Tarea 10: Optimización clásica

Ricardo Chávez Cáliz

22 de noviembre de 2017

Se tienen datos provenientes de una distribución Normal truncada, es decir $x = (y_1, y_2, \ldots, y_m, z)$ donde $z = (a, \ldots, a)$ que consiste de $n - m$ elementos y $x_i \sim N(\theta, 1)$ si $x_i < a$, y $x_i = a$ en caso contrario. Por lo que

$$f(x_i|a, \theta) = \frac{1}{\sqrt{2\pi}} \cdot exp\left\{\frac{-(x_i - \theta)^2}{2}\right\} \cdot [1 - \Phi(a - \theta)] \cdot I_{(a,\infty)}(x_i)$$

con función de verosimilitud

$$L(\theta|x) = \frac{1}{(2\pi)^{m/2}} \cdot \exp\left\{-\frac{1}{2}\sum_{i=1}^{m}(x_i - \theta)^2\right\} \cdot [1 - \Phi(a - \theta)]^{n-m}$$

donde $\Phi(\cdot)$ es la función de distribución de una normal estándar.

Se pide implementar el algoritmo EM, un Gibs Sampler y el método de Newton-Raphson para obtener el estimador máximo verosímil de θ.

En nuestros conjunto de datos $n = 20, m = 15, a = 21$. datos = [18.221753, 18.418174, 18.720224, 19.067637, 19.128777, 19.402623, 19.507782, 19.580571, 19.632340, 19.930952, 20.116566, 20.142095, 20.445327, 20.461254, 20.646856, 21.000000, 21.000000, 21.000000, 21.000000, 21.000000,]

1. Algoritmo EM

Para cada paso de maximización se busca $\theta^{(j+1)} = \text{argmax} Q(\theta|\theta^{(j)}, x)$ donde $Q(\theta|\theta^{(j)}, x) = E_z\left[logf(x, z|\theta)|\theta^{(j)}, x\right]$ calculada en el paso de expectativa. En este caso el logaritmo de la distribución conjunta está dada por

$$\log\left(f(x, z|\theta)\right) \propto -\frac{1}{2}\sum_{i=1}^{m}(x_i - \theta)^2 - \frac{1}{2}\sum_{i=m+1}^{n}(z_i - \theta)^2$$

De esta manera tenemos que:

$$Q(\theta|\theta^{(j)}, x) = \mathbb{E}_z\left[-\frac{1}{2}\sum_{i=1}^{m}(x_i - \theta)^2 - \frac{1}{2}\sum_{i=m+1}^{n}(z_i - \theta)^2 \mid \theta^{(j)}, x\right]$$

la cual probamos que es maximizada con

$$\widehat{\theta}^{(j+1)} = \frac{m}{n}\bar{X} + \frac{n-m}{n} \cdot \left[\widehat{\theta}^{(j)} + \frac{\varphi(a - \widehat{\theta}^{(j)})}{1 - \Phi(a - \widehat{\theta}^{(j)})} \right]$$

donde $\bar{X}$ es el promedio muestral y $\varphi(\cdot)$, $\Phi(\cdot)$ son las funciones de densidad y de distribución para una normal estándar respectivamente.

Con la expresión anterior para $\widehat{\theta}^{(j)}$ la implementación del algoritmo EM es directa ya que estableciendo $\theta^0 = \bar{X}$ basta calcular $\theta^{(j)}$ hasta un cierto límite de iteraciones (se estableció 100 como límite) verificando que el $\theta^{(j+1)} - \theta^{(j)} > 1e-6$, de lo contrario se detienen las iteraciones y se devuelve el último θ^j calculado. Dicha implementación puede encontrarse en el código de `Python` adjunto.

Para verificar el buen comportamiento de la implementación se simuló una muestra normal estándar censurada para valores mayores o iguales a 1 de tamaño 1000. Se calculó el promedio muestral sin tomar en cuenta los datos censurados -0.306571650501 y luego el estimador obtenido con el algoritmo EM y en 4 iteraciones se obtuvo $\widehat{\theta} = -0.0496753870332$ que es más cercano a cero que el primer estimador como se espera.

