Tarea 9. MCMC: Tarea final

Ricardo Chávez Cáliz

15 de noviembre de 2017

1. Problema en ecología

Sean $X_1, ...X_m$ variables aleatorias donde X_i denota el número de individuos de una especie en cierta región. Suponga que $X_i|N,p \sim \text{Binomial}(N,p)$, entonces

$$f(\bar{x}|N,p) = \prod_{i=1}^{m} \frac{N!}{x_i!(N-x_i)!} p^{x_i}(1-p)^{N-x_i}.$$

Como se espera que sea difícil observar a los individuos se considera a priori $p \sim \text{Beta}(\alpha,\beta)$ con $\alpha = 1, \beta = 20$, para N consideraremos una distribución uniforme discreta, i.e. $N \sim U\{0,1,2,\ldots,N_{max}\}$, como la especie no es muy abundante consideramos $N_{max} = 1000$.

A partir del algoritmo MH, se simularon valores de la distribución posterior $f(N,p|\bar{x})$ con los siguientes datos 4, 9, 6, 7, 8, 2, 8, 7, 5, 5, 3, 9, 4, 5, 9, 8, 7, 5, 3, 2; $m = 20$. Para esto se consideró como distribución inicial $p \sim \text{U}(0,1)$ y $N \sim \text{U}_d\{M, M+1, ..., N_{max}\}$ donde $M = \text{máx}_{i\epsilon\{1,...m\}}(X_i)$ y un kernel híbrido con las siguientes propuestas:

- Propuesta 1: Condicional total de p (kernel Gibbs).

$$p'|N,\bar{x} \sim \text{Beta}(\alpha + \sum_{i=1}^{m} X_i, \beta + mN - \sum_{i=1}^{m} X_i)$$

- Propuesta 2: La apriori.

$$p' \sim \text{Beta}(1,20) \text{ y } N_p \sim U\{0,1,2,\ldots,N_{max}\}$$

- Propuesta 3: Caminata aleatoria

$$N_p = N + \epsilon, \quad \mathbb{P}(\epsilon = 1) = \frac{1}{2} = \mathbb{P}(\epsilon = -1).$$

Tenemos que los datos se distribuyen binomialmente por lo que la verosimilitud está dada por

$$f(\bar{x}|N,p) = \prod_{i=1}^{m} \frac{N!}{x_i!(N-x_i)!} p^{x_i}(1-p)^{N-x_i}.$$

Con las distribuciones a priori consideradas tenemos que la posterior $f(N,p|\bar{x})$ es proporcional a

$$\frac{(N!)^m}{\prod_{i=1}^m (N-x_i)!} \cdot p^{\sum_{i=1}^m x_i} \cdot (1-p)^{N\cdot m} \cdot (1-p)^{-\sum_{i=1}^m x_i} \cdot p^{\alpha-1} \cdot (1-p)^{\beta-1}$$

así $log(f(N,p|\bar{x})) = m \cdot log(N!) + \sum_{i=1}^m x_i \cdot log(p) + N \cdot m \cdot log(1-p) - \sum_{i=1}^m x_i \cdot log(1-p) + (\alpha-1)\cdot log(p) + (\beta-1)\cdot log(1-p) - \sum_{i=1}^m log((N-X_i)!)$. Como $log(k!) = log(\Gamma(k+1))$ se usó `gammaln` de `scipy.special` para calculos adecuados de $log(f(N,p|\bar{x}))$

Con la información anterior se pudo calcular de manera adecuada la probabilidad de acepación para cada propuesta

1. Como la propuesta es de Gibbs, entonces directamente se tiene que $\rho_1 = 1$

2. Considerando las funciones de densidad de las distibuciones a priori sin considerar las constantes si $g(N,p) = log(f(N,p|\bar{x})) + (\alpha-1)\cdot log(p) + (\beta-1)\cdot log(1-p)$ se tiene

$$\rho_2 = e^{min\{0, g(N_p,p'|\bar{x}) - g(N,p|\bar{x})\}}$$

3. Como la caminata aleatoria es una propuesta simétrica se tiene que $\rho_3 = min\{1, \frac{f(N_p,p')}{f(N,p)}\}$, para cálculos computacionales se usó

$$\rho_3 = e^{min\{0, log(f(N_p,p'|\bar{x})) - log(f(N,p|\bar{x}))\}}$$

Para la ejecución del algoritmo aquí mostrada se rechazaron el 18.58 % de las propuestas, el valor promedio de p fue de 0.0135318691986 y para N fue 455.966193239.

