# MAT703.Deber Seminario Investigación. Ejemplos Capítulo 11 "Bayesian Data Analysis"

Fausto Fabian Crespo Fernandez

Junio 2016

## 0.1. Monte Carlo Markov Chain MCMC

El método consiste en extraer valores de $\theta$ de aproximaciones a la distribución y luego corregir esos valores para aproximar mejor la distribución objetivo $p(\theta|y)$. El muestreo se hace secuencialmente y la distribución de los valores extraídos depende del último valor extraído. por tanto las extracciones son de una cadena de Markov.El método funcionan porque en cada paso las distribuciones aproximadas convergen a la distribución objetivo.
En la simulación de cadena de Markov, creamos secuencias inpedendientes y cada secuencias es comenzada por un valor inicial $\theta^0$ y entonces para cada $t$ escogemos $\theta^t$ de la dstribución de transición $T_t(\theta^t|\theta^{t-1})$ que depende de la extraccion anterior $\theta^{t-1}$. A veces es conveniente que la dstribución de transición depende de $t$ y por eso la notación $T_t$.Las distribuciones de transición deben construirse de manera que converjan a $p(\theta|y)$.
La simulación de cadena de Markov, se usa cuando no es posible o no es eficiente computacionalmente extraer directamente valores de $\theta$ de la a posteriori $p(\theta|y)$ y en lugar de eso se extrae de manera iterada de forma tal que en cada paso del proceso extraemos de una distribuión que esta más cerca de $p(\theta|y)$.

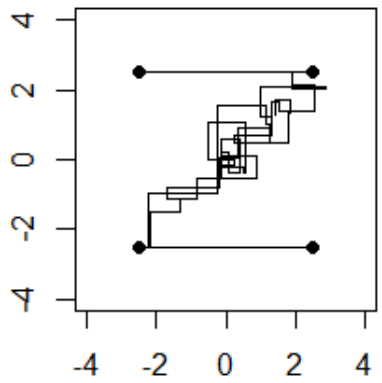
## 0.2. Muestreo de Gibbs

Es útil en problemas multi dimensionales y también se conoce como muestreo condicional alternante y esta definido en términos de subvectores de $\theta$ Si el vector $\theta$ está subdividido en $d$ comppnentes o subvectores $\theta = (\theta_1, \ldots, \theta_d)$, cada iteración recorreo las componentes de $\theta$, extrayendo cada componenente condicionalmente a las otras.Por tanto hay $d$ pasos en iteración $t$. En la iteración $t$, se escoje un ordenamiento de los $d$ subvectores de $\theta$ y cada $\theta_j^t$ se extrae de la distribuión condicional dado los otros componentes:$p(\theta_j|\theta_{-j}^{t-1}, y)$ donde $\theta_{-j}^{t-1}$ son todas las componentes de $\theta$ excepto $\theta_j$ en sus valores actuales $\theta_{-j}^{t-1} = (\theta_1^t, \ldots, \theta_{j-1}^t, \theta_{j+1}^{t-1}, \ldots, \theta_d^{t-1})$ o sea cada $\theta_j$ es actualizado condicionalmente a los últimos valores de las otras componentes de $\theta$

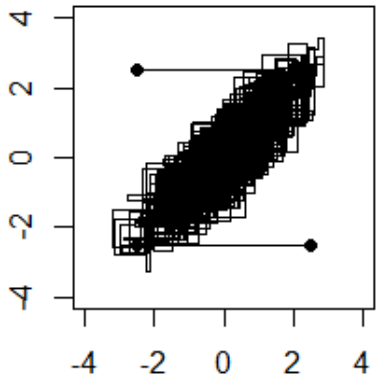
## 0.3. Ejemplo Distribución normal bivariada

