EP4:
Integração de Monte Carlo por cadeias de Markov

José Paulo Silva Cavalcante

5 de julho de 2020

1 Introdução

O Método MCMC(Markov Chain Monte Carlo) consiste na simulação de uma função distribuição de probabilidade $f(x)$ através da construção de uma cadeia de Markov de modo que a distribuição estacionária seja dada por $\pi(x) = f(x)$.

2 Considerações Iniciais

Estamos interessados em estimar o z, onde:

$$z = \int_0^\infty h(x)g(x)dx \tag{1}$$

por sua vez, $h(x) = I(1 < x < 2)$ e $g(x) \propto gamma(C, x)|cos(Rx)|$ onde gamma tem distribuição exponencial(C), para C = 1.373003882 e R = 1.10297373.
Vamos implementar o MCMC com o Núcleo Normal de média 0 e variância s^2.
Utilizaremos duas probabilidades de aceitação, chamadas de α:

Metrópolis Hasting:

$$\alpha(x^{(k)}, x^{(k-1)}) = min\left(1, \frac{g(x^{(k)})}{g(x^{(k-1)})}\right) \tag{2}$$

Barker:

$$\alpha(x^{(k)}, x^{(k-1)}) = min\left(1, \frac{g(x^{(k)})}{g(x^{(k-1)}) + g(x^{(k)})}\right) \tag{3}$$

Seja $f(x) = gamma(C, x)|cos(Rx)|$, temos que $g(x) = a \cdot gamma(C, x)|cos(Rx)|$, logo, neste caso teremos as seguintes probabilidades de aceitação:

Metropolis:

$$\alpha(x^{(k)}, x^{(k-1)}) = min\left(1, \frac{a \cdot f(x^{(k)})}{a \cdot f(x^{(k-1)})}\right) = min\left(1, \frac{f(x^{(k)})}{f(x^{(k-1)})}\right) \quad (4)$$

Barker:

$$\alpha(x^{(k)}, x^{(k-1)}) = min\left(1, \frac{a \cdot f(x^{(k)})}{a \cdot f(x^{(k-1)}) + a \cdot f(x^{(k)})}\right) = min\left(1, \frac{f(x^{(k)})}{f(x^{(k-1)}) + f(x^{(k)})}\right) \quad (5)$$

Logo, podemos observar que as duas probabilidades de aceitação não possuem a constante de integração a, portanto, não precisamos saber a expressão da função g(x), precisamos somente de uma função proporcional a ela: f(x).

2.1 Algoritmo

```
aceita_metropolis <- function(x_inicial , s){
x_proposto <- rnorm(1,x_inicial ,s)
alpha <- min(1, f(x_proposto)/f(x_inicial))
if (runif(1) < alpha)
x_inicial <- x_proposto
return (x_inicial)
}

aceita_barker <- function(x_inicial ,s){
x_proposto <- rnorm(1,x_inicial ,s)
alpha <- f(x_proposto)/(f(x_inicial) + f(x_proposto))
if (runif(1) < alpha)
x_inicial <- x_proposto
return (x_inicial)
}
```

3 MCMC

O algoritmo de MCMC segue os seguintes passos:
1. Especifique um valor inicial $x^{(0)}$(k=1) e s^2
2. Gere uma amostra x^{k+1} com distribuição Normal($x^{(k)}$,s)
3. Calcule a probabilidade de aceitação (utilizando (2) ou (3))
4. Gere uma u Uniforme(0,1)
5. Se u $\leq \alpha(x^{(k)}, x^{(k+1)})$: a cadeia recebe $x^{(k+1)}$ e $x^{(k)} = x^{(k+1)}$, caso contrário, recebe $x^{(k)}$

Volte para o passo (2), terminamos após n iterações.
Podemos ver que devido as expressões só serão aceitas amostras positivas, pois, ao inserirmos um valor negativo em ambos os alfas, temos zero como resultado, o que é sempre menor do que a amostra gerada pela Uniforme, nos permitindo ficar sempre dentro do domínio de integração.