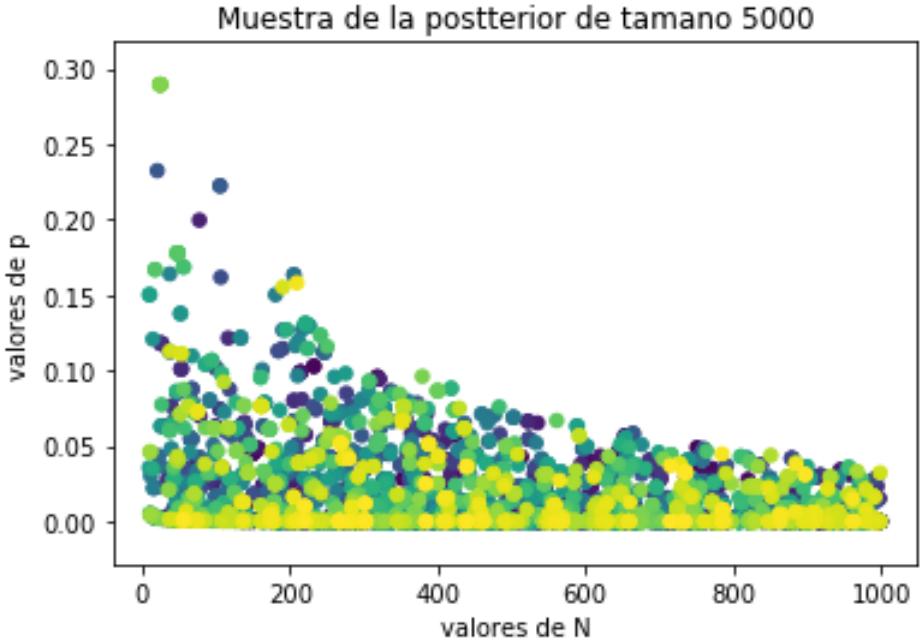

Figura 1: Muestra de tamaño 5000 para $f(N,p|\bar{x})$ obtenida del algoritmo MCMC

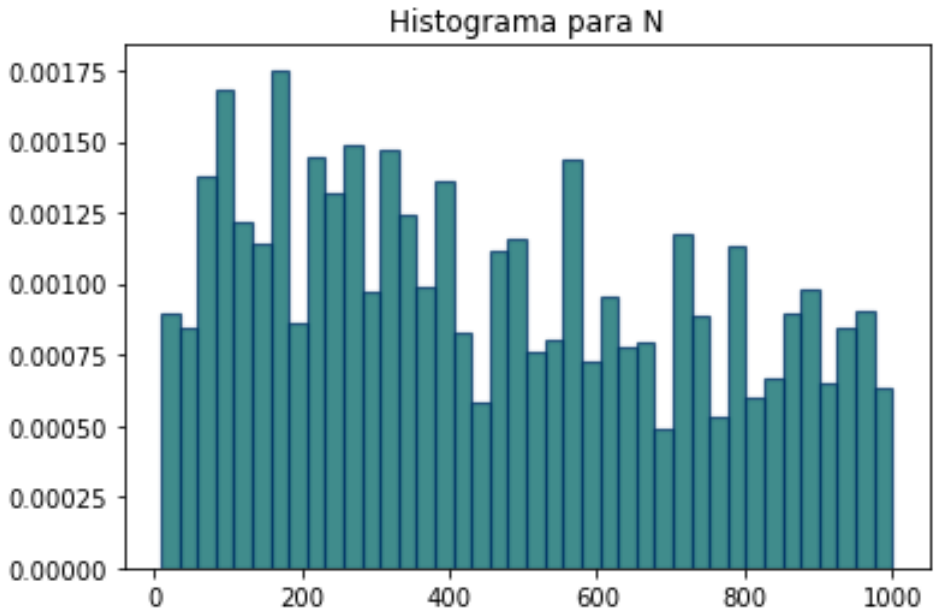

Figura 2: Histograma para N de la muestra de tamaño 5000 para $f(N, p|\bar{x})$ obtenida del algoritmo MCMC