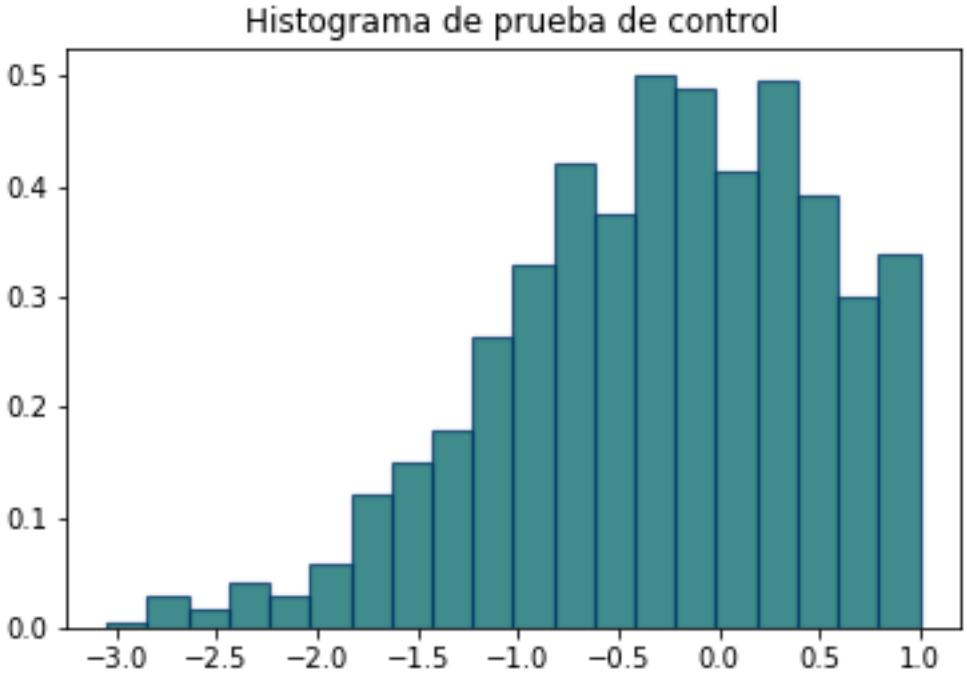

Figura 1: Histograma de la muestra dada, removiendo los datos mayores o iguales a 1

Para los datos aquí presentados se obtuvo que el estimador de la media ignorando censura es 19.92114655, el estimador de la media con algoritmoEM en 5 iteraciones es 20.0552242528 en un tiempo de ejecución 0.00895652321867.

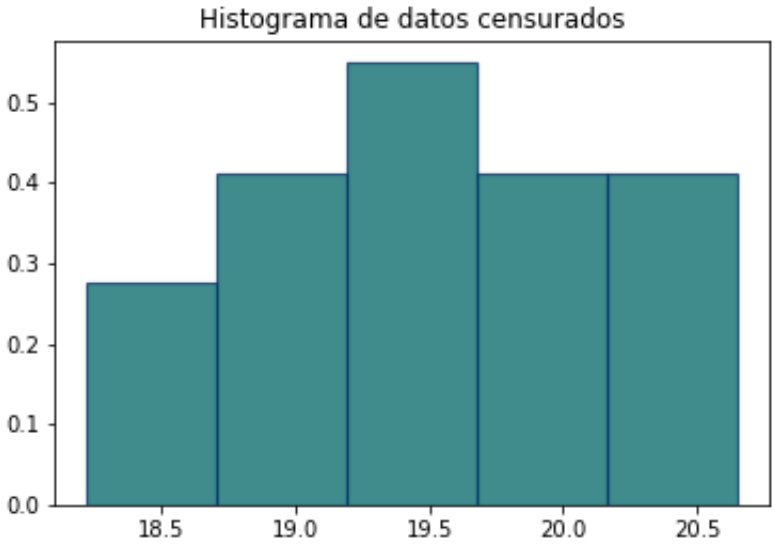

Figura 2: Histograma de la muestra normal estándar con censura, removiendo los datos mayores o iguales a 21

2. Gibs sampler

se consideró un Gibs Sampler con parámetro artificiales $z_1, \ldots, Z_{n-m}$ los cuales fueron añadidos en la posterior $f(\theta, z_1, \ldots, z_{n-m}|x)$ y se implementó MCMC. Para esto consideramos las condicionales totales $f(z_i|\theta, x)$ las cuales se distribuyen como una normal truncada las cuales podemos simular con la inversa numérica Φ^{-1}, incluida en `scipy.stats.norm` con la función ppf. Considerando la distribución a priori para $\theta \sim N(18, 1)$ se obtuvo que el promedio de las θ simuladas para una muestra de tamaño 5000 fue de 20.054202679 en un tiempo de ejecución de 5.38168182815.

Para analizar el comportamiento de la cadena se analiza la muestra simulada para los θ obtenidos, para esto se grafica el logaritmo de el valor obtenido para θ en cada simulación respecto a la iteración correspondiente

Como se muestra en la figura 3 el burn-in de la cadena es muy corto y por ello en la figura 4 podemos exponer los últimos 4985 elementos de la muestra para θ y $z1$, si riesgo a notar elementos que no corresponden a las simulaciones deseadas. Note que los valores de θ se comportan como una normal con media aproximada a lo calculado antes y z_i correspondiente a los valores de la cola de una normal. En la figura 5 se muestra la gráfica de la trayectoria de la cadena con un menor número de simulaciones.

