MÉTODOS DE SIMULACIÓN

Capítulo 3. Generación de variables aleatorias

Departamento de Inteligencia Artificial

ÍNDICE

1. Generación de distribuciones continuas

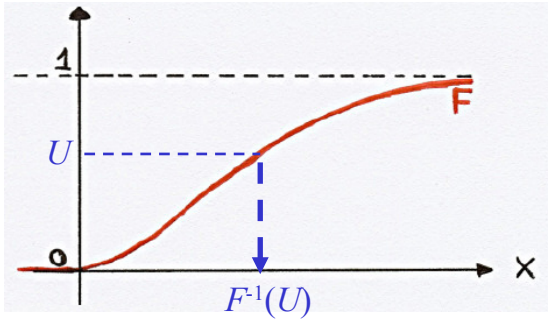
Métodos genéricos

Método de inversión

Proposición. Supongamos que la variable aleatoria X tiene función de distribución *F continua y estrictamente creciente*, siempre que $0 < F(x) < 1$. Sea U una variable aleatoria con distribución uniforme en (0,1). Entonces, la variable aleatoria $F^{-1}(U)$ tiene función de distribución F.

Para muestrear de una variable aleatoria X de la que se conoce F^{-1}, generar números U uniformes en (0,1) y hacer $X = F^{-1}(U)$:

Generar $U \sim U(0,1)$
Hacer $X = F^{-1}(U)$
Salir X



Condición mínima → conocer la forma explícita de F^{-1}. Esto ocurre para muchas distribuciones, como la uniforme, la exponencial, la de Weibull, de Cauchy,...

Tal condición no es suficiente: por ejemplo, para la distribución beta es teóricamente posible la simulación vía inversión, pero ésta puede resultar muy costosa.

En ocasiones, disponemos de una buena aproximación de F^{-1}, con lo que podemos utilizar el método por aproximación.

Ejemplo. *Generación de una variable Weibull W*(α, 1).

Su función de distribución es: $$F(x) = \begin{cases} 0 & x < 0 \\ 1 - \exp(-x^\alpha) & x \geq 0 \end{cases}$$

Hacemos $u = 1 - e^{-x^\alpha}$

luego $$e^{-x^\alpha} = 1 - u \quad \Rightarrow \quad -x^\alpha = \ln(1-u) \quad \Rightarrow$$

$$x^\alpha = -\ln(1-u) \quad \Rightarrow \quad x = (-\ln(1-u))^{1/\alpha}$$

o, lo que es lo mismo, $x = (-\ln(u))^{1/\alpha}$

Método de rechazo

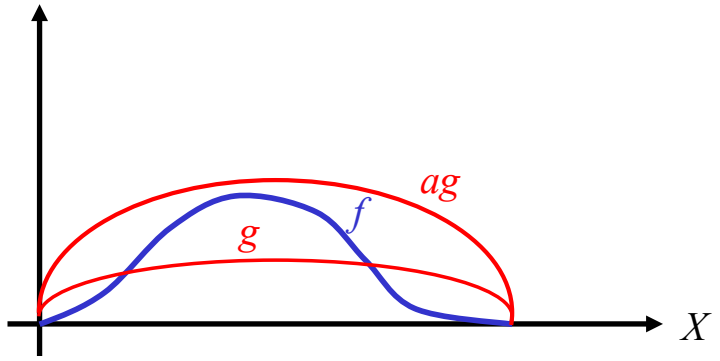
Introducido por Von Neumann (1951). *En el método de inversión es necesario conocer la fun. de distribución de la variable* de la que se quieren generar valores.

En otros casos, conocemos la función de densidad pero no la función de distribución, por ejemplo, en la *distribución normal*.

Supongamos que deseamos muestrear de una variable aleatoria X con función de densidad f. No lo sabemos hacer directamente, pero disponemos de un procedimiento para muestrear de una función de densidad g tal que $f(x) \leq ag(x)$ para todo x (con $a < \infty$).

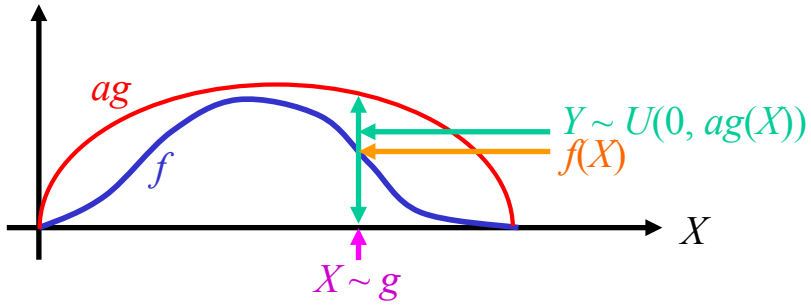
El *método de rechazo* sugiere

Hasta que $U \leq f(X)/ag(X)$
  Generar $X \sim g$
  Generar $U \sim U(0,1)$
Salir X



El método de rechazo equivale a:

- generar valor $X \sim g$
- generar valor $Y \sim U(0, ag(X))$
- aceptar X si $Y \leq f(X)$.



Cuanto más próximo esté a a 1 (siempre $a \geq 1$) más *eficiente* será el método, pues más parecidas serán f y g.

$P(X \text{ aceptado}) = 1/a$
Nº de iteraciones hasta la aceptación de un valor ~ $Ge(1/a)$
Nº medio de iteraciones hasta aceptar un valor es a

Ejemplo. *Generación de una variable Be*(3, 4).

Recordemos que para *Be*(α, β):

$$f(x) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}, \quad 0 < x < 1$$

En nuestro caso:

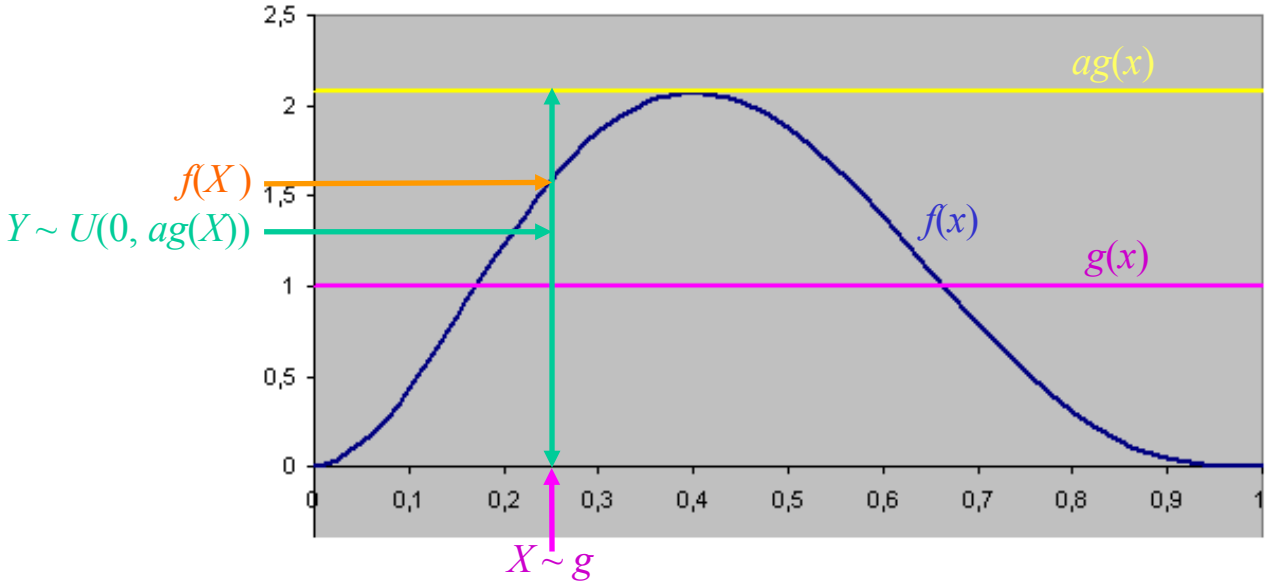
$$f(x) = \frac{\Gamma(7)}{\Gamma(3)\Gamma(4)} x^2(1-x)^3 = \frac{6!}{2!\times 3!} x^2(1-x)^3 = 60x^2(1-x)^3, \quad 0 < x < 1$$

Tomamos como g la densidad de la distribución $U(0,1)$, es decir, $g(x) = 1/(1\text{-}0) = 1$. Determinamos una constante a tal que $f \le ag$. El máximo de la función $f(x)/g(x)$ se alcanza en $x = 2/5$. Así

$$f(x)/g(x) \le 60(2/5)^2(1-2/5)^3 = \frac{1296}{625} = a$$

Por lo tanto,

$$\frac{f(x)}{ag(x)} = \frac{3125}{108} x^2(1-x)^3$$



El procedimiento de rechazo queda:

Hasta que $U_2 \leq (3125/108)\ U_1^2(1 - U_1)^3$
  Generar $U_1, U_2 \sim U(0,1)$
Salir U_1

