

Algoritmos de Muestreo

El/los problema/s

- 1) Generar *muestras* de $p(\mathbf{x})$
- 2) Estimar el valor esperado de funciones bajo $p(\mathbf{x})$

$$\Phi = \int \phi(\mathbf{x})p(\mathbf{x})d\mathbf{x}$$

Si resolvemos el primero... $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(R)}\}$

Estimador $\hat{\Phi} = \frac{1}{R} \sum_r \phi(\mathbf{x}^{(r)})$

¿Por qué es difícil?

¡¡Tenemos $p(\mathbf{x})$!!

Dos dificultades:

1) Normalización: $p(\mathbf{x}) = p^*(\mathbf{x})/Z$

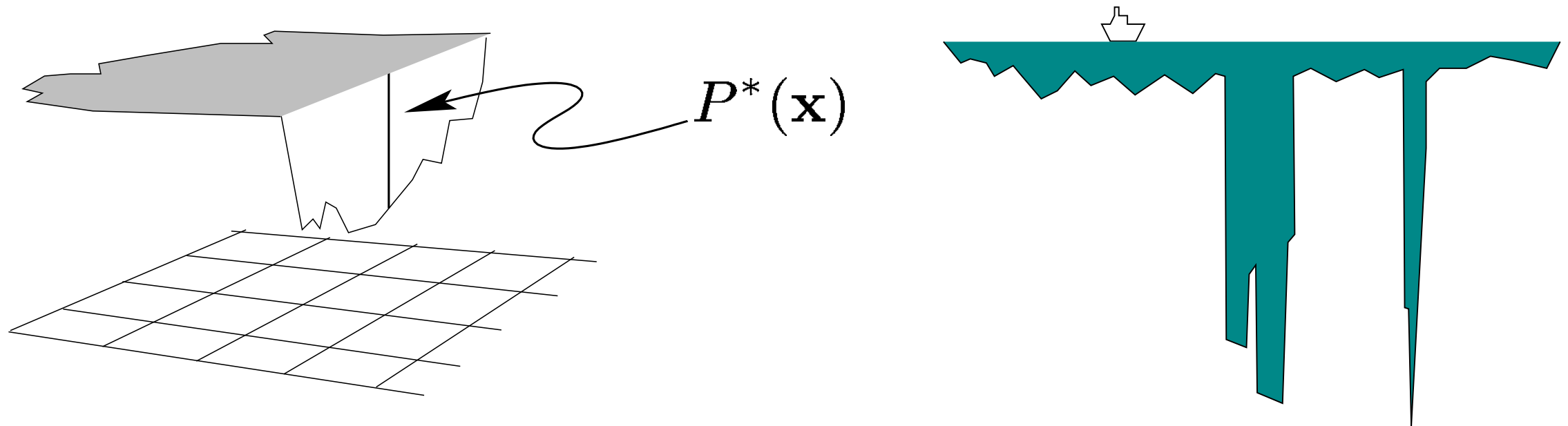
Muchas veces, tenemos sólo $p^*(\mathbf{x})$

Ej. Bayes: $p(H|D) \propto p(D|H)p(H)$

2) Aun con Z , podemos evaluar $p(\mathbf{x})$ en **cualquier** punto,
pero no en **todo** punto \mathbf{x}

Las muestras deberían venir principalmente de dónde $p(\mathbf{x})$ es grande,
pero, ¿cómo saber dónde es grande sin evaluarla en todos lados?

Analogía: medir la concentración de plankton en un lago



Problema 1: tomar muestras de agua

Problema 2: estimar la concentración media de plankton

Algunos números

Modelo de predictibilidad de palabras: $n_w + 1$ dimensiones

20 sujetos, eligen en total 30 palabras distintas,
tomamos un *grid* de 50 pasos para cada dimensión,
y una computadora de 10 GHz que evalúa la
posterior 10^{10} veces por segundo...