Consideremos una única observación $(y_1, y_2)$de una normal bivariada con parámetros $\theta = (\theta_1, \theta_2)$ con matriz de varianza conocida $\begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$ con una distribución uniforme a priori para $\theta$, la distribución a posteriori es:
$\begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} \Big| y \sim N(\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix})$
A pesar de que es simple extraer de la aposteriori de $(\theta_1, \theta_2)$ podemos usar muestreo de Gibbs.Necesitamos las distribuciones condicionales a posteriori
$\theta_1|\theta_2, y \sim N(y_1 + \rho(\theta_2 - y_2), 1 - \rho^2)$
$\theta_2|\theta_2, y \sim N(y_2 + \rho(\theta_1 - y_1), 1 - \rho^2)$
El muestreo Gibbs iterativamente muestrea de estas dos distribuciones normales . Ua forma natural de empezar las iteraciones esextraer de una aproximación normal a la a posteriori .Para $\rho = 0{,}8$ y $(y_1, y_2) = (0, 0)$ y 4 secuancias inpependientes que se inician en $(\pm 2{,}5, \pm 2{,}5)$. Primero con 10 iteraciones, en R :
$chains < -4$
$iter < -10$
$rho = 0{,}8$
$sims < -array(NA, c(iter, chains, 2))$
$dimnames(sims) < -list(NULL, NULL,$
$c("theta1","theta2"))$
$sims[1, 1, ] = c(2{,}5, 2{,}5)$
$sims[1, 2, ] = c(2{,}5, -2{,}5)$
$sims[1, 3, ] = c(-2{,}5, 2{,}5)$
$sims[1, 4, ] = c(-2{,}5, -2{,}5)$
$theta1_update = function(){$
$return(rnorm(1, rho * theta2, sqrt(1 - rho^2)))$
$}$
$theta2_update = function(){$
$return(rnorm(1, rho * theta1, sqrt(1 - rho^2)))$
$}$
$par(pty = "s")$
$plot(x = NULL,$
$y = NULL,$
$xlim = range(-4 : 4),$
$ylim = range(-4 : 4), xlab = "", ylab = "")$

$)$
$points(x = c(2{,}5, 2{,}5, -2{,}5, -2{,}5), y = c(2{,}5, -2{,}5, 2{,}5, -2{,}5), pch = 19)$
$for(min1 : chains)\{$
$for(tin1 : (iter - 1))\{$
$theta1 < -sims[t, m, 1]$
$theta2 < -sims[t, m, 2]$
$theta1 < -theta1_update()$
$segments(x0 = sims[t, m, 1], y0 = sims[t, m, 2], x1 = theta1, y1 =$
$sims[t, m, 2])$
$theta2 < -theta2_update()$
$sims[t + 1, m, ] < -c(theta1, theta2)$
$segments(x0 = theta1, y0 = sims[t, m, 2], x1 = theta1, y1 = theta2)$
$\}$
$\}$




Para 500 iteraciones se llega a convergencia aproximada




## 0.4. 11.2 Algoritmos Metropolis y Metropolis-Hastings

El muestreo de Gibbs se puede ver como un caso particular de Metropolis-Hastings.El algoritmo de Metropolis-Hastings es una adaptación de camino aleatorio con un regla de aceptación-rechazo para converger a una distribución objetivo especifica. Consiste en:
1-Escoger un punto inicial $\theta_0$ para el cual $p(\theta_0|y) > 0$a partir de una distribución inicial $p_0(\theta)$
2-Para $t = 1, 2, \ldots$
a) Escoger una propuesta $\theta^*$ de la distribicón de salto(o distribución propuesta) en el tiempo $t$,$J_t(\theta^*|\theta^{t-1})$. Para el algoritmo Metropolis(pero no para el Metropolis-Hastings) la distribución de salto puede ser simétrica o sea que satisface la condición $J_t(\theta_a|\theta_b) = J_t(\theta_b|\theta_a)$ para todo $\theta_a, \theta_b$ y $t$
b) Calcular la razón de las densidades $r = \frac{p(\theta^*|y)}{p(\theta^{t-1}|y)}$
c) Hacer

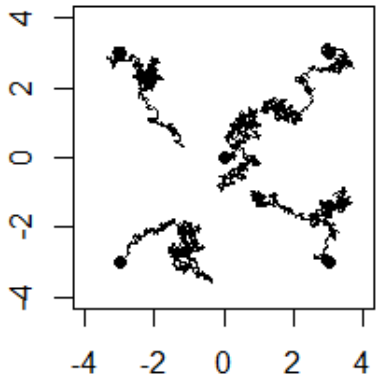
$$\theta^t = \begin{cases} \theta^* & \text{con probabilidad } min(r, 1) \\ \theta^{t-1} & \text{en otro caso} \end{cases}$$