El número medio de iteraciones hasta aceptar es $a \approx 2.07$. La eficiencia es $1/a \approx 0.48$

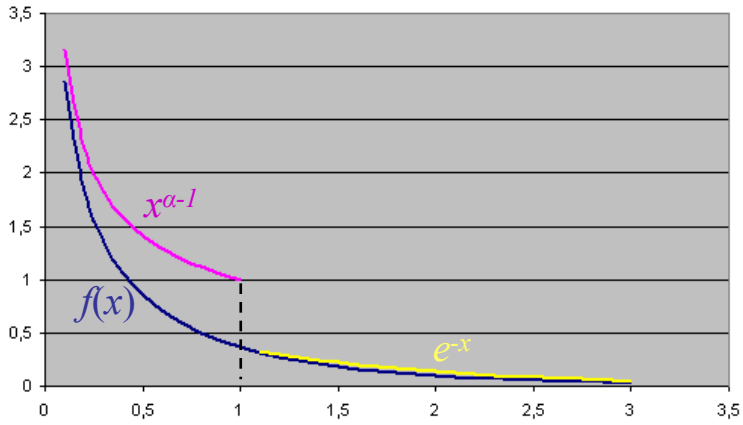
Ejemplo. *Generación de una variable Gamma*(α, 1).

Su función de densidad es: $$f(x)=\frac{x^{\alpha-1}e^{-x}}{\Gamma(\alpha)},\ 0\le x<\infty,\ \alpha>0$$

Consideramos el caso en el que α < 1. Se verifica la siguiente desigualdad:

$$x^{\alpha-1}e^{-x}\le\begin{cases}x^{\alpha-1} & \text{si } 0\le x\le 1\\ e^{-x} & \text{si } x>1\end{cases}$$

Para $\alpha = 0.5$



Podemos escribir $f(x)\le ag(x)$ mediante

$$g(x)=\begin{cases}\frac{1}{c}x^{\alpha-1} & \text{si } 0\le x\le 1\\ \frac{1}{c}e^{-x} & \text{si } x>1\end{cases}$$

y $c=\frac{1}{\alpha}+\frac{1}{e}$, para que g sea un función de densidad.

Además, $a = \dfrac{c}{\Gamma(\alpha)}$

Por tanto, $$\frac{f(x)}{ag(x)} = \begin{cases} e^{-x} & \text{si } 0 \le x \le 1 \\ x^{\alpha\text{-}1} & \text{si } x > 1 \end{cases}$$

La generación de la variable $X \sim g$ se hace de manera sencilla por el *método de inversión*.

La aplicación del *método de rechazo* queda:

Hasta que $U \le \begin{cases} e^{-X} & \text{si } 0 \le x \le 1 \\ X^{\alpha\text{-}1} & \text{si } x > 1 \end{cases}$

Generar $X \sim g$

Generar $U \sim U(0,1)$

Salir X

La eficiencia es $\dfrac{\alpha e \Gamma(\alpha)}{\alpha + e}$

Método del cociente de uniformes

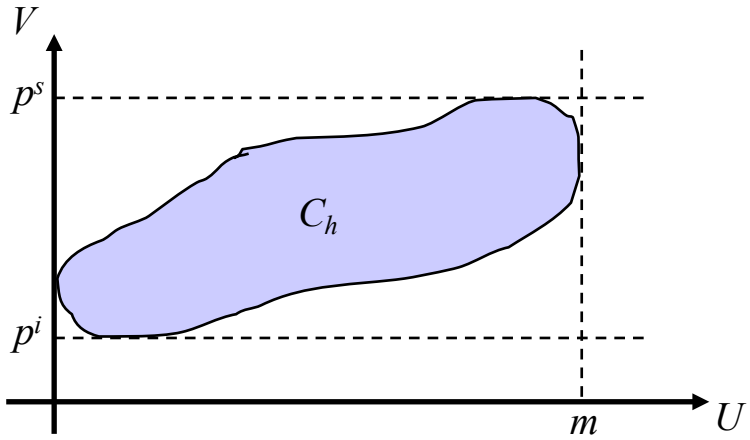
Supongamos que (U, V) se distribuye uniformemente en el disco unidad. Se demuestra que V / U sigue una *distribución de Cauchy*. Podemos entonces preguntarnos si es posible muestrear de otras distribuciones como cociente de variables distribuidas uniformemente sobre cierto subconjunto. Se tiene el siguiente resultado:

Proposición: Sea h una función no negativa con $0 < \int h < \infty$. Sea

$$C_h = \{(u, v): 0 \leq u \leq (h(v/u))^{1/2}\}$$

C_h tiene área finita. Si (U, V) se distribuye uniformemente sobre C_h, entonces $X = V/U$ tiene función de densidad $h/(\int h)$.

El resultado es especialmente útil cuando C_h está contenido en un rectángulo $[0,m] \times [p^i, p^s]$, pues podemos emplear muestreo por rechazo:



Podemos dar el siguiente algoritmo:

Hasta que $(U, V) \in C_h$
  Generar $U_1, U_2 \sim U(0,1)$
  Hacer $U = mU_1$, $V = p^i + (p^s - p^i)U_2$
Salir $X = V/U$

¿cómo calcular el rectángulo en el que está contenido C_h?

Proposición: Supongamos que las funciones $h(x)$ y $x^2h(x)$ están acotadas (en el dominio de f). Entonces $C_h \subset [0, m] \times [p^i, p^s]$, con $m = (\sup h)^{1/2}$, $p^s = (\sup\{x^2h(x) : x \geq 0\})^{1/2}$ y $p^i = -(\sup\{x^2h(x): x \leq 0\})^{1/2}$

Ejemplo. *Generación de la distribución de Cauchy* $C(0,1)$. (estándar)

Tomamos $h(x) = 1/(1+x^2)$ en $(-\infty,\infty)$. Puesto que $h(x)$ y $x^2h(x)$ son acotadas, obtenemos $m = 1$, $p^s = 1$ y $p^i = -1$.

Se tienen las equivalencias $(u,v) \in C_h \Leftrightarrow 0 \le u \le \sqrt{\dfrac{1}{1+\left(\dfrac{v}{u}\right)^2}} \Leftrightarrow 0 \le u \le \sqrt{1-v^2}$

Por tanto, el algoritmo queda

Hasta que $U \le (1 - V^2)^{1/2}$
  Generar $U_1, U_2 \sim U(0,1)$.
  Hacer $U = U_1$, $V = 2U_2$-1.
Salir $X = V/U$

En ocasiones es posible incluir C_h en conjuntos poligonales más eficientes que un rectángulo. Sin embargo, el mayor coste computacional suele estar en contrastar si $(u,v) \in C_h$, por lo que el método suele ir acompañado por el uso de *precontrastes*.

Uso de precontrastes

En el *método de rechazo*, si es sencillo obtener funciones u, v que cumplen las desigualdades $u(x) \leq f(x)/g(x) \leq v(x)$ para todo x, podemos establecer la siguiente regla: Si $aU \leq u(X)$ aceptar y salir X; si $aU > v(X)$ rechazar X. Sólo en el caso en que $u(X) \leq aU \leq v(X)$, hemos de comprobar la condición más costosa $aU \leq f(X)/g(X)$.

Para el *método del cociente de uniformes*, si es sencillo obtener regiones C_i, C_s tales que $C_i \in C_h \in C_s$ y es fácil determinar si $(u,v) \in C_i$ y $(u,v) \notin C_s$, podemos establecer la regla: Si $(U,V) \in C_i$, aceptar y salir $X = U/V$; si $(U,V) \notin C_s$, rechazar. Sólo en el caso en que $(U,V) \in C_s \backslash C_i$, hemos de comprobar la condición más costosa $(U,V) \in C_h$. La obtención de las cotas puede ser complicada, por lo que el procedimiento requiere importantes dosis de ingenio para que resulte eficiente y ventajoso.

Empleo de transformaciones

En ocasiones es posible utilizar transformaciones entre variables aleatorias, de manera que si sabemos generar de una de ellas, podemos hacerlo de la otra.

(1) *Generación de la distribución lognormal*. Supongamos que tenemos acceso a un generador de variables normales Y. Sabemos que si X es lognormal, $\log X$ es normal. Por tanto, basta hacer

Generar $Y \sim Normal$
Salir $X = e^Y$

(2) *Generación de la distribución Gamma(α, β).* Supongamos que tenemos acceso a un generador de variables $Gamma(\alpha,1)$. Sabemos que si $Y \sim Gamma(\alpha,1)$, entonces $Y/\beta \sim Gamma(\alpha, \beta)$. Por tanto, basta hacer

Generar $Y \sim Gamma(\alpha,1)$
Salir $X = Y/\beta$

ÍNDICE

1. Generación de distribuciones continuas

Métodos específicos

Distribución normal

Consideramos distintos métodos para generar de la distribución normal estándar $Y \sim N(0,1)$. Si deseásemos hacerlo de la distribución normal $X \sim N(\mu, \sigma^2)$, bastaría hacer la transformación $X = \mu + \sigma Y$.