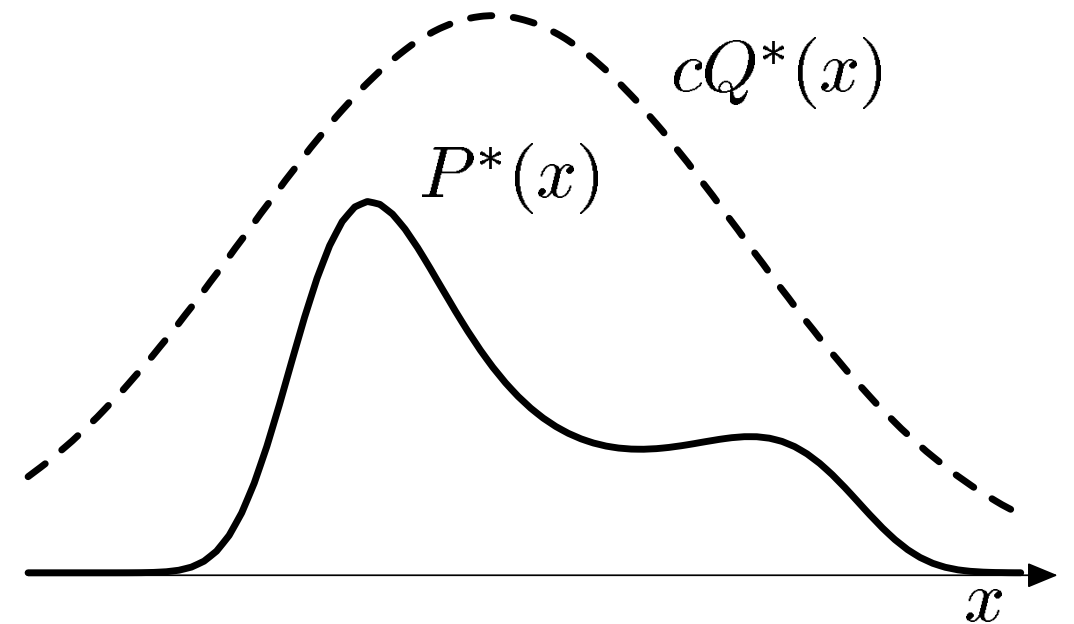
50^{30} evaluaciones / 10^{10} (evaluaciones/s) $\sim 2^{132}$ s

Edad del universo: 2^{58} s

Rejection Sampling

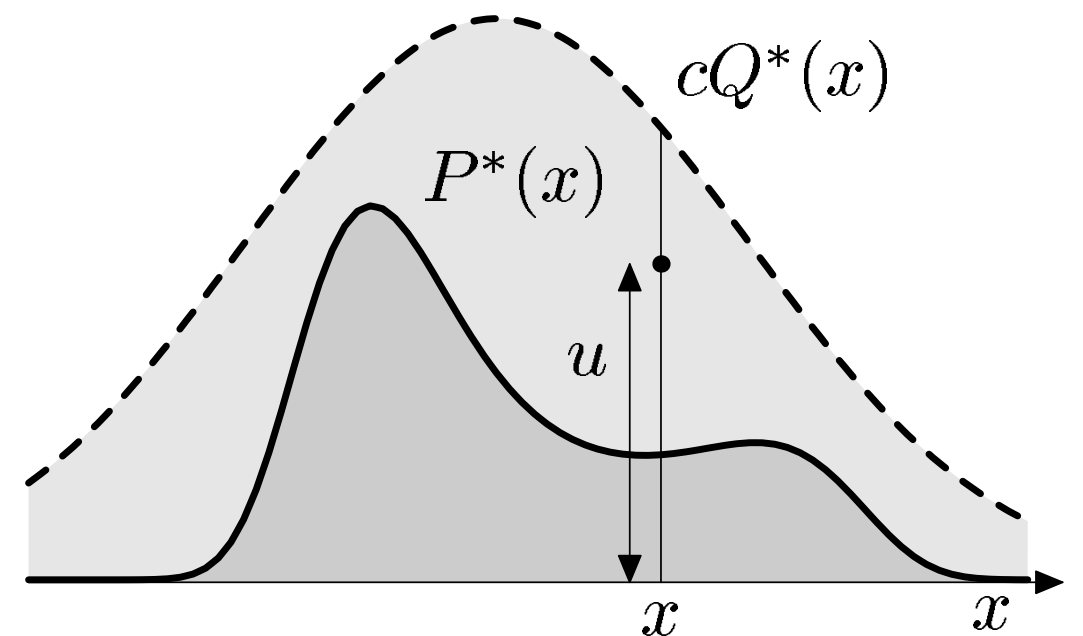
Tenemos $Q(x)$ de la que **sí** podemos tomar muestras, y c tal que:

$$cQ^*(x) > P^*(x)$$

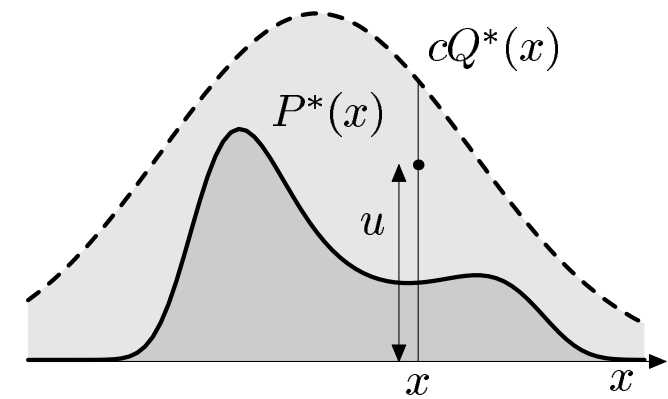


Algoritmo:

- 1) Tomamos muestra x de $Q(x)$
- 2) Evaluamos $cQ^*(x)$ y tomamos muestra u de $Uniforme(0, cQ^*(x))$
- 3) Si $u > P^*(x)$, rechazamos x , si no, la aceptamos



Rejection Sampling



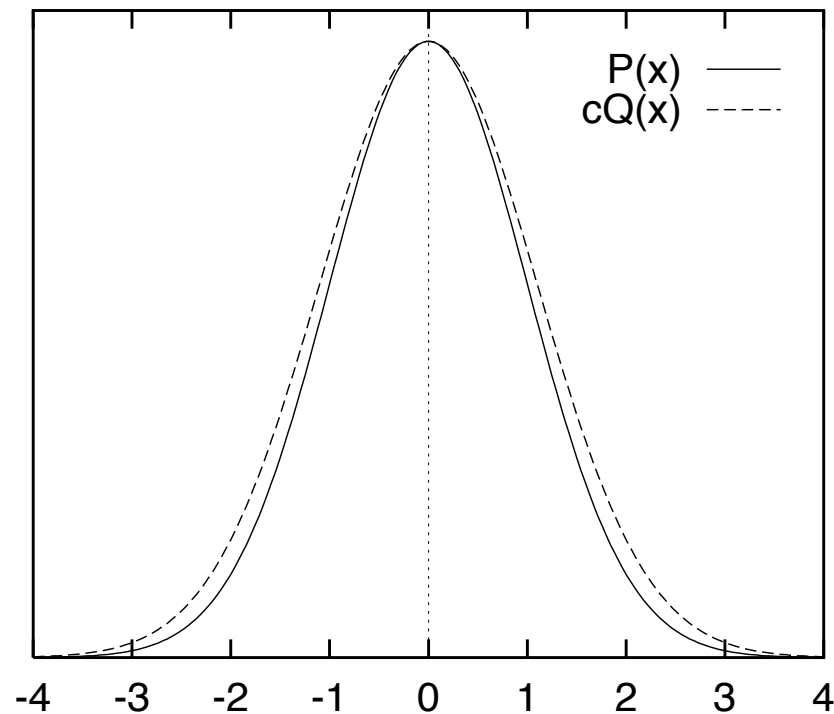
Funciona bien si Q es una buena aproximación a P

Si no, c va a tener que ser grande, y
habrá muchos rechazos

En muchas dimensiones: por lo general,
difícil incluso *hallar* c

Rejection Sampling

Ej. dos gaussianas, una con desvio 1% mayor



$$p(x) = \frac{1}{(2\pi\sigma^2)^{N/2}} e^{-\frac{x^2}{2\sigma^2}}$$

$$c = \frac{(2\pi\sigma_Q^2)^{N/2}}{(2\pi\sigma_P^2)^{N/2}} = \exp\left(N \ln \frac{\sigma_Q}{\sigma_P}\right)$$

c crece exponencialmente con la dimensión N
aquí $c \sim 1.35$ para $N=30$ pero $c \sim 20000$ para $N=1000$