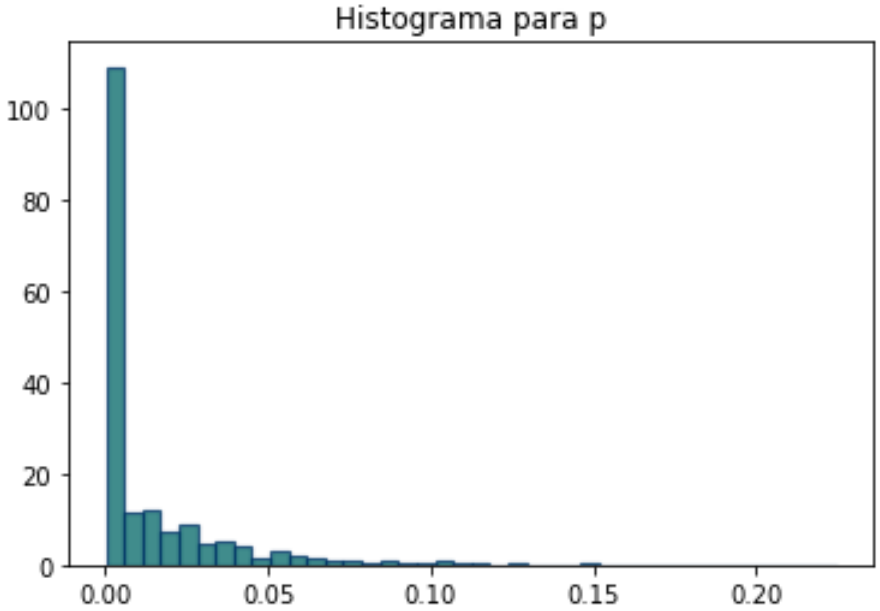

Figura 3: Histograma para p de la muestra de tamaño 5000 para $f(N, p|\bar{x})$ obtenida del algoritmo MCMC

2. Estudio de mercado

Se tiene un producto y se realiza una encuesta con el fin de estudiar cuánto se consume dependiendo de la edad. Sea Y_i el monto de compra y X_i la covariable la cual representa la edad.

Suponga que $Y_i \sim Po(\lambda_i)$ (distribución Poisson con intensidad λ_i)

$$\lambda_i = cg_b(x_i - a)$$

para g_b la siguiente función de liga

$$g_b(x) = \exp\left(-\frac{x^2}{2b^2}\right).$$

Si $\lambda_i = 0$ entonces $P(Y_i = 0) = 1$. a representa los años medio del segmento (años), c = gasto promedio (pesos), b = "amplitud" del segmento (años).

Considerando las siguientes distribuciones a priori:

$$a \sim N(35, 5), \quad c \sim Gama(3, 3/950), \quad b \sim Gama(2, 2/5)$$

El segundo parámetro de la normal es desviación estándar y el segundo parámetro de las gammas es taza (*rate*).

Usando MH se simularón de la distribución posterior de a, c y b.

Los datos son estos, $n = 100$:

X = [17, 14, 28, 51, 16, 59, 16, 54, 52, 16, 31, 31, 54, 26, 19, 13, 59, 48, 54, 23, 50, 59, 55, 37, 61, 53, 56, 31, 34, 15, 41, 14, 13, 13, 32, 46, 17, 52, 54, 25, 61, 15, 53, 39, 33, 52, 65, 35, 65, 26, 54, 16, 47, 14, 42, 47, 48, 25, 15, 46, 31, 50, 42, 23, 17, 47, 32, 65, 45, 28, 12, 22, 30, 36, 33, 16, 39, 50, 13, 23, 50, 34, 19, 46, 43, 56, 52, 42, 48, 55, 37, 21, 45, 64, 53, 16, 62, 16, 25, 62]

Y = [165, 9, 493, 0, 72, 0, 89, 0, 0, 70, 79, 96, 0, 1127, 548, 4, 0, 0, 0, 1522, 0, 0, 0, 0, 0, 0, 0, 80, 5, 38, 0, 11, 8, 4, 31, 0, 174, 0, 0, 1305, 0, 39, 0, 0, 18, 0, 0, 4, 0, 1102, 0, 94, 0, 13, 0, 0, 0, 1308, 33, 0, 90, 0, 0, 1466, 156, 0, 39, 0, 0, 496, 2, 1368, 190, 0, 12, 76, 0, 0, 5, 1497, 0, 6, 533, 0, 0, 0, 0, 0, 0, 0, 0, 1090, 0, 0, 0, 93, 0, 88, 1275, 0]

