMC para muestrear de una distribución posterior se puede obtener reemplazando la densidad propuesta en el algoritmo Metropolis-Hastings por la distribución condicional de los parámetros que ya fueron muestreados. Esto es el muestreador de Gibbs.

Muestreador de Gibbs

Suponga que $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_q)$ y se tienen las siguientes distribuciones posteriores condicionales:

$$
\begin{aligned}
p_1(\theta_1|\theta_2, \cdots, \theta_q, datos) \\
p_2(\theta_2|\theta_1, \theta_3, \cdots, \theta_q, datos) \\
\vdots \\
p_2(\theta_q|\theta_1, \cdots, \theta_{q-1}, datos)
\end{aligned}
$$

Muestreador de Gibbs

Algoritmo del muestreador de Gibbs

0. Inicializar $j = 0, \boldsymbol{\theta}^{(0)} = (\theta_1^{(0)}, \cdots, \theta_q^{(0)})$.

Muestreador de Gibbs

Algoritmo del muestreador de Gibbs

0. Inicializar $j = 0, \boldsymbol{\theta}^{(0)} = (\theta_1^{(0)}, \cdots, \theta_q^{(0)})$.
1. Generar $\theta_1^{(j+1)} \sim p_1(\theta_1 | \theta_2^{(j)}, \cdots, \theta_q^{(j)}, datos)$

Muestreador de Gibbs

Algoritmo del muestreador de Gibbs

0. Inicializar $j = 0, \boldsymbol{\theta}^{(0)} = (\theta_1^{(0)}, \cdots, \theta_q^{(0)})$.
1. Generar $\theta_1^{(j+1)} \sim p_1(\theta_1 | \theta_2^{(j)}, \cdots, \theta_q^{(j)}, datos)$
2. Generar $\theta_2^{(j+1)} \sim p_2(\theta_2 | \theta_1^{(j+1)}, \theta_3^{(j)}, \cdots, \theta_q^{(j)}, datos)$.

$\vdots$

Muestreador de Gibbs

Algoritmo del muestreador de Gibbs

0. Inicializar $j = 0, \boldsymbol{\theta}^{(0)} = (\theta_1^{(0)}, \cdots, \theta_q^{(0)})$.
1. Generar $\theta_1^{(j+1)} \sim p_1(\theta_1 | \theta_2^{(j)}, \cdots, \theta_q^{(j)}, datos)$
2. Generar $\theta_2^{(j+1)} \sim p_2(\theta_2 | \theta_1^{(j+1)}, \theta_3^{(j)}, \cdots, \theta_q^{(j)}, datos)$.

$\vdots$

q. Generar $\theta_q^{(j+1)} \sim p_2(\theta_q | \theta_1^{(j+1)}, \theta_2^{(j+1)}, \cdots, \theta_{q-1}^{(j+1)}, datos)$.

Muestreador de Gibbs

Algoritmo del muestreador de Gibbs

0. Inicializar $j = 0, \boldsymbol{\theta}^{(0)} = (\theta_1^{(0)}, \cdots, \theta_q^{(0)})$.
1. Generar $\theta_1^{(j+1)} \sim p_1(\theta_1 | \theta_2^{(j)}, \cdots, \theta_q^{(j)}, datos)$
2. Generar $\theta_2^{(j+1)} \sim p_2(\theta_2 | \theta_1^{(j+1)}, \theta_3^{(j)}, \cdots, \theta_q^{(j)}, datos)$.

$\vdots$

q. Generar $\theta_q^{(j+1)} \sim p_2(\theta_q | \theta_1^{(j+1)}, \theta_2^{(j+1)}, \cdots, \theta_{q-1}^{(j+1)}, datos)$.

q+1. Incremente j y regrese al paso 1.