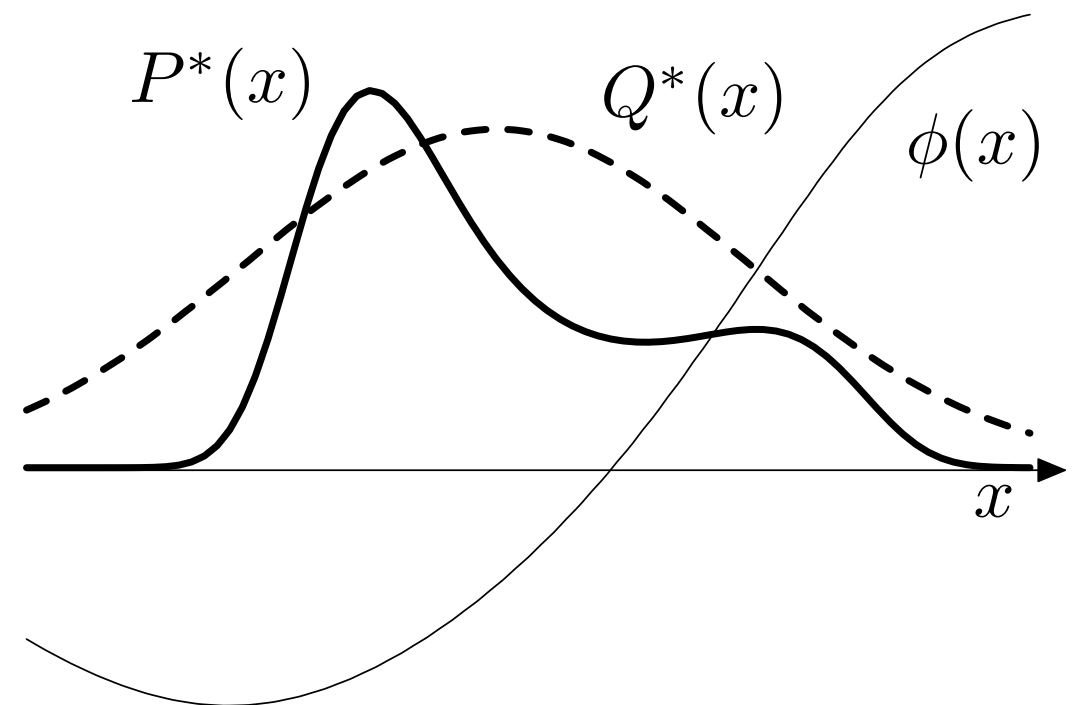
Útil para distribuciones unidimensionales,
pero no para dimensiones altas

Importance Sampling

Técnica para el problema 2, estimar valores esperados de funciones, no para tomar muestras.

Nuevamente, podemos evaluar $P^*(x)$ pero no tomar muestras de $P(x)$, y contamos con $Q(x)$ de la que podemos tomar muestras y podemos evaluar $Q^*(x)$

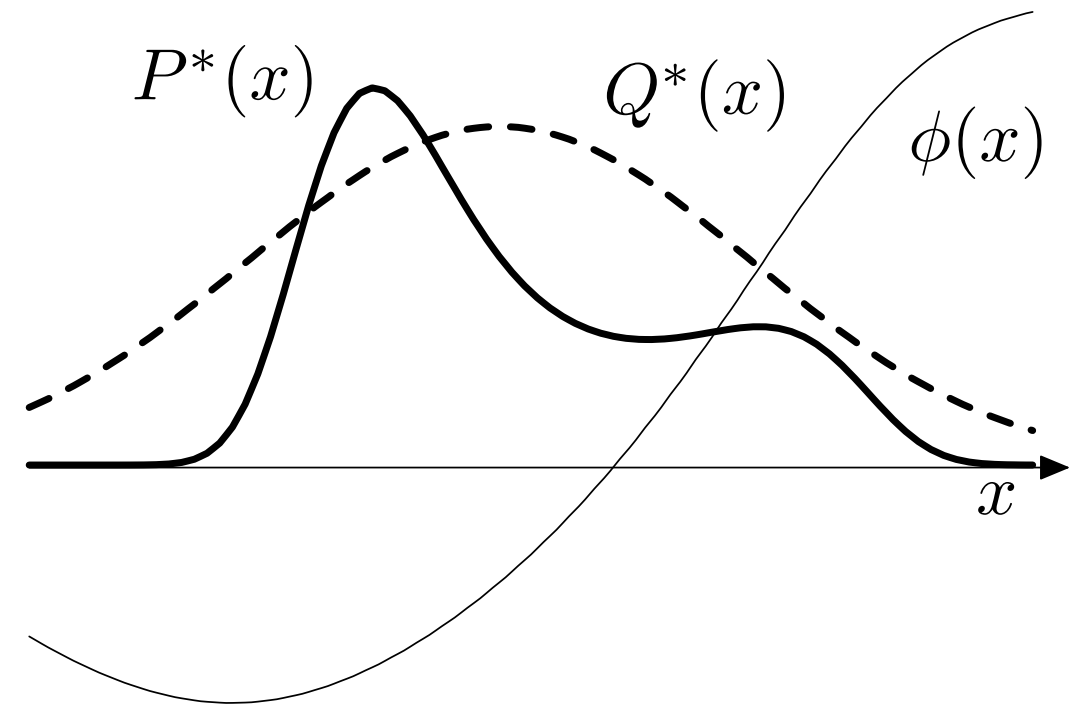
Queremos estimar el valor esperado de $\phi(x)$