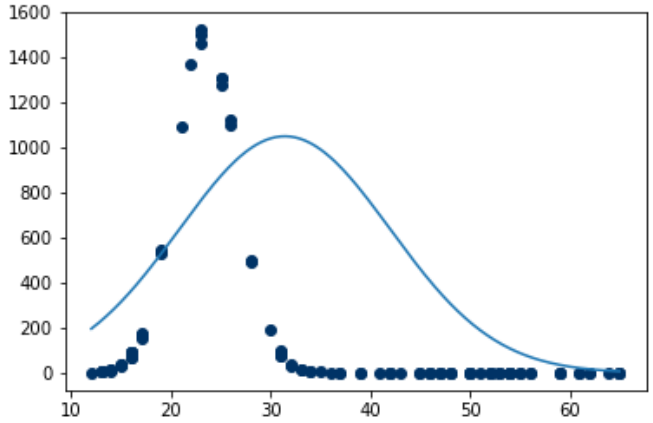

Figura 4: Muestra dada y gráfica de liga con valores aleatorios distribuidos como las funciones a priori

Para poder dar propuestas eficientes se graficó la muestra y a función de liga con parámetros aleatorios obtenidos según la distribución a priori dada, en este caso fueron a = 31.4166339314 b = 10.5963029238 y c = 1049.90123303, en base a lo observado parece razonable dar un kernel hibrido sencillo dado por caminatas aleatorias en cada componente, es decir

- Propuesta 1: Caminata aleatoria

$$a_p = a + N(0,1)$$

- Propuesta 2: Caminata aleatoria

$$b_p = b + N(0,0.1)$$

- Propuesta 3: Caminata aleatoria

$$c_p = c + N(0,2)$$

Tenemos que los datos tienen distribución Poisson con parámetro $g_b(x-a)$, por lo tanto la verosimilitud está dada por

$$f(\bar{x}|a,b,c) = \prod_{i=1}^{m} \frac{c \cdot \exp\left(-\frac{(x_i-a)^2}{2b^2}\right)^{y_i} \cdot \exp\left(-c \cdot \exp\left(-\frac{(x_i-a)^2}{2b^2}\right)\right)}{y_i!}$$

Con las distribuciones a priori consideradas tenemos que el logaritmo de la posterior $f(a,b,c|\bar{x})$ es proporcional a

$$log(c)\cdot\sum_{i=1}^{m} y_i + \left(\frac{1}{2b^2}\right)\cdot\sum_{i=1}^{m}\left(-y_i\cdot(x_i-a)^2\right) - \frac{c}{2b^2}\cdot\sum_{i=1}^{m} exp(-(x_i-a)^2) - \frac{(a-35)^2}{2\cdot 5^2} - \frac{5}{2}\cdot b + log(b) - \frac{950}{3}\dot{c} + 2\cdot log(c)$$

Con la información anterior se pudo calcular de manera adecuada la probabilidad de aceptación para cada propuesta, pues como la caminata aleatoria es una propuesta simétrica se tiene que $\rho = min\{1, \frac{f(a',b',c')}{f(a,b,c)}\}$, para cálculos computacionales se usó

$$\rho = e^{min\{0, log(f(a',b',c'|\bar{x})) - log(f(a,b,c|\bar{x}))\}}$$

Para la ejecución del algoritmo aquí mostrada se rechazaron el 62.7 % de las propuestas en una muestra de 1000 elementos, los valores promedio para a, b y c fueron 23.6166478266, 3.13611422563, 1391.7006915. Usando estos valores para gráfica la función de liga se obtuvo el siguiente resultado.

Note que el parece que el modelo escogido con los parámetros estimados ajusta de buen modo a los datos y podría ser usado para la toma de decisiones en la estrategia mercadotécnica del producto.

Los resultados anteriores se pueden interpretar pensando que la edad promedio del consumidor del producto hipotético es de aproximadamente 23.61 años , gastando un promedio de 1391.70 pesos en el producto aproximadamente, con un rango de edad de + o - 3.1361.