Muestreador de Gibbs

Ejemplo

Suponga que se desea estudiar tiempos de falla de una proceso, se considera una verosimilitud lognormal, en este caso hay dos parámetros desconocidos μ y σ^2. Se tiene el siguiente modelo:

$$p(\mathbf{y}|\mu,\sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(y_i-\mu)^2\right]$$

$$p(\mu,\sigma^2) \propto \frac{1}{\sigma^2}$$

donde $y_i = \log(t_i)$. Indique los pasos del muestreador de Gibbs para muestrear de las distribuciones posteriores de μ y σ^2.

Inferencia y Evaluación de la convergencia

Para hacer inferencia Bayesiana a partir de las simulaciones obtenidas con los métodos descritos anteriormente es necesario utilizar las muestras de la distribución posterior $p(\theta|\mathbf{y})$ para calcular cuantiles, promedios, momentos y otros estadísticos que resumen el comportamiento de la distribución de probabilidad. De igual forma, para la inferencia predictiva posterior de valores no observados $\tilde{y}$ se toman muestras de la distribución de los datos condicional a los valores simulados de $p(\theta|\mathbf{y})$ y se obtienen los estadísticos de interés que resumen el comportamiento de $p(\tilde{y}|\mathbf{y})$.

Sin embargo hay que tomar algunas previsiones con las simulaciones obtenidas para asegurar que la inferencia se haga sobre simulaciones que son representativas de la distribución de interés.

Inferencia y Evaluación de la convergencia

Si las simulaciones no son lo suficientemente largas, pueden no ser representativas de la distribución de interés. El segundo problema es que si se presenta una alta autocorrelación dentro de la secuencia simulada, el número efectivo de muestras representativas de la distribución de interés es menor que el número de valores simulados. Las posibles formas de manejar estos problemas son enumeradas a continuación:

Inferencia y Evaluación de la convergencia

Si las simulaciones no son lo suficientemente largas, pueden no ser representativas de la distribución de interés. El segundo problema es que si se presenta una alta autocorrelación dentro de la secuencia simulada, el número efectivo de muestras representativas de la distribución de interés es menor que el número de valores simulados. Las posibles formas de manejar estos problemas son enumeradas a continuación:

- Simular múltiples secuencias con valores iniciales dispersos a través del espacio de parámetros.

Inferencia y Evaluación de la convergencia

Si las simulaciones no son lo suficientemente largas, pueden no ser representativas de la distribución de interés. El segundo problema es que si se presenta una alta autocorrelación dentro de la secuencia simulada, el número efectivo de muestras representativas de la distribución de interés es menor que el número de valores simulados. Las posibles formas de manejar estos problemas son enumeradas a continuación:

- Simular múltiples secuencias con valores iniciales dispersos a través del espacio de parámetros.
- Si la eficiencia en la simulación es muy baja, el algoritmo puede ser alterado mediante reparametrizaciones o la construcciones de mejores densidades propuestas.

Inferencia y Evaluación de la convergencia

Si las simulaciones no son lo suficientemente largas, pueden no ser representativas de la distribución de interés. El segundo problema es que si se presenta una alta autocorrelación dentro de la secuencia simulada, el número efectivo de muestras representativas de la distribución de interés es menor que el número de valores simulados. Las posibles formas de manejar estos problemas son enumeradas a continuación:

- Simular múltiples secuencias con valores iniciales dispersos a través del espacio de parámetros.
- Si la eficiencia en la simulación es muy baja, el algoritmo puede ser alterado mediante reparametrizaciones o la construcciones de mejores densidades propuestas.
- Tomar una muestra para cada k-ésima interacción para algún k, de tal forma que se puedan obtener muestras aproximadamente independientes de la distribución de interés (thinnig).

Inferencia y Evaluación de la convergencia

Para explorar si la cadena está explorando bien el espacio de estados, se pueden graficar los valoressimulados de θ_i^k versus t y analizar si se producen desviaciones de estacionaridad.
Para examinar la convergencia de las medias posteriores de los parámetros estimados, se puede graficar el valor de la media muestral de los valores simulados en función de t y verificar si han convergido.
Finalmente se pueden graficar las funciones de autocorrelación de los valores generados.