Inversión. Se conocen numerosas aproximaciones a la función de distribución de la normal y a su inversa. Una de ellas conduce a la fórmula de inversión aproximada.

$$X = \frac{U^{0.135} - (1-U)^{0.135}}{0.1975}$$

Es más rápido que los otros métodos que describimos. En ciertas aplicaciones proporciona una aproximación suficientemente buena.

Suma de 12 uniformes. Este procedimiento se basa en el *Teorema Central del Límite* y puede verse como un ejemplo de transformación.

Por el *TCL*, si las variables U_i, $i = 1,..,n$, son *i.i.d.* $U(0,1)$, con lo que $E(U_i)=1/2$, $Var(U_i)=1/12$, la variable

$$X = \frac{\left(\sum_{i=1}^{n} U_i - \frac{n}{2}\right)}{\sqrt{\frac{n}{12}}}$$

se distribuye aproximadamente como una normal estándar, para n suficientemente grande.

Una buena aproximación se tiene ya para $n = 12$, con lo que

$$X = \left(\sum_{i=1}^{12} U_i\right) - 6$$

y el procedimiento queda

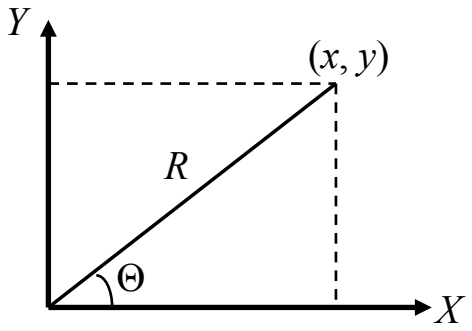
Generar $U_1,...,U_{12} \sim U(0,1)$

Hacer $X = \left(\sum_{i=1}^{12} U_i\right) - 6$

Salir X

Método Box-Muller. El método exacto para generar de la normal más conocido es el de Box y Muller (1958), que genera un par de variables (X,Y) normales estándar e independientes.

La función de densidad de (X,Y) es $f(x,y) = \frac{1}{2\pi} e^{\left(-\frac{x^2+y^2}{2}\right)}$



Sean (R,Θ) las *coordenadas polares* de (X,Y), esto es,

$$R^2 = X^2 + Y^2 \qquad \tan\Theta = Y/X$$

que tienen función de densidad $g(r,\Theta) = \frac{1}{2\pi}\frac{1}{2} e^{\left(-\frac{r^2}{2}\right)} = g_1(\Theta)g_2(r)$ en $(0, 2\pi) \times (0, \infty)$,

con

$g_1(\Theta)$ tiene una densidad uniforme $U(0, 2\pi)$, y

$g_2(r)$ es decir una densidad tal que R^2 es $Exp(1/2) \equiv \chi^2_2$.

R se genera fácilmente por inversión pues

$$F(r^2) = 1 - e^{-\frac{r^2}{2}}$$

Así, si $U_1 \sim U(0,1)$, se tiene $R = (-2\ln(1-U_1))^{1/2}$, que tiene la misma distribución que

$$R = (-2\ln U_1)^{1/2} .$$

Se tiene entonces el algoritmo

Generar $U_1, U_2 \sim U(0,1)$
Hacer $R = (-2\ln U_1)^{1/2}$, $\Theta = 2\pi U_2$
Hacer $X = R \cos \Theta = (-2\ln U_1)^{1/2} \cos 2\pi U_2$
Hacer $Y = R \text{ sen } \Theta = (-2\ln U_1)^{1/2} \text{ sen } 2\pi U_2$
Salir X, Y

Variante de Marsaglia. Marsaglia introduce su variante polar del método de Box-Muller, que incorpora el método de rechazo para evitar las operaciones trigonométricas de senos y cosenos, poco eficientes. Descripción en Ríos Insua et al. (2008).

Otros métodos para generar de la normal estándar:

- *GRAND* (Brent, 1974)
- *Ahrens-Dieter Table-Free method* (Ahrens y Dieter, 1988)
- *The recursive method* (Wallace, 1996)
- *Monty Python method* (Marsaglia y Tsang, 1998)
- *Ziggurat method* (Marsaglia y Tsang, 2000)

En Thomas et al. (2007) se realizar un estudio tanto del rendimiento como de la precisión de estos y otros métodos.

Distribución exponencial

Sabemos que si Y se muestrea de una distribución $Exp(1)$, entonces $X = Y/\lambda$ tiene distribución $Exp(\lambda)$. Por ello, suponemos en lo que sigue que $\lambda = 1$.

Inversión. El método de inversión es sencillo, pues la función de distribución es

$$F(x) = 1 - e^{-x}$$

Se tiene $X = -\ln(1-U)$, que es lo mismo que $X = -\ln U$. El algoritmo queda:

Generar $U \sim U(0,1)$
Hacer $X = -\ln U$
Salir X

Suele sugerirse este método, por su sencillez, pero puede resultar lento si la operación logaritmo neperiano no está implementada en hardware. En tal caso, un método competitivo puede ser el de cociente de uniforme con contrastes.

Distribución Gamma y Erlang

Distribución *Gamma*(α,β): Puesto que β es un parámetro de escala nos basta estudiar el caso $X \sim Gamma(\alpha,1)$ y hacer $Y = X/\beta$ cuando $\beta \neq 1$.

Cuando α es entero, tenemos un distribución de *Erlang* α, que es la suma de α variables independientes *Exp*(1), con lo que se sigue el algoritmo

$X = 0$
Desde $i = 1$ hasta α
    Generar $Y \sim Exp(1)$
    Hacer $X = X + Y$
Salir X

Cuando α es muy grande (>50), es mejor utilizar una aproximación normal basada en el *Teorema Central del Límite*. Esto se aplica también al caso en que α no es entero.

Cuando $\alpha < 1$, podemos utilizar el *algoritmo de rechazo*.

Para el caso $\alpha > 1$, el algoritmo más utilizado es una *modificación del de cociente de uniformes* con $h(x) = x^{\alpha-1}e^{-x}$ propuesto en Cheng y Feast (1979). En Ríos Insua et al. (2008) está disponible una descripción del mismo.

Distribución de Cauchy

Consideremos ahora el caso de generación de una variable aleatoria X con distribución de Cauchy $C(\alpha,\beta)$ utilizando el *método de inversión*.

La función de distribución de X es

$$F(x) = \frac{1}{2} + \frac{1}{\pi}\arctan\left(\frac{x-\alpha}{\beta}\right)$$

Por inversión, se obtiene

$$X = F^{-1}(U) = \alpha + \beta\tan\left[\pi\left(u - \frac{1}{2}\right)\right]$$

El algoritmo es

Generar $U \sim U(0,1)$
Hacer $X = \alpha + \beta\tan(\pi(U\text{-}0.5))$
Salir X

Distribución Beta

Si X_1 y X_2 son dos variables aleatorias independientes con distribuciones *Gamma*(α,1) y *Gamma* (β,1), respectivamente, entonces la variable $X=X_1/(X_1+X_2)$ sigue una distribución Be(α, β). El algoritmo es

Generar $X_1 \sim Gamma(\alpha, 1)$, $X_2 \sim Gamma(\beta, 1)$
Hacer $X = X_1/(X_1+X_2)$
Salir X

Distribución Weibull

Si Y es una variable aleatoria con distribución *Exp*(β), entonces la variable $X = Y^{1/\alpha}$ tiene una distribución $W(\alpha, \beta)$. El algoritmo es

Generar $Y \sim Exp(\beta)$
Hacer $X = Y^{1/\alpha}$
Salir X

Distribución χ_n^2

Si las variables $Z_1,...,Z_n$ son independientes con distribución $N(0,1)$, entonces $X = \sum_{i=1}^{n} Z_i^2$ tiene distribución χ_n^2. Esto sugiere generar n normales estándar, elevarlas al cuadrado y sumarlas para generar de la χ_n^2.

Distribución *t* de Student

Si Z es una variable con distribución $N(0,1)$ e Y una variable independiente de Z con distribución χ_n^2, entonces $$X = \frac{Z}{\sqrt{Y/n}}$$ tiene distribución t de Student con n grados de libertad.
Cuando n es grande (>30) utilizamos la aproximación a la normal.