Importance Sampling

Queremos estimar el valor esperado de $\phi(x)$

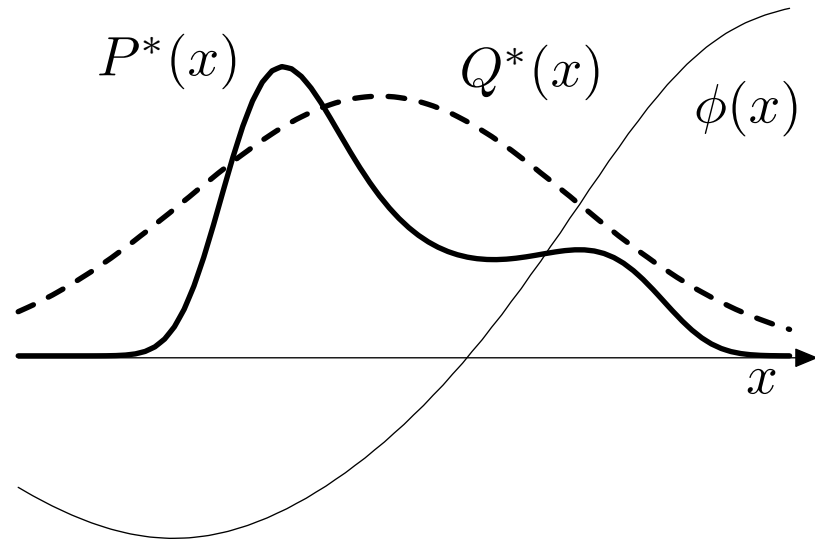
$$\hat{\Phi} = \frac{1}{R} \sum_r \phi(\mathbf{x}^{(r)})$$



Algoritmo:

- 1) Generar R muestras de $Q(x)$: $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(R)}\}$
- 2) Computar los pesos de las distintas muestras: $w_r \equiv \frac{P^*(x^{(r)})}{Q^*(x^{(r)})}$
- 3) Estimar el valor esperado como: $\hat{\Phi} \equiv \frac{\sum_r w_r \phi(x^{(r)})}{\sum_r w_r}$

Importance Sampling



$$w_r \equiv \frac{P^*(x^{(r)})}{Q^*(x^{(r)})} \quad \hat{\Phi} \equiv \frac{\sum_r w_r \phi(x^{(r)})}{\sum_r w_r}$$

Difícil estimar cuán confiable es el estimador: si $Q(x)$ es chica en un lugar donde $|\phi(x)P^*(x)|$ es grande, estimamos mal sin enterarnos..

Tamaño efectivo de muestra

$$S_{\text{eff}} = \frac{1}{\sum_r (\tilde{w}(x^{(r)}))^2}$$

Pesos normalizados

$$\tilde{w}(x^{(r)}) = \frac{R w(x^{(r)})}{\sum_{r'} w(x^{(r')})}$$

En muchas dimensiones, difícil “embocar” la región típica de probabilidad de $P(x)$, luego gran variación en los pesos.. Incluso en la región típica, los pesos difieren por factores de orden $\exp(\sqrt{N})$

Entonces.. ¿qué hacemos?

MCMC

Markov Chain Monte Carlo

Cadenas de Markov: la probabilidad del siguiente estado depende del estado en que estamos

Monte Carlo: Proceso Aleatorio

Algoritmo de *Metropolis*

Nuevamente, somos capaces de evaluar $P^*(x)$ en cualquier x

Densidad de *propuestas* $Q(x;x')$, que *depende del estado actual* x' , y es *simétrica* en x, x'

Algoritmo:

1) Elegir valor inicial $x^{(0)}$

2) Para $t=1\dots$, repetir:

a) Tomar muestra de $Q(x'; x^{(t-1)})$

b) Calcular la razón de densidades $r = \frac{P^*(x')}{P^*(x^{(t-1)})}$

c) Tomar:

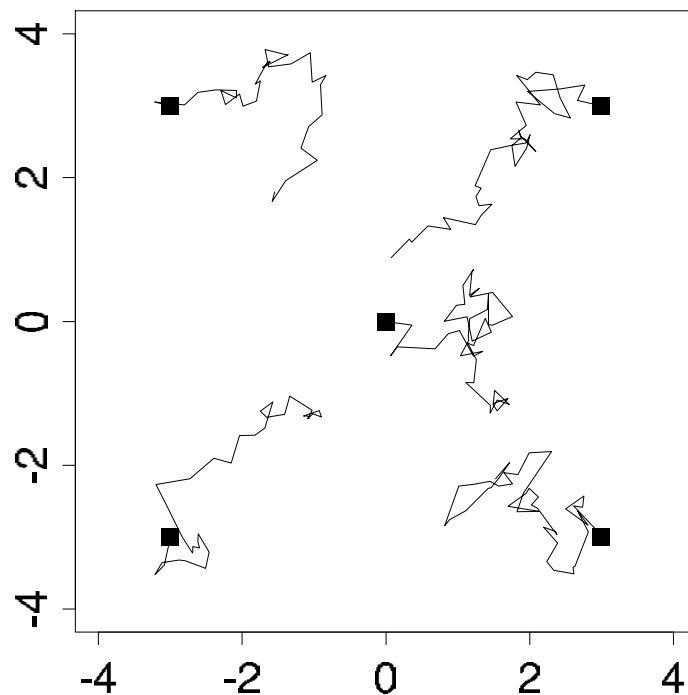
$$x^{(t)} = \begin{cases} x' & \text{con probabilidad } \min(r, 1) \\ x^{(t-1)} & \text{en otro caso} \end{cases}$$