Dado el valor actual $\theta^{t-1}$ la distribución de transición de la cadena de Markox $T_t(\theta^t|\theta^{t-1})$ de densidad de masa $\theta^t = \theta^{t-1}$ y una versión ponderada de la distribución de salto $J_t(\theta^t|\theta^{t-1})$ que ajusta la tasa de aceptación.
El método requiere la posibilidad de calcular la razón $r$ para todos $(\theta, \theta^*)$ y extraer $\theta$ de la distribución de salto $J_t(\theta^t|\theta^{t-1})$ para todo $\theta$ y $t$ y el paso c) requiere generar un número aleatorio uniforme.
Cuando $\theta^t = \theta^{t-1}$ o sea cuando el salto no es aceptado, aún asi cuenta como una iteración del algoritmo.

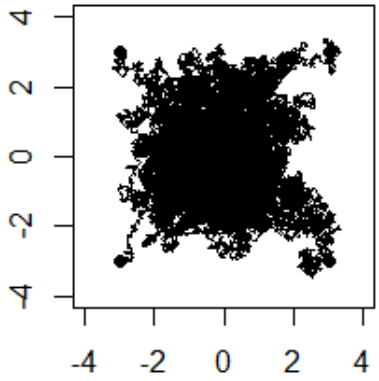
## 0.5. Ejemplo Densidad normal bivariada unitaria con kernel de salto normal

La densidad objetivo es una normal bivarada unitaria $p(\theta|y) = N(\theta|0, I)$ donde $I$ es la matrix identidad $2x2$, la distribución de salto es también normal bivariada, centrada en cada iteración y escalada por factor $1/5$ del tamaño: $J_t(\theta^*|\theta^{t-1}) = N(\theta^*|\theta^{t-1}, 0{,}2^2 I)$. En cada paso es fácil calcular la razón de densidad $r = \frac{N(\theta^*|0,I)}{N(\theta^{t-1}|0,I)}$. Es claro por la forma de la distribución normal que la regla de salto es simétrica .En R
$chains < -5$
$iter < -1000$
$sims < -array(NA, c(iter, chains, 2))$
$dimnames(sims) < -list(NULL, NULL,$
$c("theta1", "theta2"))$
$sims[1, 1, ] = c(3, 3)$
$sims[1, 2, ] = c(3, -3)$
$sims[1, 3, ] = c(-3, 3)$
$sims[1, 4, ] = c(-3, -3)$
$sims[1, 5, ] = c(0, 0)$
$theta1_update = function(){$
$return(rnorm(1, theta1, 0{,}2^2))$
$\}$
$theta2_update = function(){$
$return(rnorm(1, theta2, 0{,}2^2))$
$\}$
$par(pty = "s")$
$plot(x = NULL,$

$y = NULL,$
$xlim = range(-4:4),$
$ylim = range(-4:4), xlab = "", ylab = ""$
$)$
$points(x = c(3,3,-3,-3,0), y = c(3,-3,3,-3,0), pch = 19)$
$r = 0$
$u = 0$
$for(min1 : chains)\{$
$for(tin1 : (iter-1))\{$
$theta1 < -sims[t,m,1]$
$theta2 < -sims[t,m,2]$
$theta1 < -theta1_u pdate()$
$theta2 < -theta2_u pdate()$
$r = dnorm(theta1,0,1) * dnorm(theta2,0,1)/(dnorm(sims[t,m,1],0,1) *$
$dnorm(sims[t,m,2],0,1))$
$r = min(r,1)$
$u = runif(1,0,1)$
$if(u < r)\{$
$sims[t+1,m,] < -c(theta1, theta2)$
$\}$
$else\{$
$sims[t+1,m,] < -c(sims[t,m,1], sims[t,m,2])$
$\}$
$segments(x0 = sims[t,m,1], y0 = sims[t,m,2], x1 = sims[t+1,m,1], y1 =$
$sims[t+1,m,2])$
$\}$
$\}$




Y con 20000 iteraciones obtenemos:




La regla de aceptación/rechazo del algoritmo Metropolis se puede expresar como sigue: si el salto aumenta la densidad a posteriori, entonces hacer $\theta^t = \theta^*$, si el salto decrementa la densidad a posteriori hacer $\theta^t = \theta^*$ con probabilidad igual a la razón de densidad $r$ y hacer $\theta^t = \theta^{t-1}$ en otro caso.