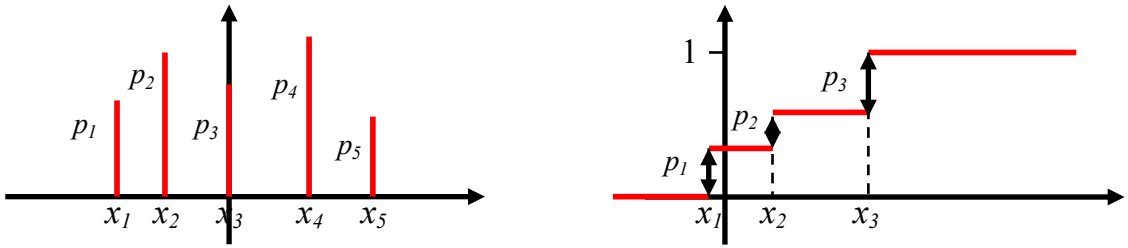
Distribución *F* de Fisher-Snedecor

Si Y_1 es una variable aleatoria con distribución χ_{n1}^2 e Y_2 es χ_{n2}^2 y ambas son independientes, entonces $$X = \frac{Y_1/n_1}{Y_2/n_2}$$ sigue una distribución F con n_1 y n_2 grados de libertad.

ÍNDICE

2. Generación de distribuciones discretas

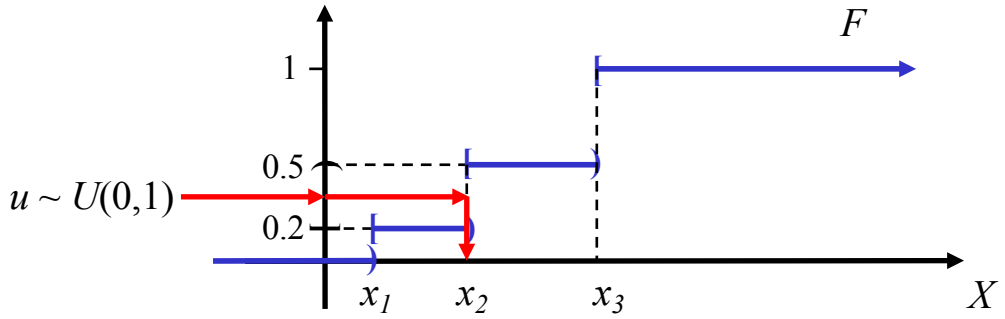
Denotaremos como *función de masa* de una variable discreta X mediante $\{p_i\}$ donde $p_i = P(X = i)$, $i = 1,2,...$ con $p_i \geq 0$ y $\sum_i p_i = 1$, y su *función de distribución* con $F_i = P(X \leq i)$.



Métodos genéricos

Método de inversión

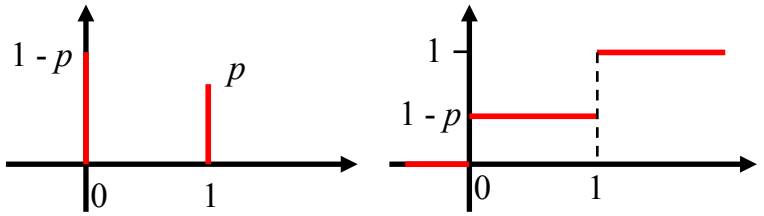
Proposición. Sea $\overline{F}(u) = \min\{x: F(x) \geq u\}$. Si U es una variable aleatoria con distribución $U(0,1)$, entonces la variable $X = \overline{F}(U)$ tiene función de distribución F.



Ejemplo. *Prueba de Bernoulli.*

Sea una variable aleatoria $X \sim B(1, p)$. Su función de distribución es

$$F(x) = (1\text{-}p) + pI_{(x\geq 1)}$$



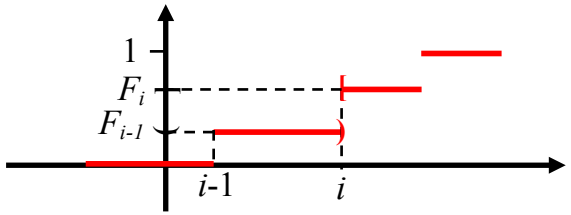
Así, $\overline{F}(U) = I_{(u\geq 1-p)} = \begin{cases} 1 \text{ si } u \geq 1-p \\ 0 \text{ si } u < 1-p \end{cases}$

siendo el algoritmo:

Generar $U \sim U(0,1)$
Si $U \geq 1 - p$, salir $X = 1$
En caso contrario, salir $X = 0$

Ejemplo. *Generación de una variable aleatoria discreta general.*

Resulta $\overline{F}(u) = i$, con $F_{i-1} < u \leq F_i$, por lo que el método de inversión es equivalente a buscar el índice i adecuado en la lista de números (F_i).



En efecto, a la variable X se le asignará el valor i si

$$F_{i-1} = \sum_{j=1}^{i-1} p_j \leq U < \sum_{j=1}^{i} p_j = F_i$$

Puesto que $P(a \leq U \leq b) = b - a$, para $0 \leq a < b \leq 1$, se tiene que

$$P(X = i) = P\left(\sum_{j=1}^{i-1} p_j \leq U < \sum_{j=1}^{i} p_j\right) = p_i$$

Formalmente, tenemos

Generar $U \sim U(0,1)$. Hacer $i = 1$
Mientras $F_i \leq U$, hacer $i = i + 1$
Salir $X = i$

El número esperado de comparaciones es $E(X)$, puesto que hacemos i comparaciones si $X = i$.

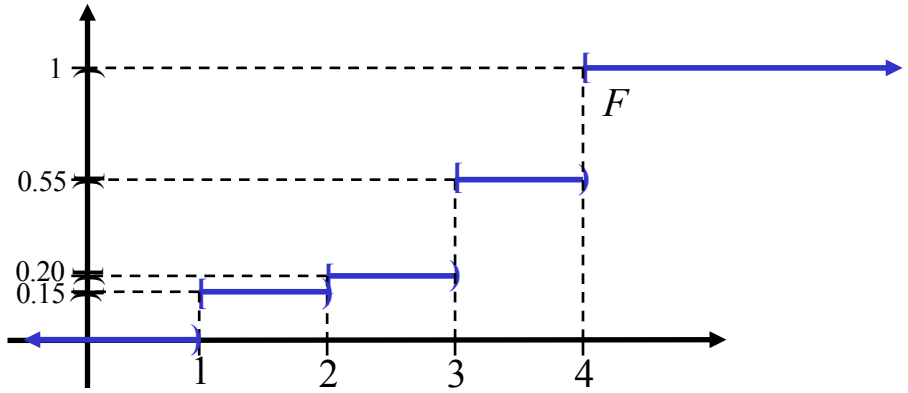
El tiempo que utiliza el procedimiento anterior para generar una variable aleatoria discreta es proporcional al número de intervalos en los que hay que hacer la búsqueda. Por este motivo, se puede acelerar el método si se ordenan los p_i en orden decreciente pues, además, reducimos $E(X)$ lo más posible.

Podemos también recuperar la distribución inicial, a costa de cierto tiempo de inicialización y gasto de memoria.

Ejemplo. *Generación de una variable discreta finita.*

Supongamos que deseamos simular una variable aleatoria X con función de masa $p_i = P(X = i)$ y *función de distribución* F_i, como aparecen en la tabla

i	1	2	3	4
p_i	0.15	0.05	0.35	0.45
F_i	0.15	0.20	0.55	1.0



El procedimiento inicial sería

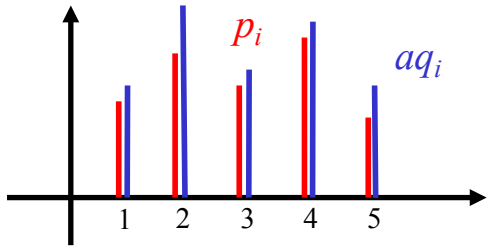
Generar $U \sim U(0,1)$
Si $U < 0.15$, salir $X = 1$
Si $U < 0.20$, salir $X = 2$
Si $U < 0.55$, salir $X = 3$
Salir $X = 4$

Si ordenamos los p_i decrecientemente tenemos el procedimiento más eficiente

Generar $U \sim U(0,1)$
Si $U < 0.45$, salir $X = 4$
Si $U < 0.80$, salir $X = 3$
Si $U < 0.95$, salir $X = 1$
Salir $X = 2$

Método de rechazo

La idea es la misma que en el caso continuo. Deseamos simular de una variable discreta X con *función de masa* $\{p_i,\ i \geq 0\}$. Disponemos de un método eficiente para generar de una variable discreta Y con función de masa $\{q_i,\ i \geq 0\}$.



Para generar de X, primero se genera de Y y se acepta el valor generado con probabilidad proporcional a p_i/q_i.

Sea a una constante tal que $p_i/q_i \leq a$, para todo i. El método de rechazo para generar X viene dado por el siguiente algoritmo

Hasta que $U \leq p_Y / aq_Y$
  Generar $Y \sim \{q_i,\ i \geq 0\}$
  Generar $U \sim U(0,1)$
Salir $X = Y$

Como en el caso continuo se demuestra que la variable aleatoria X tiene función de masa $P(X = i) = p_i$. En cada iteración se acepta el valor (de forma independiente) con probabilidad $1/a$ y el número esperado de iteraciones hasta aceptar un valor es a.

