Tarea 12

fl.gomez10 at uniandes.edu.co

2 de mayo de 2019

Horario de atención: Principalmente de 2:00pm a 5:00pm en la oficina i-109. También se pueden enviar dudas al correo electrónico. Entregar antes de finalizar la clase.

Trabaje iniciando sesión en la máquina virtual en línea mybinder.org/ [1].

1. Ejercicio 1 (40 puntos) Trabajo en Casa - Integrar una función bidimensional con método Monte Carlo.

Se tiene la función:

$$f(x, y) = \sin(2\pi x)\, e^{-(x^2+y^2)/(\pi^2)} + 5 \tag{1}$$

- (10 pts) Use `scipy.integrate.dblquad` para integrar. Este será nuestro valor de referencia.
- (30 pts) Evaluar la integral en $x \in (-5, 5)$ e $y \in (-3, 3)$ usando un método Monte Carlo de integración. Ajuste su método para tener un error absoluto inferior al 2 % respecto al valor de referencia.

Se tienen x_{min}, x_{max}, y_{min} e y_{max}. Se puede definir $z_{min} = 0$, y se puede calcular z_{max} como el valor más grande que pueda tomar $f(x, y)$ dentro del domino, o puede ser un valor ligeramente mayor.

Se puede definir alto, largo y ancho como las longitudes $L = w_{max} - w_{min}$ (con $w = \{x, y, z\}$). El volumen del paralelepípedo se calcula como alto por largo por ancho.

Se genera una muestra de puntos aleatorios dentro del paralelepípedo.

Para cada punto aleatorio con coordenadas (x, y, z) se evalúa si está debajo de la superficie definida por $f(x, y)$, esto es, evaluar si $z \leq f(x, y)$. Se lleva la cuenta de cuántos puntos cumplen esta condición.

Al final, la integral de $f(x, y)$ será igual al volumen del paralelepípedo por la fracción de puntos que están debajo de $f(x, y)$.

[1] https://mybinder.org/v2/gh/ComputoCienciasUniandes/FISI2026-201910/master?urlpath=lab

2. Ejercicio 2 (60 puntos) - Muestreo con el algorigmo Metropolis-Hastings

La variable x está definida dentro del dominio real $(-10, 10)$. Se tiene una distribución de probabilidad no normalizada $f(x)$ definida como:

$$f(x) = 100 - x^2 \tag{2}$$

Implemente el método de Metropolis-Hastings para hacer un muestreo de $N = 100000$ puntos.

Se genera un caminante aleatorio (Random Walk) que da cada paso según la función $f(x)$ y la posición actual del caminante (Cadena de Markov). El decidir si se da el siguiente paso aleatorio o si se permanece en la posición actual se realiza según el algoritmo de Metropolis y Hastings.

- (5 pts) Implemente la función $f(x)$. (Pro-tip: Puede asignar `-np.inf` a valores de x fuera de dominio).
- Inicie la lista de sampleo "walk" con un valor aleatorio x_0 dentro del dominio.
- (5 pts) Defina un Δx usando `random.rand()`, centrado en cero, con una amplitud que pueda variar. Este será el tamaño del paso que dará el caminante aleatorio, debe ser igual hacia la derecha o hacia la izquierda.

Una vez se han definido la función de probabilidad y se ha iniciado la lista de pasos del caminante aleatorio, se define el ciclo principal de N pasos. Preste especial atención a los signos $(<, >, \geq, \leq)$, de esto depende si funciona o no su Cadena de Markov Monte Carlo (MCMC). (20 puntos)

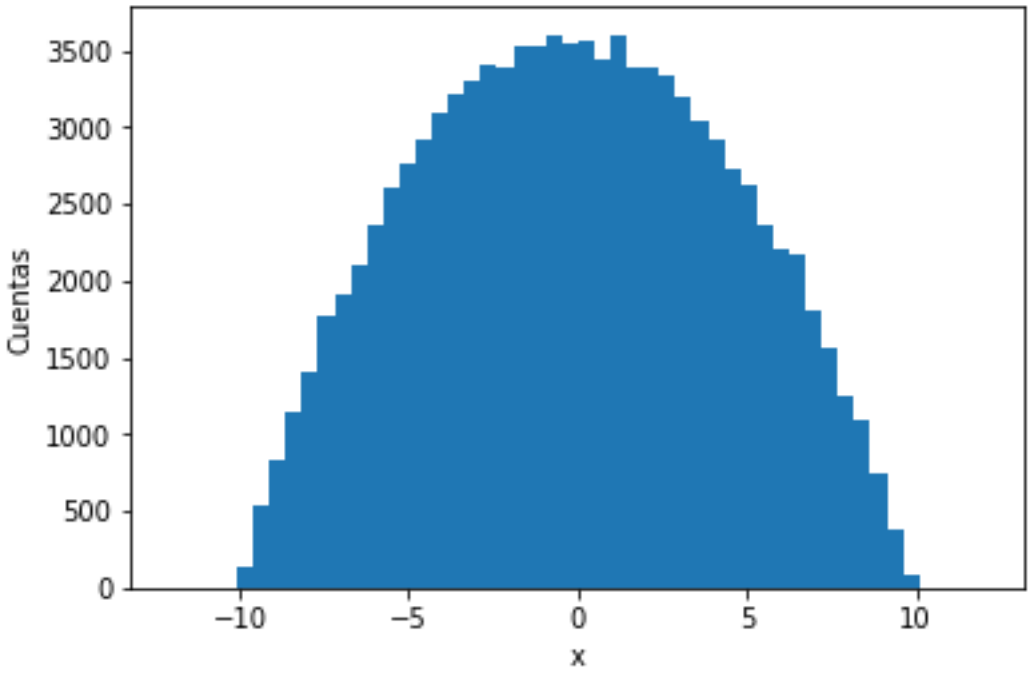
- Defina x_{old} como la última posición del caminante aleatorio
- Genere x_{new} como x_{old} mas un Δx (que puede ser positivo o negativo).
- Evalúe $\alpha = f(x_{new})/f(x_{old})$
- Si $\alpha \geq 1$ entonces se guarda x_{new} en la lista de posiciones del caminante y se repite el ciclo.
- Si $\alpha < 1$, se evalúa un número aleatorio $\beta \in [0, 1)$, si $\beta \leq \alpha$ se acepta y se guarda x_{new}, con esto se da inicio a otro ciclo. En en caso contrario se rechaza x_{new} y se guarda en la lista x_{old}.

Finalmente se realizan las gráficas. Vamos a revisar qué tan buena es la exploración que está realizando nuestro caminante aleatorio.

- (10 pts) Defina un ancho muy pequeño $|\Delta x| \sim 0{,}0001$ (el ancho del paso que da el caminante aleatorio). Graficar usando `plt.plot(walk)` y `plt.hist(walk, bins=np.linspace(-12,12,51))`

- (10 pts) Reinicie "walk", defina ahora un ancho de paso muy grande $|\Delta x| \sim 100$. Graficar usando `plt.plot(walk)` y `plt.hist(walk, bins=np.linspace(-12,12,51))`
- (10 pts) Reinicie "walk", defina un ancho de paso apropiado para el dominio de la función que estamos estudiando. Graficar usando `plt.plot(walk)` y `plt.hist(walk, bins=np.linspace(-12,12,51))`

Se espera obtener algo como esto:



Más información sobre cadenas de Markov y el diagnóstico de convergencia en esta presentación: http://halweb.uc3m.es/esp/Personal/personas/causin/esp/2012-2013/SMB/Tema8.pdf