La demostración de porque la secuancia de $\theta^1, \theta^2, ...$ converge a la distribución objetivo tiene dos pasos: primero hay que mostrar que la secuencia simulada es una cadena de Markov con una distribución estacionaria única y segundo que esta distribución estacionaria es la distribución objetivo.Lo primero se cumple si la cadena de Markov es irreducible, apériodica y no transitoria. Excepto por casos triviales, las dos últimas condiciones se cumplen para el camino aleatorio sobre cualquier distribuión propia y la irreducibilidad se cumple mientras el camino aleatorio tienen probabilidad positiva de eventualmente alcanzar cualquier estado desde cualquier otro estado o sea las distribuciones de salto $J_t$ deben ser capaz de eventualmente saltar a todos los estados con probabilidad positiva.
Pqara demostrar que la distribución objetivo es la distribución estacionaria de la cadena de Markov generada por el algoritmo de Metropolis, consideremos que el algoritmo empieza en $t-1$ con la extracción $\theta^{t-1}$ de la distribución objetivo $p(\theta|y)$. Ahora consideremos dos puntos $\theta_a$ y $\theta_b$ extraidas de $p(\theta|y)$ y etiquetadas de forma tal que $p(\theta_b|y) \geq p(\theta_a|y)$. La densidad de probabilidad incondicional de transición de $\theta_a$ a $\theta_b$ es
$p(\theta^{t-1} = \theta_a, \theta^t = \theta_b) = p(\theta_a|y)J_t(\theta_b|\theta_a)$ donde la probabilida de aceptación es 1 por cómo pusimos las etiquetas $a$ y $b$ y la densidas de probabilidad incondicional de transición de $\theta_b$ a $\theta_a$ es
$p(\theta^t = \theta_a, \theta^{t-1} = \theta_b) = p(\theta_b|y)J_t(\theta_a|\theta_b)(\frac{p(\theta_a|y)}{p(\theta_b|y)}) = p(\theta_a|y)J_t(\theta_a|\theta_b)$ o se la misma que la probabailidad de transición de $\theta_a$ a $\theta_b$ pues $J_t(.|.)$ es simétrica.Como la distribución conjunta de $\theta^{t-1}$ y $\theta^t$ es conjunta entonces tienen distribuciones marginales iguales y $p(\theta|y)$ es la distribución estacionaria de la cedna de Markov para $\theta$.
Algoritmo Metropolis-Hastings
Este algortimo generaliza la idea del algortimo de Metropolis en dos formas.Primero las distribuciones de salto $J_t$ no tienen que ser simétricas o sea que el requerimiento $J_t(\theta_a|\theta_b) = J_t(\theta_b|\theta_a)$ no es necesario . Segundo para corregir la asimetria en la regla de salto, la razón $r$ es reeemplazada por la razón de razones siguiente:
$r = \frac{p(\theta^*|y)/J_t(\theta^*|\theta^{t-1})}{p(\theta^{t-1}|y)/J_t(\theta^{t-1}|\theta^*)}$
La razón $r$ siempre está definida porque un salto de $\theta^{t-1}$ a $\theta^*$ solo puede ocurrir si ambas $p(\theta^{t-1}|y)$ y $J_t(\theta^*|\theta^{t-1})$ son distintas de 0. La falta de simetría en la regla de salto puede aumentar la velocidad de convergencia del camino aleatorio.La demostración de la convergencia es parecida a la del algoritmo Metropolis. La demostración de convergencia a una única distribución