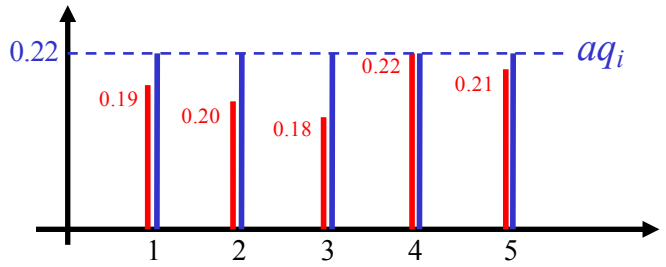
Cuanto más próximo esté el valor de a a 1 más eficiente será el algoritmo.

Ejemplo. *Generación de una variable aleatoria discreta.*

Deseamos simular la variable aleatoria discreta X cuya *función de masa* es

i	1	2	3	4	5
p_i	0.19	0.20	0.18	0.22	0.21

Tomando como variable Y la uniforme discreta en 1,...,5: $q_i = 1/5$ para todo i. Para esta variable, tomamos $a = \max(p_i/q_i) = 1.1$, en cuyo caso $aq_i = 1.1/5 = 0.22$, para todo i.



El algoritmo queda

Hasta que $U_2 \leq p_Y/0.22$
  Generar $U_1, U_2 \sim U(0,1)$
  Hacer $Y = \text{ent}(5U_1) + 1$
Salir $X = Y$

Este algoritmo necesita en promedio 1.1 iteraciones hasta aceptar un valor para la variable X.

La probabilidad de aceptación es $1/a = 1/1.1 = 0.909$.

Método de composición

Supongamos que disponemos de procedimientos eficientes para muestrear de las variables aleatorias X_1 y X_2, con funciones de masa $\{p_i\}$ y $\{q_i\}$, respectivamente; deseamos generar una variable aleatoria X con función de masa

$$P(X = i) = \alpha p_i + (1 - \alpha) q_i \quad \text{con } \alpha \in (0,1).$$

Para ello, podemos muestrear de la variable X, generando primero un número $U \sim U(0,1)$ y, después, generando de X_1 si $U < \alpha$ y de X_2 si $U > \alpha$.

Ejemplo. *Generación de una variable aleatoria discreta.*

Deseamos simular la variable aleatoria discreta X cuya *función de masa* es

i	0	1	2	3	4	5
p_i	0.12	0.12	0.12	0.12	0.32	0.20

La función de masa de X se puede escribir como composición de las funciones de masa de dos variables uniformes X_1 y X_2, dadas en la siguiente tabla:

i	0	1	2	3	4	5
p_i^1	0.20	0.20	0.20	0.20	0.20	0.0
p_i^2	0.0	0.0	0.0	0.0	0.50	0.50

con $\alpha = 0.6$, es decir, $p_i = 0.6p_i^1 + 0.4p_i^2$.

El procedimiento queda entonces

Generar $U_1, U_2 \sim U(0,1)$
Si $U_1 < 0.6$, $X = \text{ent}(5U_2)$
En caso contrario, $X = \text{ent}(2U_2)+4$
Salir X

Método de alias

El método de alias (Walker, 1977) permite la generación eficiente de variables discretas *con soporte finito*. Incluye una *fase de preprocesamiento*, pero luego es rápido y de implementación sencilla.

Supongamos que deseamos generar de una variable X con *función de masa* $P = \{p_i, i=1,...,n\}$. El método de alias es, de hecho, un método de composición basado en la representación de P como una mixtura

$$P = \frac{1}{n-1}\sum_{k=1}^{n-1} Q^{(k)}$$

con $Q^{(k)}$ una distribución concentrada sobre, a lo sumo, dos puntos de $\{1,...,n\}$. La demostración de esta descomposición se basa en el siguiente resultado:

Lema. Sea $P = \{p_i, i = 1,...,n\}$ una función de masa de probabilidad. Entonces:

a) existe i, $1 \leq i \leq n$, tal que $p_i < 1/n\text{-}1$
b) para tal i, existe $j \neq i$ tal que $p_i + p_j \geq 1/n\text{-}1$

Ejemplo. *Generación de una variable aleatoria discreta.*

Deseamos simular la variable aleatoria discreta X cuya *función de masa* es

i	1	2	3
p_i	3/8	1/2	1/8

Deseamos hacer $P = \frac{1}{3-1}\sum_{k=1}^{3-1} Q^{(k)} = \frac{1}{2}Q^{(1)} + \frac{1}{2}Q^{(2)}$

con $Q^{(1)} = (q_1{}^{(1)}, q_2{}^{(1)}, q_3{}^{(1)})$, $Q^{(2)} = (q_1{}^{(2)}, q_2{}^{(2)}, q_3{}^{(2)})$, bipuntuales a lo sumo.

Tenemos $$p_l = \frac{1}{2}q_l^{(1)} + \frac{1}{2}q_l^{(2)} \quad l = 1,2,3$$

y alguno de los $q_l{}^{(i)}$ nulos.

Aplicando el lema: a) $p_3 < 1/2$, b) $p_3 + p_2 > 1/2$, con lo que $i = 3, j = 2$.

Escogemos $Q^{(1)}$ de manera que se concentre en 2 y 3, y para que 3 tome toda su masa de P a partir de $Q^{(1)}$, es decir,

i	1	2	3
p_i	3/8	1/2	1/8
$q_i^{(1)}$	0	$q_2^{(1)}$	1 - $q_2^{(1)}$
$q_i^{(2)}$	$q_1^{(2)}$	$q_2^{(2)}$	$q_3^{(2)}$

De $p_l = \frac{1}{2} q_l^{(1)} + \frac{1}{2} q_l^{(2)}$ $l = 1,2,3,$ para $l = 1$ tengo $3/8 = (1/2) q_1^{(2)} \rightarrow q_1^{(2)} = 3/4$.

i	1	2	3
p_i	3/8	1/2	1/8
$q_i^{(1)}$	0	$q_2^{(1)}$	1 - $q_2^{(1)}$
$q_i^{(2)}$	3/4	$q_2^{(2)}$	$q_3^{(2)}$

Sabiendo que las funciones de masa auxiliares son a lo sumo bipuntuales, $q_2^{(2)} = 0$ ó $q_3^{(2)} = 0$. Hacemos $q_3^{(2)} = 0$, quedándonos $q_3^{(2)} = 1/4$ (al ser $q_i^{(2)}$ una función de masa).

i	1	2	3
p_i	3/8	1/2	1/8
$q_i^{(1)}$	0	$q_2^{(1)}$	1 - $q_2^{(1)}$
$q_i^{(2)}$	3/4	1/4	0

De $p_l = \frac{1}{2} q_l^{(1)} + \frac{1}{2} q_l^{(2)} \quad l = 1,2,3,$ obtenemos los valores restantes.

i	1	2	3
p_i	3/8	1/2	1/8
$q_i^{(1)}$	0	3/4	1/4
$q_i^{(2)}$	3/4	1/4	0

Métodos alternativos para generación de variables discretas (Marsaglia, 2004):

- *The square histogram method*
- *Condensed Table lookup*

ÍNDICE

2. Generación de distribuciones discretas

Métodos específicos

Distribución binomial

Una variable aleatoria X con *distribución binomial* $B(n,p)$ puede interpretarse como la suma de n pruebas independientes de Bernoulli. Así, se tiene una forma de muestreo generando n números aleatorios $U_1,...,U_n \sim U(0,1)$ y haciendo X igual al número de ellos que son menores que p.

Si consideramos la i-ésima prueba como éxito si $U_i < p$ y que la probabilidad de ese suceso es igual a p, es claro que se obtendrá una variable binomial con parámetros n y p. El algoritmo es

Hacer $X = 0$
Repetir n veces
    Generar $U \sim U(0,1)$
    Si $U < p$, hacer $X = X+1$
Salir X

Este método requiere n números aleatorios y n comparaciones. Por ello, suele preferirse el siguiente **método de inversión basado en la fórmula recursiva** :

$$P(X = i+1) = \frac{(n-i)p}{(i+1)(1-p)} P(X = i)$$

En el siguiente algoritmo denotamos $P = P(X = i)$ y $F = P(X \leq i)$.

Generar $U \sim U(0,1)$
Hacer $i = 0$, $P = F = (1 - p)^n$
Hasta que $U < F$
Hacer $P = \frac{(n-i)p}{(i+1)(1-p)} P$, $F = F + P$
$i = i + 1$
Salir $X = i$

El número de comparaciones es uno más que el valor de X, por lo que, en promedio, son necesarias $1 + np$ comparaciones para generar la variable X.