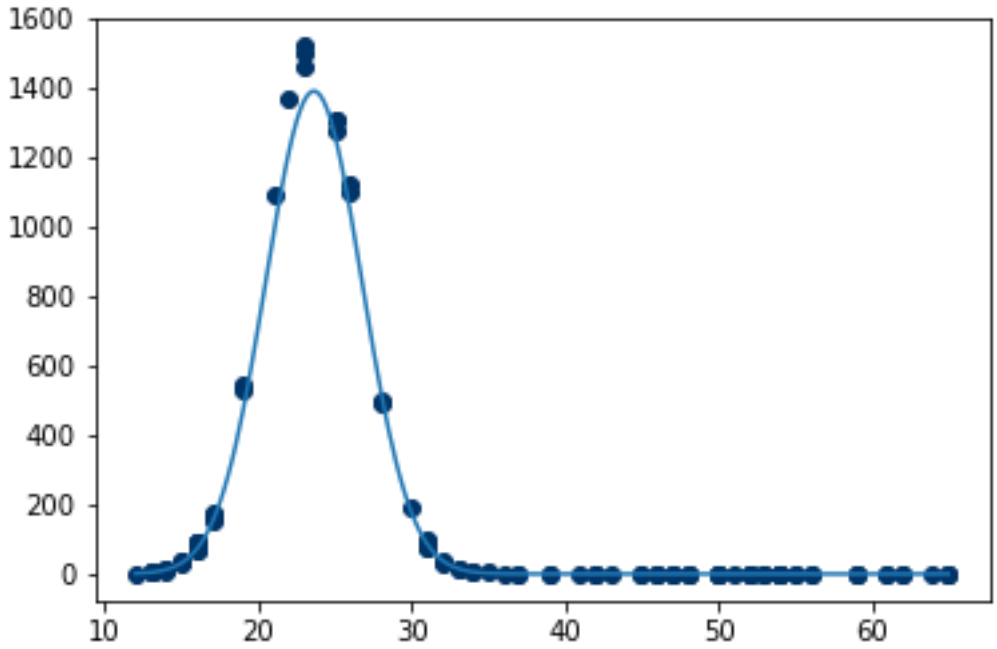

Figura 5: Muestra dada y función de liga

3. Software para estadística Bayesiana y MCMC

3.1. BUGS

En inteligencia artificial, un sistema experto es un sistema computacional que emula la habilidad de tomar decisiones como lo hace un humano experto en cierto tema. Los sistemas expertos están diseñados para resolver problemas complejos al razonar sobre conocimiento, principalmente representado por reglas del tipo "si-entonces"más que por código rutinario convencional. `BUGS` es un software que utiliza esto para realizar inferencia Bayesiana usando un Gibs sampler. El usuario especifica el modelo estadístico, que puede ser de casi cualquier complejidad, simplemente al especificar las relaciones entre las variables. El software incluye un sistema experto que determina el MCMC apropiado basado en Gibbs sampler para analizar el modelo. El software permite controlar la ejecución del esquema y da la libertad de escoger ente una amplia variedad de tipos de outputs.

¿Cómo funciona? El modelo especificado pertenece a una clase conocida de Gráficas acíclicas dirigidas (DAGS), para las cuales existe una teoría matemática subyacente. Esto permite separar el análisis de estructuras complejas arbitrariamente grandes en una secuencia de cálculos computacionales relativamente sencillos. BUGS incluye una amplia variedad de algoritmos que sus sistemas expertos pueden asignar a cada tarea computacional.

Existe el OpenBUGS y el WinBUGS, una de las principales diferencias entre ellos es la manera en que el experto del sistema toma sus decisiones. WinBUGS define un algoritmo para cada tipo de cálculo computacional posible mientas que el número del algoritmos que puedes hacr uso en OpenBUGS es mayor, haciéndolo mucho mas extenso y flexible.

3.2. NIMBLE

Este software adopta y extiende BUGS como lenguaje de modelado y permite programar con los modelos que se van creando. Otras paqueterías que usan el lenguaje de BUGS sólo para MCMC, con NIBBLE puedes convertir código BUGS en objetos modelo y usarlos para cualquier algoritmo que se quiera, lo que incluye algoritmos provistos con NIBLE y algoritmos escritos por el usuario. Usando funciones de NIMBLE, se puede extender BUGS al permitir múltiples parametrizaciones para distribuciones, funciones y distribuciones escritas por el usuario.