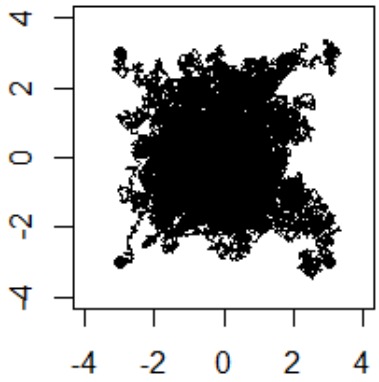
estacionaria es idéntica.Para probar que la distribución estacionaria es la distribución objetivo $p(\theta|y)$ consideremos $\theta_a$ y $\theta_b$ con densidades a posteriori etiquetadas tal que $p(\theta_b|y)J_t(\theta_a|\theta_b) \geq p(\theta_a|y)J_t(\theta_b|\theta_a)$. Si $\theta^{t-1}$ sigue la distribución objetivo, entonces es fácil mostrar que la densidad de probabilidad incondicional de transición de $\theta_a$ a $\theta_b$ es la misma que la transición inversa.
Existe relación entre la regla de salto y la eficiencia de las simulaciones. La regla ideal es extraer la propuesta $\theta^*$ de la distribución objetivo, o sea $J_t(\theta_*|\theta) = p(\theta^*|y)$ para todo $\theta$. Entonces la razón $r$ es exactamente 1 y las iteraciones $\theta_t$ son una secuencia de extracciones independientes de la distribución $p(\theta|y)$, pero en general el método se aplica cuando no es posible extraer directamente de la a posteriori.Una buena función de salto cumple:
-para todo valor de $\theta$ es fácil extraer de $J(\theta^*|\theta)$
-es fácil calcular $r$
-cada salto se mueve a una distancia razonable en el espacio de parámetros.
-los saltos no son rechazados muy frecuentemente
El algoritmo Metropolis Hastings aplicado el ejemplo anterior en R con 20000 iteraciones:

$chains < -5$
$iter < -20000$
$sims < -array(NA, c(iter, chains, 2))$
$dimnames(sims) < -list(NULL, NULL,$
$c(\text{"}theta1\text{"},$
$\text{"}theta2\text{"}))$
$sims[1, 1, ] = c(3, 3)$
$sims[1, 2, ] = c(3, -3)$
$sims[1, 3, ] = c(-3, 3)$
$sims[1, 4, ] = c(-3, -3)$
$sims[1, 5, ] = c(0, 0)$
$theta1_update = function(){$
$return(rnorm(1, theta1, 0,2^2))$
$\}$
$theta2_update = function(){$
$return(rnorm(1, theta2, 0,2^2))$

$par(pty = \text{"}s\text{"})$
$plot(x = NULL,$
$y = NULL,$
$xlim = range(-4 : 4),$

$ylim = range(-4:4), xlab = "", ylab = ""$
)
$points(x = c(3,3,-3,-3,0), y = c(3,-3,3,-3,0), pch = 19)$
$r = 0$
$u = 0$
$for(min1 : chains)\{$
$for(tin1 : (iter-1))\{$
$theta1 < -sims[t,m,1]$
$theta2 < -sims[t,m,2]$
$theta1 < -theta1_upate()$
$theta2 < -theta2_upate()$
$r = dnorm(theta1,0,1) * dnorm(theta2,0,1) * dnorm(sims[t,m,1], theta1, 0, 2^2) *$
$dnorm(sims[t,m,2], theta2, 0, 2^2)$
$/(dnorm(sims[t,m,1],0,1) * dnorm(sims[t,m,2],0,1) * dnorm(theta1, sims[t,m,1], 0, 2^2) *$
$dnorm(theta2, sims[t,m,2], 0, 2^2))r = min(r,1)$
$u = runif(1,0,1)$
$if(u < r)\{$
$sims[t+1,m,] < -c(theta1, theta2)$
$\}$
$else\{$
$sims[t+1,m,] < -c(sims[t,m,1], sims[t,m,2])$
$\}$
$segments(x0 = sims[t,m,1], y0 = sims[t,m,2], x1 = sims[t+1,m,1], y1 =$
$sims[t+1,m,2])$
$\}$
$\}$

lo que da el grafico:




## 0.6. Bibliografía

[1] http://www.stat.columbia.edu/ gelman/book/solutions2.pdf
[2] http://www.stat.columbia.edu/ gelman/book/solutions3.pdf
[3] Apéndice del libro de "Bayesian Data analysis"Gelman