Cuando n es grande, el método es poco eficiente y suele preferirse un método general como el de alias.

Distribución de Poisson

En la *distribución de Poisson* $P(\lambda)$, para λ pequeño, puede utilizarse el **método de inversión**, según la fórmula recursiva

$$P(X = i+1) = \frac{\lambda}{i+1} P(X = i)$$

Con la misma notación F, P de antes, queda

Generar $U \sim U(0,1)$
Hacer $i = 0, F = P = e^{-\lambda}$
Hasta que $U < F$
  Hacer $P = \frac{\lambda}{i+1} P, \quad F = F + P$
  $i = i + 1$
Salir $X = i$

El número de comparaciones será una más que el valor de Poisson generado. En promedio, serán necesarias $1+\lambda$ comparaciones.
Otro algoritmo interesante se basa en la **relación que existe entre la distribución de Poisson y la exponencial**.

Distribución Geométrica

La distribución geométrica *Ge*(*p*) es conocida a veces como *exponencial discreta*, pues puede generarse muestreando por discretización de una exponencial. Supongamos que $Y \sim Exp(\lambda)$. Sea $X = \text{ent}(Y)$. Entonces,

$$P(X = r) = P(r \leq Y < r+1) = \int_r^{r+1} \lambda e^{-\lambda s} ds = e^{-\lambda r} - e^{-\lambda(r+1)} = (e^{-\lambda})^r (1 - e^{-\lambda})$$

para $r = 0,1,....$, que es la *función de masa* de una distribución $Ge(p = 1 - e^{-\lambda})$.

Tomando $\lambda = -\ln(1-p)$ se observa que la expresión anterior es idéntica a la función de probabilidad de una distribución geométrica de parámetro *p*.

En definitiva, para generar de una variable geométrica *Ge*(*p*) primero generamos de una variable exponencial *Exp*(-ln(1-*p*)) y redondeamos al valor entero.

En Ríos Insua et al. (2008) se decriben algoritmos para la generación de valores aleatorios de las distribuciones **binomial negativa** e **hipergeométrica**.

ÍNDICE

3. Distribuciones multivariantes

Métodos generales

Representaremos la distribución multivariante mediante $X = (X_1, X_2, ..., X_p)$.

Distribuciones independientes

El caso más sencillo es el de independencia de las *distribuciones marginales*, esto es,

$$F(x) = \prod_{i=1}^{p} F_i(x_i)$$

con $x = (x_1, ..., x_p)$.

Basta entonces generar de cada una de las componentes X_i, que son univariantes, y salir con $X = (X_1, X_2, ..., X_p)$.

Distribuciones dependientes con condicionadas disponibles

Utilizando la descomposición

$$F(x) = F_1(x_1)F_2(x_2|x_1)\ldots F_p(x_p|x_1,\ldots,x_{p-1}),$$

si disponemos de las distribuciones

$$X_i \,|\, x_1,\ldots,x_{i-1}, \quad i = 1,\ldots,n$$

en el sentido de que son de generación eficiente, tenemos entonces

Desde $i = 1$ hasta p
  Generar $x_i \sim X_i \,|\, x_1,\ldots x_{i-1}$
Salir $x = (x_1,\ldots,x_p)$

En muchas ocasiones tales distribuciones condicionadas no estarán disponibles. Una alternativa es utilizar los *métodos basados en cadenas de Markov*.

Distribuciones discretas

En la práctica, suelen tener soporte muy grande, con lo que plantean problemas considerables a los métodos estándar.

La *búsqueda directa* suele ser muy lenta, pero los métodos de búsqueda indexada, alias,... funcionan bien, siempre y cuando se disponga de memoria suficiente.

3. Distribuciones multivariantes

Distribución normal multivariante

Sea $X = (X_1,...,X_p)$ una variable aleatoria con distribución normal multivariante $N(\mu,\Sigma)$ donde $\mu = (\mu_1,...,\mu_p)$ es el vector de medias y Σ es la matriz de covarianzas. Consideramos el siguiente *método basado en la descomposición de Cholesky*, por ser sencillo y eficiente.

Supongamos que $\Sigma = LL^t$ para alguna matriz L. Entonces, si $Z = (Z_1,...,Z_p)$ son normales estándar independientes, la variable $X = \mu + LZ$ tiene distribución $N(\mu,\Sigma)$.

Generar $Z_1,...,Z_p \sim N(0,1)$
Hacer $X = \mu + LZ$

El problema está en encontrar L, que siempre existe. Una posibilidad es utilizar la descomposición de Cholesky L de Σ, que es la única matriz triangular inferior tal que $LL^t = \Sigma$, con lo que $X = \mu + LZ$ se calcula eficientemente.

Nash (1979) proporciona información sobre la descomposición de Cholesky.

En Ríos Insua et al. (2008) se decriben algoritmos para la generación de valores aleatorios de las distribuciones **de Wishart**, **de Dirichlet** y **multinomial**.

ÍNDICE

4. Procesos estocásticos

Cadenas de Markov en tiempo discreto

Deseamos generar de una cadena de Markov con *espacio de estados* S y *matriz de transición* $P = (p_{ij})$, donde $p_{ij} = P(X_{n+1} = j \mid X_n = i)$. La forma obvia de simular la transición $(n+1)$-ésima, conocida X_n, es

$$\text{Generar } X_{n+1} \sim \{p_{x_n j} : j \in S\}$$

Puede ser interesante introducir *tablas de alias* para las filas de P, al menos para los estados más visitados, si S es grande. Una posibilidad es construir las tablas en la primera visita a S. Un posible inconveniente de esta aproximación es que puede ocurrir $X_n = X_{n+1}$, con lo que se pierde cierto esfuerzo computacional.

Alternativamente podemos simular T_n, el número de iteraciones hasta el siguiente cambio de es-tado y, después, el nuevo estado X_{n+Tn}. Si $X_n = s$, T_n sigue una distribución geomé-trica de parámetro p_{ss} y X_{n+Tn} tendrá distribución discreta con *función de masa* $\{p_{sj}/(1 - p_{ss}) : j \in S \setminus \{s\}\}$.

Si deseamos muestrear N transiciones de la cadena haremos, supuesto $X_0 = i_0$,

Hacer $t = 0$, $X_0 = i_0$
Mientras $t < N$
  Generar $h \sim Ge(p_{x_t x_t})$
  Generar $X_{t+h} \sim \{ p_{x_t j}/(1- p_{x_t x_t}): j \in S \setminus \{x_t\}$
  Hacer $t = t + h$

Como antes, pueden usarse tablas de alias para las correspondientes distribuciones discretas.

Cadenas de Markov en tiempo continuo

La simulación asíncrona de *cadenas de Markov en tiempo* continuo es sencilla. Recordemos que una cadena de Markov en tiempo continuo viene caracterizada por los parámetros v_i de las distribuciones exponenciales de *tiempo de permanencia* en el estado i y la matriz de transición P, con $p_{ii} = 0$, $\sum_{i \neq j} p_{ij} = 1$.

Sea P_i la distribución de la fila i-ésima. Entonces, supuesto que $X_0 = i_0$ y deseamos simular hasta el instante T, podemos utilizar el algoritmo

Hacer $t = 0, X_0 = i_0, j = 0$
Mientras $t < T$
  Generar $t_j \sim Exp(v_{X_j})$
  Hacer $t = t + t_j$
  Hacer $j = j + 1$
  Generar $X_j \sim P_{X_{j-1}}$

De nuevo, puede hacerse uso de *tablas alias* en la generación de X_j.

ÍNDICE

5. Métodos basados en cadenas de Markov

Constituyen un campo en el que la investigación es mucho más activa y dinámica y está alcanzando especial relevancia en numerosas áreas de aplicación, lo que ha permitido revolucionar áreas como la Estadística Bayesiana o la Inteligencia Artificial.

La idea básica de los *MCM* es muy sencilla: deseamos generar una muestra de una distribución $\pi(x)$ con $x \subset X \in R^n$, pero no lo podemos hacer directamente. Sin embargo, podemos construir un *proceso de Markov* $p(\cdot|\cdot)$ cuyo espacio de estados es X, del que es sencillo muestrear, y cuya distribución de equilibrio es $\pi(x)$. Si dejamos correr el proceso un periodo suficientemente largo, muestrearemos aproximadamente de π.

Podemos afirmar que gracias a ellos es posible muestrear de forma eficiente de casi cualquier distribución relevante (siendo un contexto típico el de las distribuciones multivariantes que no sean estándar)

Tenemos entonces la siguiente estrategia para generar una muestra aproximada de de tamaño N.

$i = 0$, escoger X_0 arbitrariamente
Hasta que se juzgue convergencia
  Generar $X_{i+1} \sim p(\cdot \mid X_i)$
  Hacer $i = i + 1$
Desde $j = 1$ hasta N
  Generar $X_{i+j} \sim p(\cdot \mid X_{i+j-1})$
  Salir X_{i+j}
  Hacer $j = j + 1$

El problema que se plantea es cómo construir un proceso de Markov con cierta distribución de interés.