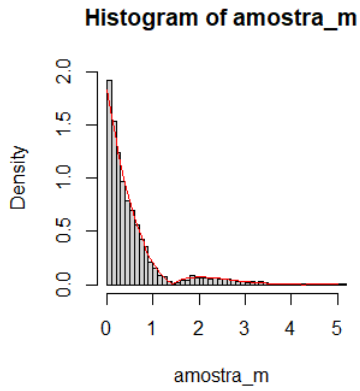

Figura 1: Comportamento do histograma em relação a g(x)

3.1 Algoritmo

```
amostra_metropolis <- function(x, s, n){
amostra <- numeric(n)
for (i in seq_len(n)){
amostra[i] <- x <- aceita_metropolis(x,s)
}
return(amostra)
}

amostra_barker <- function(x, s, n){
amostra <- numeric(n)
rejeitados <- FALSE
for (i in seq_len(n)){
amostra[i] <- x <- aceita_barker(x,s)
}
return(amostra)
}
```

4 Aquecimento da cadeia

Para utilizarmos a cadeia gerada pelo MCMC, precisamos descartar uma faixa inicial de pontos, isso se justifica pelo fato que no passo (2) utilizamos o ponto anterior como média da Normal, logo, não temos independência entre os pontos gerados. Porém, conforme o crescimento do número de iterações, a cadeia começa a ter comportamento Markoviano, onde somente o ponto anterior tem influência no novo a ser gerado. Ou seja, queremos descartar os pontos iniciais

que possuem uma alta correlação linear. Não há consenso na literatura sobre a porcentagem de pontos a serem descartados na cadeia, vamos inicialmente trabalhar com a exclusão de 30% (serão eliminados do início da cadeia), a escolha foi baseada no gráfico de auto correlação:

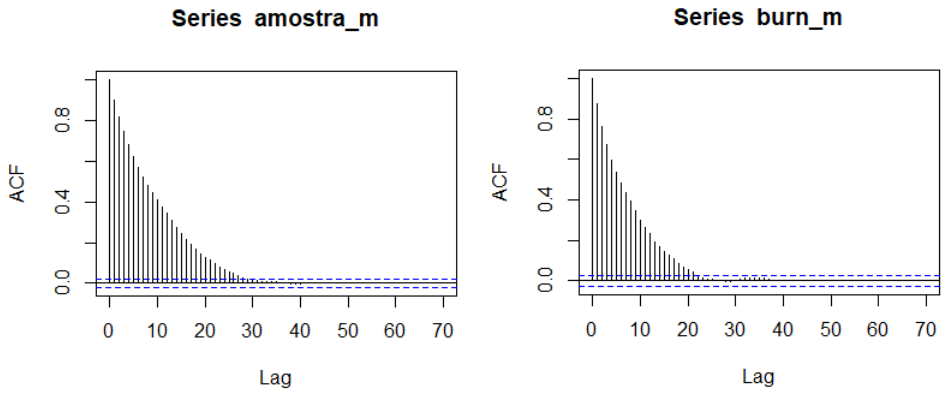

Figura 2: Efeito do Burn-in nas autocorrelações

Podemos observar que há uma redução nas auto correlações das amostras ao retirar 30% dos pontos iniciais, quando aumentamos a porcentagem não há mudança significativa no gráfico gerado.

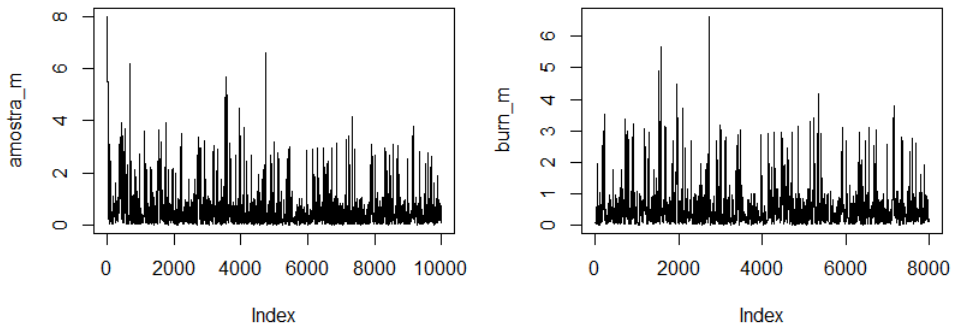

Figura 3: Efeito do Burn-in na cadeia de Markov

Para ilustrar melhor o efeito do burn-in, definimos $x_0 = 10$, logo, no gráfico acima podemos ver que graças à retirada do ponto inicial o burn-in nos oferece

uma cadeia mais estável, independentemente do ponto inicial adotado(que pode ser discrepante em relação à amostra esperada).

5 Calibragem do s

Como antes dito, o núcleo do nosso MCMC tem distribuição Normal($x^{(k)}, s$), no ultimo tópico vimos que podemos definir qualquer ponto $x^{(0)}$ devido ao burn-in, porém, o mesmo não vale para o s, já que temos que escolher um desvio padrão que não permita grandes saltos dentro da cadeia de Markov. Podemos utilizar a proporção de pontos aceitos para cada alfa como ilustração do tamanho do salto da cadeia, adotamos como ideal uma taxa de pelo menos 50%.