Las muestras repetidas *no se descartan*, son muestras válidas

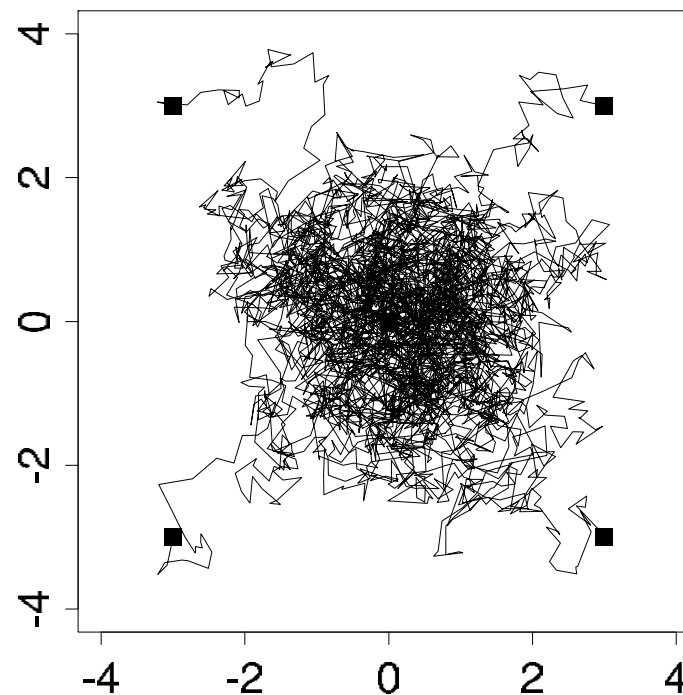
Algoritmo de *Metropolis*

Normal bivariada

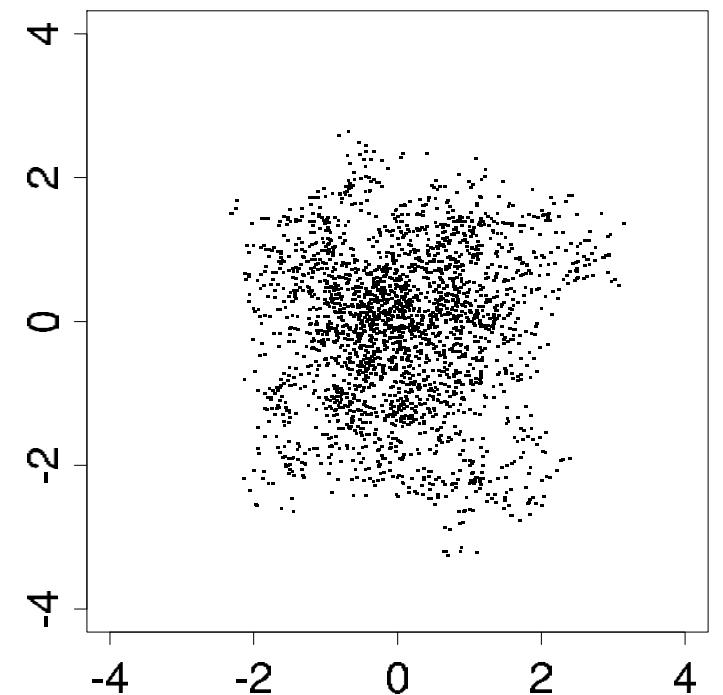
50 pasos



1000 pasos



Muestras



Aquí: saltos pequeños, comportamiento
random walk.. ineficiente

¡Método útil en altas dimensiones!

Algoritmo de *Metropolis* ¿Por qué funciona?

Esquema de la prueba, en dos pasos:

1) Probar que la secuencia simulada es una cadena de Markov con una distribución estacionaria única (paso técnico, usando propiedades de estas cadenas)

$$\Pi(x)Q(x'; x) = \Pi(x')Q(x; x') \quad \text{Balance detallado, condición suficiente para estacionariedad}$$

2) Probar que la distribución estacionaria de la cadena es la distribución deseada

$$\text{Tomamos } x_a, x_b : P^*(x_b) \geq P^*(x_a)$$

$$x_a \rightarrow x_b \quad P^*(x_a)Q(x_b; x_a) \quad \swarrow \quad r = \frac{P^*(x')}{P^*(x^{(t-1)})}$$

$$x_b \rightarrow x_a \quad P^*(x_b)Q(x_a; x_b) \frac{P^*(x_a)}{P^*(x_b)} = P^*(x_a)Q(x_b; x_a)$$

Algoritmo de *Metropolis-Hastings*

Similar a *Metropolis*, sin imponer simetría en la densidad $Q(x;x')$

Algoritmo:

1) Elegir valor inicial $x^{(0)}$

2) Para $t=1\dots$, repetir:

a) Tomar muestra de $Q(x'; x^{(t-1)})$

b) Calcular la razón de densidades $r = \frac{P^*(x')}{P^*(x^{(t-1)})} \frac{Q(x^{(t-1)}; x')}{Q(x'; x^{(t-1)})}$

c) Tomar:

$$x^{(t)} = \begin{cases} x' & \text{con probabilidad } \min(r, 1) \\ x^{(t-1)} & \text{en otro caso} \end{cases}$$

$$r = \frac{P^*(x')}{P^*(x^{(t-1)})}$$



Las muestras repetidas *no se descartan*, son muestras válidas

Prueba idéntica, ahora los factores Q en r compensan la falta de simetría

Algoritmo de *Gibbs*

No podemos muestrear de $P(\mathbf{x})$, pero sí de las
condicionales $P(x_i | \{x_j\}_{j \neq i})$

Muestreamos una a una...

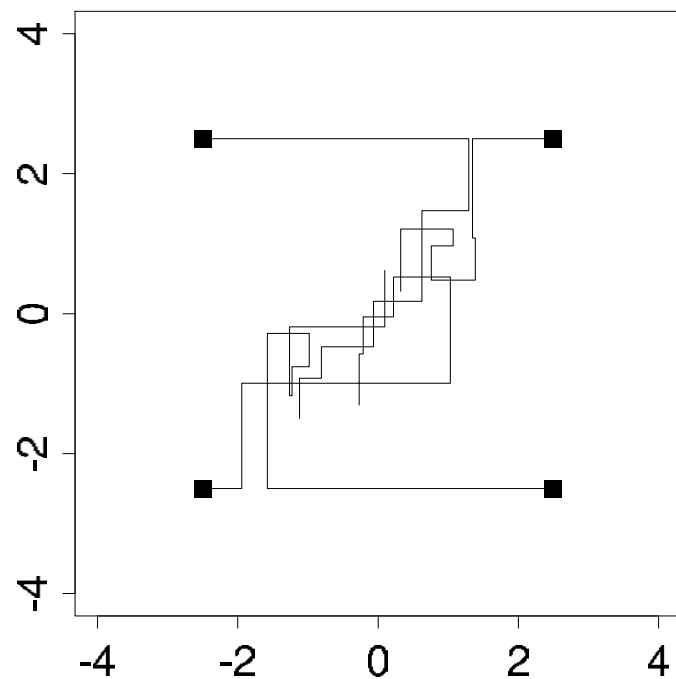
$$\begin{aligned}x_1^{(t+1)} &\sim P(x_1 | x_2^{(t)}, x_3^{(t)}, \dots, x_K^{(t)}) \\x_2^{(t+1)} &\sim P(x_2 | x_1^{(t+1)}, x_3^{(t)}, \dots, x_K^{(t)}) \\x_3^{(t+1)} &\sim P(x_3 | x_1^{(t+1)}, x_2^{(t+1)}, \dots, x_K^{(t)}), \text{ etc.}\end{aligned}$$

