Cómputo científico para probabilidad y estadística. Tarea 6. MCMC: Metropolis-Hastings.

Juan Esaul González Rangel

Octubre 2023

1. Simular $n = 5$ y $n = 40$ v.a Bernoulli $Be(1/3)$; sea r el número de éxitos en cada caso.

   La simulación de las variables aleaotorias se logra con el siguiente código,

```
# Samples of Bernoulli
sample5 = np.random.binomial(5,1/3,1)
sample40 = np.random.binomial(40,1/3,1)
r5 = np.sum(sample5)
r40 = np.sum(sample40)
```

   Usando la semilla 57 encontramos los siguiente valores para r_5 y r_{40},

$$r_5 = 0,$$
$$r_{40} = 11.$$

2. Implementar el algoritmo Metropolis-Hastings para simular de la posterior

$$f(p|\bar{x}) \propto p^r(1-p)^{n-r}\cos(\pi p)I_{[0,1/2]}(p),$$

   con los dos casos de n y r de arriba. Para ello poner la propuesta $(p'|p) = p' \sim Beta(r+1, n-r+1)$ y la distribución inicial de la cadena $\mu \sim U(0, 1/2)$.

   El algoritmo se encuentra implementado en la función `MH_beta()` del archivo `Tarea6.py`. La función toma tres parámetros con nombre `iterations`, `trials` y `successes`. El segundo y tercer argumentos se refieren a la cantidad de ensayos Bernoulli independientes que se realizaron y cuáles de ellos resultaron en éxito.

   El primer argumento representa la cantidad de iteraciones que realizará el algoritmo, pero este número no coincide necesariamente con la cantidad de valores de la muestra que se obtienen, pues algunos serán repetidos. Una manera de evaluar qué tan buena es la propuesta de transición dada para el problema es cuestión es medir qué tanto difieren la cantidad de iteraciones y la cantidad de valores de la muestra.

   El siguiente es un ejemplo sencillo de uso de la función,

```
simple_sample = MH_beta(100,50,28)
plt.hist(simple_sample,density=True)
```

   En el código anterior, se considera que se obtuvo una muestra de 50 ensayos Bernoulli de los cuáles 28 fueron éxitos.

   El histograma de la muestra anterior es el siguiente,

3. Argumentar porque la cadena es f-irreducible y porque es ergódica. Implementar el algoritmo con los datos descritos y discutir los resultados.

Por diseño del kernel de transición de nuestra cadena, este satisface las ecuaciones de balance detallado con $f(p \mid \bar{x})$, de donde se sigue que f es estacionaria para K. Por lo tanto, basta mostrar que K es $f(p \mid \bar{x})$-irreducible para gerantizar que la cadena es Harris-estacionaria, Harris, periódica y ergódica.

Mostramos a continuación que el kernel construido efectivamente es $f(p \mid \bar{x})$-irreducuble.

Demostración. Denotemos por μ_f a la medida de probabilidad asociada a $f(p \mid \bar{x})$, y sea A un conjunto medible tal que $\mu_f(A) > 0$. Notemos que el soporte de f es un subconjunto del soporte de $\cos(\pi p)\mathbb{1}_{[0,1/2]}(p)$ y el soporte de $p^r(1-p)^{n-r}$. Entonces el soporte de f es un subconjunto del intervalo $(0, 1/2)$, y se sigue que $A \subset (0, 1/2)$.

Para $K(x, A)$ tenemos

$$\begin{aligned} K(x, A) &= q(A \mid x)\frac{f(y)q(A \mid y)}{\mu_f(A)q(y \mid A)} + (1 - r(A))\mathbb{1}_A(y), \\ &= \int_A B(r+1, n-r+1)p^r(1-p)^{n-r}dp\frac{p^r(1-p)^{n-r}\cos(\pi p)\mathbb{1}_{(0,1/2)}(p)}{\mu_f(A)p^r(1-p)^{n-r}}. \end{aligned}$$

En la última igualdad el término $(1 - r(A))\mathbb{1}_A(y)$ no aparece, pues □

Al implementar el algoritmo con los datos discretos obtenemos los siguientes histogramas

Podemos evaluar cuál es el tamaño de la muestra que se obtuvo para cada observación de las Bernoulli. Obtenemos que de las 1,000 propuestas en cada caso, 887 fueron aceptadas para r_5 y 832 fueron aceptadas para r_{40}. El porcentaje de aceptación es de 88.7 % y 83.2 % respectivamente. Basandanos únicamente en estas observaciones, los datos parecen sugerir que la distribución de transición propuesta funciona mejor para muestrear de $f(p|r_5)$ que de $f(p|r_{40})$.

Otra observación importante es que, a pesar de que se obtuvieron 887 y 832 valores de muestreo, estos no constituyen una muestra i. i. d., pues al provenir de observaciones simuladas de una cadena de Márkov existe una correlación. Para poder considerar a la muestra independiente, sería necesario tomar valores suficientemente espaciados a lo largo de toda la muestra, de manera que la correlación entre ellos sea cercana a cero.

Similarmente, no podemos afirmar que desde el primer valor que obtuvimos estamos muestreando de f, ya que la convergencia a la distribución objetivo es asintótica. Para poder hablar de un muestreo real de f sería necesario descartar una cantidad fija de valores que se consideran observaciones de la cadena que aún no son "suficientemente cercanas" a la distribución de f.

4. Implementar el algoritmo Metropolis-Hastings con la posterior de arriba tomando una propuesta diferente.

   Como buscamos distribuciones que tengan soporte en $(0, 1)$, una propuesta puede ser la uniforme continua en $(0, 1)$. Una variante del algoritmo de Metropolos-Hastings que utiliza una densidad uniforme como función de transición de la cadena original se encuentra implementada en el archivo `Tarea6.py` con el nombre de `MH_unif()`. La función toma como entrada los mismos argumentos que la función `MH_beta()`.

   Al hacer 1,000 iteraciones de `MH_unif()` para r_5 y 1,000 para r_{40} encontramos que el total de datos en la muestra es 210 y 212, respectivamente.

   Los histogramas de los valores son los siguientes,

   Evidentemente, la distribución uniforme es una peor propuesta que la beta para este problema. En las siguiente gráficas podemos comparar el comportamiento de las cadenas con transición beta y uniforme para r_5 y r_{40}. Notemos que a pesar de ser la misma cantidad de puntos en cada caso, la cadena con distribución uniforme rechaza la transición con más frecuencia, por lo que el muestreo es más lento.

   Tras observar los ejemplos anteriores podemos concluir que, a pesar de que la implementación del algoritmo de Metropolis-Hastings es conceptualmente simple, obtener un algoritmo que pueda muestrear de una distribución objetivo de manera eficiente requiere varias decisiones de diseño.