6 Estimação de z

Para estimarmos o valor de z, utilizaremos o Teorema do Limite Central, assim como nos EPs anteriores,porém, devemos assumir que todas as amostras presentes na cadeia são independentes(o que não ocorre), além disso, como a função $h(x)$ é uma indicadora para o intervalo (1,2), $\widehat{z}$ será dado por:

$$\widehat{z} = \frac{\sum_0^n h(x_i)}{n} \tag{6}$$

Onde n é a quantidade de pontos x_i da cadeia.

6.1 Algoritmo

```
estimativa <- function(x){sum(h(x))/length(x)}
```

7 Erro

Levando em consideração que queremos um erro de 1% podemos estimar o erro através do intervalo de confiança para $\widehat{z}$ e de acordo com o TLC, temos:

$$e = Z(0.99) \cdot \sqrt{\frac{\sigma}{n}} \tag{7}$$

Para calcularmos um erro relativo de 1% podemos repetir a cadeia M vezes e calcular a esperança das estimativas de todas elas.

8 Implementação

Função que recebe o x inicial, desvio padrão(s),tamanho da cadeia (n) e porcentagem de descarte para o aquecimento
Retorna as estimativas, erros e proporções de aceitos para cada tipo de rejeição.

```
estima_b_m <- function(x0,s,n,p){
n2 <- n*p
amostra_m <- amostra_metropolis(x0,s,n)
amostra_b <- amostra_barker(x0,s,n)
library(dplyr)
prop_aceitos_m <- n_distinct(amostra_m)/length(amostra_m)
prop_aceitos_b <- n_distinct(amostra_b)/length(amostra_b)
burn_m <- amostra_m[n2:n]
burn_b <- amostra_b[n2:n]
erro_m <- dnorm(0.99)*sqrt(var(burn_m)/length(burn_m))
erro_b <- dnorm(0.99)*sqrt(var(burn_b)/length(burn_b))
return(list(estimativa_metropolis = estimativa(burn_m),
erro_metropolis = erro_m, proporcao_metropolis =prop_aceitos_m,
estimativa_barker = estimativa(burn_b), erro_barker=erro_b,
 proporcao_barker = prop_aceitos_b))
```

9 Simulações

Simulamos para os seguintes valores:

```
primeira tentativa
x0 = 1, s = 1, n  = 10000,p =0.2
estima_b_m(1,1,10000,0.2)
estimativa_metropolis
[1]0.06849144

erro_metropolis
[1] 0.001691854

proporcao_metropolis
[1] 0.3436

estimativa_barker
[1] 0.06211724

erro_barker
[1] 0.001907863

proporcao_barker
[1] 0.2161

podemos ver que nessa tentativa houve uma pequena
porcentagem de aceitacao para os dois metodos
```

```
vamos reduzir a variancia para melhorar essa porcentagem:

segunda tentativa

x0 = 1, s = 0.5, n  = 10000,p =0.2
estima_b_m(1,0.5,10000,0.2)

estimativa_metropolis
[1] 0.06874141

erro_metropolis
[1] 0.002044394

proporcao_metropolis
[1] 0.5136

estimativa_barker
[1] 0.06349206

erro_barker
[1] 0.00155439

proporcao_barker
[1] 0.3041
vemos uma boa proporcao para o metodo de metropolis,
mas ainda temos uma baixa proporcao em Barker

terceira tentativa
desta vez, vamos descartar 30% dos pontos gerados,
devido ao burn-in
x0 = 1, s = 0.2, n  = 10000,p =0.3
estima_b_m(1,0.2,10000,0.3)
estimativa_metropolis
[1] 0.0659562

erro_metropolis
[1] 0.0007491115

proporcao_metropolis
[1] 0.749

estimativa_barker
[1] 0.06128484

erro_barker
[1] 0.0005599133
```

```
proporcao_barker
[1] 0.42121
```

10 Resultados

As estimativas e os erros apresentados através de replicações foram:

	Metropolis	Barker
stimativa	0.0639942938659059	0.0654493580599144
Erro	0.00128409221137405	0.0015384302317290

Figura 4: Resultados finais

11 Referências

[1] STERN, Julio Michael. Cognitive Construtivism and the Epistemic Significance of Sharp Statistical Hypotheses in Natural Sciences IME-USP, 2012. Disponível em : https://www.ime.usp.br/ jstern/books/evli.pdf. Acesso em 2 abr. 2020.