NIMBLE también provee MCMC, Monte Carlo secuenciales entre otras cosas.

Los algoritmos en NIMBLE están escritos para poderse adaptar a distintos modelos estadísticos, para MCMC se pueden asignar parámetros default para la elecciones del sampler pero es posible personalizar los samplers desde R por ejemplo, se pueden escoger los parámetros a muestrear en bloque, y es posible incluir tus propios samplers

3.3. JAGS

JAGS (Just Another Gibbs Sampler) es un programa para análisis de modelos Bayesianos jerárquicos que usan simulación MCMC no tan difrente a BUGS. Fue desarrollado teniendo en cuenta

- Tener un motor de cross-platform para el lenguaje de BUGs
- Se extendible, permitiendo a los usuarios escribir sus propias distribuciones y muetreadores
- Ser una plataforma para experimentar con ideas en modelado Bayesiano

JAGS tiene una licencia del tipo GNU General Public License version.

3.4. DRAM

Es una propuesta hecha por Heikki Haario, Marko Laine, Antonietta Mira y Eero Saksman que combina dos poderosas ideas que aparecen en la literatura de MCMC: *Adaptive Metropolis samplers* y *delayed rejection*. Está demostrado que este sampler no-Markoviano es ergódico y se ha visto e varios ejemplos que la combinación de las ideas es eficiente. La adaptació claramene mejora la eficiencia del algoritmo y la parte del rechazo retrasado sirve en los casos donde buenas propuestas de distribución no están disponibles. Similarmente, el rechazo retrasado provee una manera sistemática de remediar un comienzo lento en el procesos de adaptación.

3.5. Rtwalk

Es una implementación del algoritmo de MCMC para "t-walk.[en] R, desarrollado por J. Andres Christen , Depends R (>= 2.8.0). Es un sampler de propósitos generales para distribuciones continuas arbitrarias que no requiere ajuste. Este algorítmo está implementado en Python, R, C++, C y MatLab.

emcee: The MCMC Hammer. Es un software libre para implementar de manera estable y certificiada del affine-invariant ensemble sampler de MCMC propuesta por Goodman y Weare (2010). El algoritmo detrás de emcee tiene varias ventajas sobre los métodos tradicionales de muestreo de MCMC y tiene un excelente desempeño al medir la correlación en tiempo de la muestra.

Una de las mayores ventajas del algoritmo es que requiere ajuste manual en sólo uno o dos parámetos comparados con los aproximadamente N^2 que se ocupan ajustar para el algoritmo tradicional en un espacio de parámetros N-dimensional.

Explotando el paralelismo del método de ensable, emcee permite a cualquier usuario tomar ventaja de los múltiples núcleos del PC sin esfuerzo adicional. El código está disponible en línea en http://dan.iel.fm/emcee/current/ bajo licencia del MIT

3.6. PyMCMC

Es un software creado por Chris Fonnesbeck, Anand Patil, David Huard, John Salvatier de licencia libre. PyMC es un modulo de Python que implementa modelos de estadística Bayesiana y algoritmos de ajuste, incluyendo MCMC. Debido a su flexibilidad y extensión es usado para una gran clase de problemas. Junto con funcionalidad de muestreo de núcleo, PyMC incluye métodos para resumir resultados, gráficar, realizar bondad de ajuste y diagnostico de convergencia.

Características:

- Ajusta modelos estadísticos Bayesianos con cadenas de Markov y otros algoritmos
- Usa Numpy para las cuestiones numéricas siempre que es posible e incluye un sección para modelar procesos Gaussianos.
- Los muestreos en ciclos pueden ser pausados y ajustados manualmente, o bien ser salvados y reiniciados luego.
- Capacidad de resumir información, incluyendo tablas y gráficas.
- Los resultados pueden ser salvados en el disco como texto plano, Pyhton pickles, SQLite o bases de datos en MySQL, archivos hdf5.
- Disponibilidad de varios diagnósticos de convergencia.
- Extendibilidad: incorpora personalización sencilla de pasos en los métodos y distribuciones de probabilidad inusuales.
- Ciclos de MCMC pueden ser usados en programas mas extensos y los resultados puedes ser analizados con todo el poder de Python.