Prueba de convergencia: cada paso es un método
Metropolis en el que se aceptan todas las propuestas

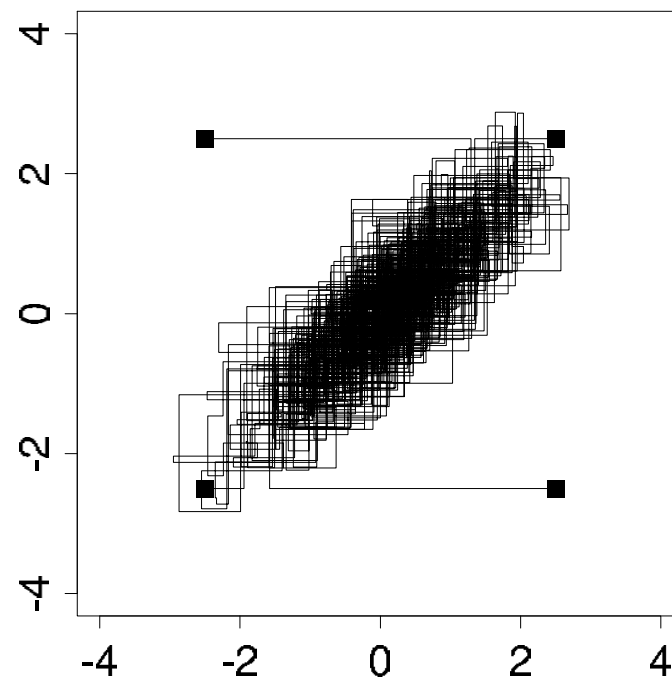
Algoritmo de *Gibbs*

Normal bivariada con $\text{corr}=0.8$

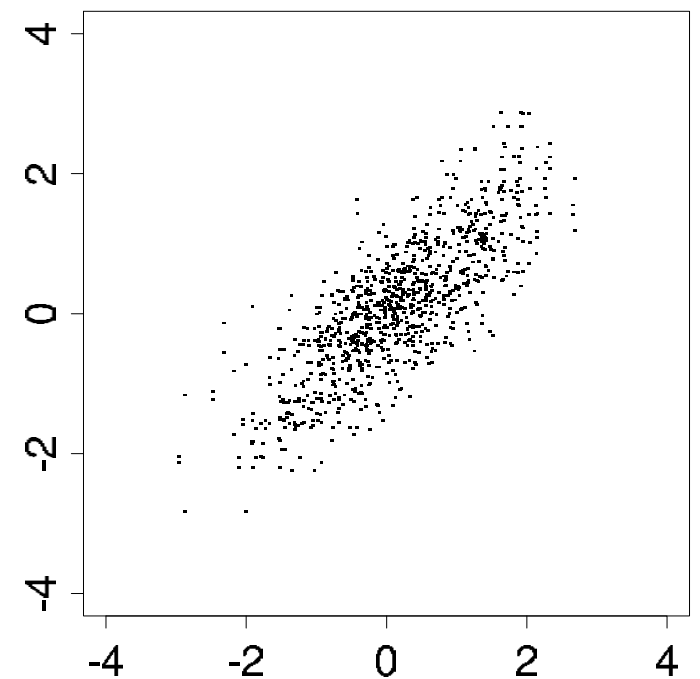
10 pasos



50 pasos



Muestras



Random walk “cuadrado”

Otras técnicas

- *Hamiltonian Monte Carlo (HMC)*
Inspirado en física, usa el *momento* de las muestras además de la posición. Método robusto y de uso general (STAN).
- *Sequential Monte Carlo (SMC)*
En lugar de acumular muestras en la *historia (MCMC)*, a cada paso llevamos una estimación de la distribución.
Ejemplos: *particle filters*.
En general, buenos modelos cognitivos, ya que demandan menos, por lo que son más plausibles como mecanismo psicológico.