Tienen un origen relativamente antiguo (Metropolis *et al.*, 1953), aunque su popularización se debe a Gelfand y Smith (1990).

El muestreador de Gibbs

Especial utilizado en Inferencia Bayesiana. Su nombre más correcto sería *muestreador de sustitución* (Schervish, 1995), pero en la primera exposición (Geman y Geman, 1984) intervenía la distribución de Gibbs, de ahí su nombre.

Como motivación consideramos el siguiente ejemplo sencillo debido a Casella y George (1992). Supongamos que (X,Y) son variables de Bernoulli con distribución conjunta

X	Y	$P(X,Y)$
0	0	p_1
1	0	p_2
0	1	p_3
1	1	p_4

con $p_i > 0$, $\sum p_i = 1$.

La marginal de X es una distribución de Bernoulli con probabilidad de éxito p_2+p_4, es decir, $P(X=1) = p_2+p_4$. Las distribuciones de $X|Y=y$, $Y|X=x$ son también de fácil cálculo. Por ejemplo, la distribución de $X|Y=1$ es de Bernoulli con probabilidad de éxito $p_4/(p_3+p_4)$, es decir $P(X=1|Y=1) = p_4/(p_3+p_4)$.

De hecho, todas las distribuciones condicionadas pueden expresarse mediante dos matrices

$$A_{yx} = \begin{pmatrix} P(Y=0\,|\,X=0) & P(Y=1\,|\,X=0) \\ P(Y=0\,|\,X=1) & P(Y=1\,|\,X=1) \end{pmatrix} = \begin{pmatrix} \dfrac{p_1}{p_1+p_3} & \dfrac{p_3}{p_1+p_3} \\ \dfrac{p_2}{p_2+p_4} & \dfrac{p_4}{p_2+p_4} \end{pmatrix}$$

$$A_{xy} = \begin{pmatrix} \dfrac{p_1}{p_1+p_2} & \dfrac{p_2}{p_1+p_2} \\ \dfrac{p_3}{p_3+p_4} & \dfrac{p_4}{p_3+p_4} \end{pmatrix}$$

Consideramos el siguiente esquema iterativo

Escoger $Y_0 = y_0, j = 1$
Repetir
  Generar $X_j \sim X \mid Y = Y_{j-1}$
  Generar $Y_j \sim Y \mid X = X_j$
$j = j + 1$

La sucesión $\{X_n\}$ define una *cadena de Markov con matriz de transición*

$$A = A_{yx} A_{xy} .$$

Como las probabilidades p_i son positivas, esta cadena es *ergódica* y tiene *distribución límite*, que es la marginal de X, con lo cual

$$X_n \xrightarrow{d} X$$

Análogamente, $Y_n \xrightarrow{d} Y \quad (X_n, Y_n) \xrightarrow{d} (X,Y)$

El procedimiento descrito se denomina **muestreador de Gibbs**. Nos da en este caso una cadena de Markov con la distribución límite deseada, y es, de hecho, bastante general.

En efecto, supongamos que deseamos muestrear de una variable aleatoria p-variante $X = (X_1, X_2,..., X_p)$ con distribución Π. No somos capaces de hacerlo, pero conocemos las densidades condicionadas de $X_s|X_r$, $r \neq s$, con $X = (X_1,..., X_{s-1}, X_{s+1}, ..., X_p)$, que designamos $\pi(x_s|x_r, r \neq s)$.

Escoger $X_1^0, X_2^0, ..., X_p^0, \; j = 1$

Repetir

$$\begin{aligned}
&\text{Generar } X_1^j \sim X_1|X_2^{j-1}, ..., X_p^{j-1} \\
&\text{Generar } X_2^j \sim X_2|X_1^j, X_3^{j-1}, ..., X_p^{j-1} \\
&\quad ... \\
&\text{Generar } X_p^j \sim X_p|X_1^j, X_2^j, ..., X_{p-1}^j
\end{aligned}$$

$j=j+1$

Observemos que $X_n = (X_1^n, X_2^n, ..., X_p^n)$ define una cadena de Markov con transición

$$p_G(x_n, x_{n+1}) = \prod_{i=1}^{p} \pi(x_i^{n+1}|x_j^n, \; j > i; \; x_j^{n+1}, \; j < i)$$

Bajo condiciones suficientemente generales, ver Tierney (1994) y Roberts y Smith (1994), se obtiene la correcta definición y convergencia del muestreador.

Ejemplo: Supongamos que queremos muestrear de la densidad

$$\pi(x_1, x_2) = \frac{1}{\pi} e^{-x_1(1+x_2^2)} \qquad \text{para} \quad (x_1, x_2) \in (0, \infty) \times (-\infty, \infty).$$

Tenemos que $\pi(x_1|x_2) = \dfrac{\pi(x_1, x_2)}{\pi(x_2)} \propto \pi(x_1, x_2) \propto e^{-x_1(1+x_2^2)}$

$$\pi(x_2|x_1) \propto \pi(x_1, x_2) \propto e^{-x_1 x_2^2},$$

con lo que $X_1|X_2 = x_2 \sim \mathcal{E}xp(1 + x_2^2)$

$$X_2|X_1 = x_1 \sim \mathcal{N}\left(0, \sigma^2 = \frac{1}{2x_1}\right)$$

y el muestreador queda:

Escoger un valor inicial X_2^0, j=1

Repetir

Generar $X_1^j \sim \mathcal{E}xp(1 + (X_2^{j-1})^2)$

Generar $X_2^j \sim \mathcal{N}\left(0, \frac{1}{2X_1^j}\right)$

j=j+1

Algoritmo de Metropolis-Hastings (*MH*)

El muestreador de Gibbs exige disponer de algoritmos eficientes para generar de las distribuciones marginales, condicionadas por las restantes variables, lo que no es siempre posible conseguir.

El algoritmo de **Metropolis-Hastings** (MH) requiere menos estructura en el problema, aunque típicamente, pero no siempre, será más lento. Sólo necesitamos poder:

•evaluar puntualmente la distribución objetivo, salvo una constante, y

•generar observaciones candidatas de una distribución de prueba $q(\cdot, \cdot)$, que satisfaga ciertas condiciones técnicas.

Entonces generamos una observación candidata y actualizamos el estado según ciertas probabilidades de rechazo de los candidatos.

Concretamente, se proporcionan transiciones de X_n a X_{n+1} como sigue: Sea $q(x,y)$ una función de transición de probabilidades; entonces, supuesto $X_n=x$, la transición es

Generar $y \sim q(x,Y)$

Hacer $\alpha(x,y) = \min\left\{\dfrac{\pi(y)q(y,x)}{\pi(x)q(x,y)},1\right\}$

Generar $u \sim U(0,1)$
Si $u < \alpha(x,y)$, hacer $X_{n+1} = y$
En caso contrario, hacer $X_{n+1} = x$

El *algoritmo de Metropolis et al.* (1953) se corresponde al caso anterior con distribución de prueba simétrica, esto es $q(x,y)=q(y,x)$, con las consiguientes simplificaciones en $\alpha(x,y)$, que pasa a ser $\min\{\pi(y)/\pi(x),1\}$.

Si y tiene densidad mayor que la de x, se aceptará siempre. En caso contrario, se aceptará con cierta $\pi(y)/\pi(x)$. Para calcular $\alpha(x,y)$, no es necesario conocer π, sino sólo una función $\pi_1 \propto \pi$.

Tierney (1994) describe muchas otras variantes.

El algoritmo *MH* define un proceso de Markov con probabilidad de transición de x a y dada por

$$p_{MH}(x,y)=\begin{cases} q(x,y)\alpha(x,y) & y\neq x \\ 1-\int q(x,z)\alpha(x,z) & y=x \end{cases}$$

Es fácil ver que π es invariante para el proceso con transición p_{MH} por lo que basta asegurar la aperiodicidad y la π-irreducibilidad de p_{MH} para tener que $X_n \rightarrow X$.

El problema estaría en escoger q, que en principio es arbitraria, siempre que asegure la convergencia. Suele escogerse distribuciones de prueba simétricas y de fácil generación.

Por ejemplo, en el caso continuo escogemos distribuciones normales de media x y matriz de covarianzas Σ. Σ se elige para que la tasa de aceptación esté en torno al 25%, Gelman *et al.* (1996). Para el caso discreto, escogemos una distribución uniforme sobre un entorno de x, Bielza *et al*. (1996).

Algoritmos híbridos

Uno de ellos, debido a Müller (1991), consiste en hacer pasos de Gibbs siempre que estén disponibles las distribuciones condicionadas, y sean de generación eficiente, y pasos de Metropolis en caso contrario.
En Ríos Insua et al. (2008) se dispone de un ejemplo de aplicación del mismo.