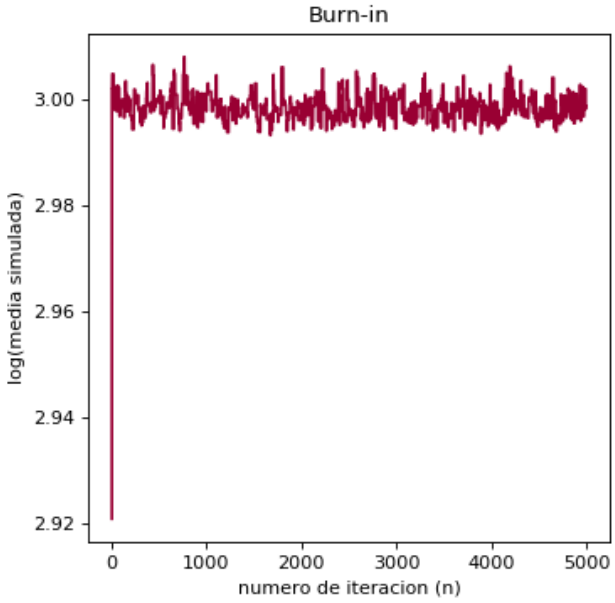

Figura 3: Se muestra el burn-in de la cadena

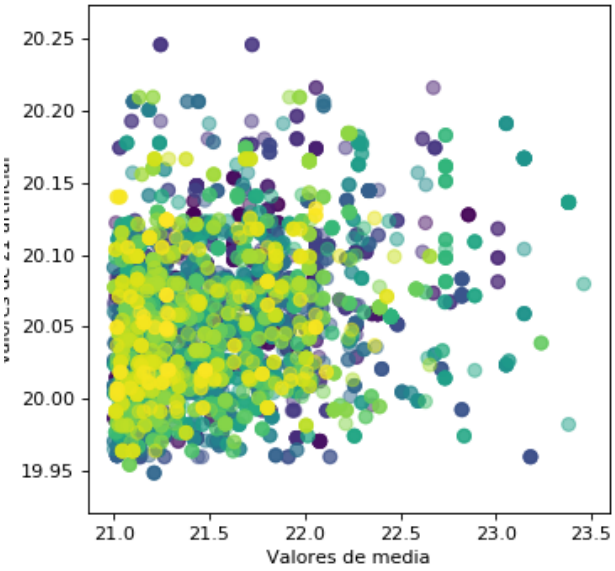

Figura 4: Muestra para θ y z_1

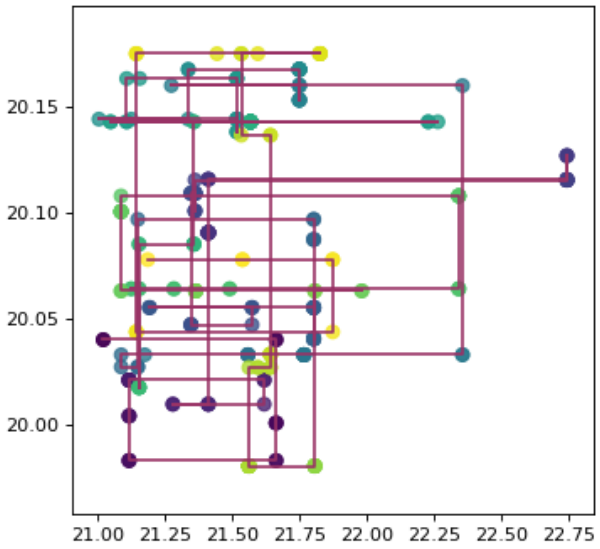

Figura 5: Trayectoria de la cadena

3. Método de Newton-Raphson

Se implementó el método de Newton-Raphson para encontrar a $\hat{\theta}$ que maximice la log-verosimilitud $l(\theta|x) = log(L(\theta|x))$. Para esto se calculó la derivada de log-verosimilitud:

$$\frac{d\ l(\theta|x)}{d\theta} = \sum_{i=1}^{m}(X_i - \theta) + \frac{(n-m)\cdot\varphi(a-\theta)}{1-\Phi(a-\theta)}$$

Para encontrar sus raíces por el método de Newton-Raphson fue necesario también calcular la segunda derivada.

$$\frac{d^2\ l(\theta|x)}{d\theta^2} = -m - \frac{(n-m)\cdot\varphi(a-\theta)}{[1-\phi(a-\theta)]^2}\cdot\left\{(a-\theta)\cdot[\phi(a-\theta)-1] + (\varphi(a-\theta))^2\right\}$$

Dicha implementación puede encontrarse en el código de `Python` adjunto. Para los datos aquí presentados se obtuvo que la raíz es 20.0552242606 y se llegó a la solución en 3 iteraciones con tiempo de ejecución 0.0212303297718

Método	Estimación	tiempo de ejecución
AlgoritmoME	20.0552242528	0.00895652321867
Gibs Sampler	20.054202679	5.38168182815
Newton-Raphson	20.0552242606	0.0212303297718

Tabla 1: Tabla comparativa de resultados