Muestreador golpea y corre

Smith (1984), Chen y Schmeiser (1993) y Belisle et al. (1993). Las distribuciones objetivo con variables altamente correladas producen problemas de convergencia a los algoritmos basados en cadenas de Markov, especialmente a los muestreadores de Gibbs.

Muestreador por rodajas

Swendsen y Wang (1987). Añade variables auxiliares para intentar simplificar la generación.

ÍNDICE

6. Conclusiones

La mayoría de los algoritmos descritos están disponibles en software comercial o de libre distribución, sugiriéndose su uso para evitar errores de programación. Algunos ejemplos son:

•En las Recetas Numéricas (Press et al., 2007) se dispone de código (en C o FORTRAN) para generar de las distribuciones univariantes continuas exponencial, logística, normal, Rayleigh, Cauchy, gamma, χ^2, t, Beta y F; distribuciones univariantes discretas Binomial y Poisson, y de la distribución normal multivariante.

•El software estadístico R, disponible en http://www.r-project.org/, contiene en su versión básica generadores de las siguientes distribuciones: beta, binomial, Cauchy, χ^2, exponencial, F, gamma, geométrica, hipergeométrica, lognormal, logística, multinomial, binomial negativa, normal, Poisson, t, uniforme, Weibull y del estadístico de Wilcoxon. Por ser abierto, su código es reutilizable.

- El software de simulación EXTEND, dispone de generadores de las distribuciones beta, binomial, erlang, exponencial, gamma, empírica, geométrica, hiperexponencial, uniforme discreta, loglogística, lognormal, binomial negativa, normal, Pearson de tipos V y VI, Poisson, uniforme, triangular y Weibull.

- En netlib http://www.netlib.org/random se obtienen algunos generadores de buena calidad. Destaca la librería ranlib, disponible en C y FORTRAN, que incluye generadores para las distribuciones: beta, χ^2, exponencial, F, gamma, normal multivariante, uniforme, binomial, binomial negativa, multinomial y Poisson.

- La página de Devroye http://cg.scs.carleton.ca/luc/rng.html incluye enlaces a numerosos generadores de variables, destacando el paquete winrand de Stadlober, escrito en C++ y disponible en http://www.stat.tugraz.at/stadl/random.html.

- Statlib http://lib.stat.cmu.edu/ incluye abundante software en relación con generadores de variables aleatorias, pero el orden es algo caótico.

En ocasiones, por razones de mejores propiedades de reducción de la varianza, se empleará el método de inversión aunque haya otro método más eficiente o más exacto.

Además, que por razones como la anterior, en ocasiones renunciaremos a una exactitud absoluta en la generación a costa de cierta eficiencia computacional.

En general, un problema muy importante es el de proporcionar algoritmos que funcionen bien con grandes familias de distribuciones. En este sentido es Hormann y Leydold (2003) proponen un método automático y adaptativo utilizando interpolación de Hermite.

El MCMC Preprint Service http://www.statslab.cam.ac.uk/mcmc/ contiene los últimos desarrollos en métodos basados en cadenas de Markov. Algunos de los algoritmos descritos están disponibles en software comercial, de código abierto o de libre acceso,.
Otros enlaces para descarga de sw son BUGS disponible en http://www.mrc-bsu.cam.ac.uk/bugs/ e HYDRA, disponible en http://sourceforge.net/projects/hydra-mcmc/.

7. Referencias

Ahrens, J.H. y Dieter, U. (1988). Efficient table-free Sampling Methods for the Exponential, Cauchy, and Normal Distributions. Comm. ACM **31**, 1330–1337.

Belisle, C.J.P., Romeijn, H.E. y Smith, R.L. (1993). Hit-and-run Algorithms for Generating Multivariate Distributions, Math. Oper. Res. **18**, 255-266. (disponible en Aula Virtual)

Bielza, C. Muller, P. y Ríos-Insua, D. (1996). Monte Carlo Methods for Decision Analysis with Applications to Influence Diagrams, Technical Report, Duke University.

Box, G.E.P. y Muller, M.E. (1958). A Note on the Generation of Random Normal Deviates, Ann. Math. Statist. **29**, 610-611.

Brent, R.P. (1974). Algorithm 488: A Gaussian Pseudo-Random Number Generator. Comm. ACM **17**, 704–706.

Casella, G. y George, E. (1992). Explaining the Gibbs Sampler, The American Statistician **10**, 273-304.

Chen, M. y Schmeiser, B.W. (1993). Performance of the Gibbs, Hit-and-Run and Metropolis Samplers, J. Comput. Graphical Statist. **2**, 251-272. (disponible en Aula Virtual)

Cheng, R.C.H. y Feast, G.M. (1979). Some Simple Gamma Variate Generators, Appl. Statist. **28**, 290-295.

Fishman, G.S. (1996). Monte Carlo: Concepts, Algorithms and Applications, Springer.

Geldfand, A.E. y Smith, A.F.M. (1990). Sampling-based Approaches to Calculating Marginal Densities, J. Am. Stat. Assoc. **85**, 394-409.

Gelman, A., Roberts, G. y Gilks, W. (1996). Efficient Metropolis Sampling Rules, in: Bernardo, Berger, Dawid y Smith (eds.) Bayesian Statistics 5, Oxford University Press. (disponible en Aula Virtual)

Geman, S. y Geman, D. (1984). Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images, IEEE Trans. Pa. Anal. Mach. Intel. **6**, 721-741.

Gilks, W., Richardson, S., Spieglehalter, D. (1996). Markov Chain Monte Carlo in Practice, Chapman Hall.

Hastings, W.K. (1970). Monte Carlo Sampling Methods Using Markov Chains and their Applications, Biometrika **57**, 97-109.

Hormann, W. y Leydold, J. (2003). Continuous Random Variate Generation by Fast Numerical Inversion, ACM Trans. Model. Comput. Simul. **13**, 347-362.

Kronmal, R. y Peterson, A. (1981). A Variance of the Acceptance-Rejection Method for the Computer Generation of Random Variables, JASA **76**, 446-451.

Marsaglia, G. (1977). The Squeeze Method for Generating Gamma Variates, Comp. Math. Appl. **3**, 321-325.

Marsaglia, G., y Tsang, W.W. (1998). The Monty Python Method for Generating Random Variables. ACM Trans. Math. Softw. **24**, 341–350. (disponible en Aula Virtual)

Marsaglia, G., y Tsang, W.W. (2000). The Ziggurat Method for Generating Random Variables. J. Stat. Soft. **5**(8), 1-7. (disponible en Aula Virtual)

Marsaglia, G., Tsang, W.W., y Wang, J. (2004). Fast Generation of Discrete Random Variables. J. Stat. Soft. **11**(3), 1-11. (disponible en Aula Virtual)

Metropolis N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. y Teller, E. (1953). Equations for State Calculations by Fast Computing Machine, J. Chem. Phys. **21**, 1087-1091. (disponible en Aula Virtual)

Müller, P. (1991) A Generic Approach to Posterior Integration and Gibbs Sampling, Technical Report 91.09, Purdue University.

Nash, J.C. (1979). Compact Numerical Methods for Computers, Adam Hilger.

Ríos Insua, D., Ríos Insua, S., Martín, J. and Jiménez, A. (2008), Simulación: Métodos y Aplicaciones, RA-MA, Madrid.

Roberts, G.O. y Smith, A.F.M. (1994). Simple Conditions for the Convergence of the Gibbs Sampler and Metropolis Hastings Algorithm, Stoch. Proc. And Appl. **49**, 207-216.

Schervish, M. (1995). Theory of Statistics, Springer.

Smith, R.L. (1984). Efficient Monte Carlo Procedures for Generating Points Uniformly Distributed over Bounded Regions, Oper. Res. **32**, 1296-1308. (disponible en Aula Virtual)

Swendsen, R. y Wang, J. (1987). Nonuniversal Critical Dynamics in Monte Carlo Simulations, Phys. Rev. Lett. **58**, 86-88.

Thomas, D.B., Luk, W., Leong, P.H., y Villasenor, J.D. (2007). Gaussian Random Number Generators. ACM Computing Surveys (CSUR), 39(4), 11. (disponible en Aula Virtual)

Tierney, L. (1994). Markov Chains for Exploring Posterior Distributions (con discusión), Ann. Statist. **22**, 1701-1762. (disponible en Aula Virtual)

Von Neumann, J. (1951). Various Techniques in Connection with Random Digits, NBS Appl. Math. Ser. **12**, 36-38.

Walker, A.J. (1977). An Efficient Method for Generating Discrete Random Variables with General Distributions, ACM Trans. Math. Soft. **3**, 253-256. (disponible en Aula Virtual)

Wallace, C.S. (1996). Fast Pseudorandom Generators for Normal and Exponential Variates. ACM Trans. Math. Softw. **22**, 119–127.