10/05/2015

Exercício de Programa 4:

Monte Carlo com Metropolis-Hastings

Instituto de Matemática e Estatística da Universidade de São Paulo

Por

Caio Vinícius Dadauto 7994808

Professor

Julio Michael Stern

1 Monte Carlo

Seja a função,

$$f(t) = 0.25(2^{1.97994808(1-t)})(1 - \sin(0.88084997\pi t)); \quad [0,1] \tag{1}$$

segue duas abordagens distintas para o método de Monte Carlo utilizando cadeias de Markov através do algoritimo Metropolis-Hastings para determinar a integral de $f(t)$. Cada abordagem utiliza uma função de núcleo, assim como apresentado nas demais subseções.

1.1 Núcleo Normal

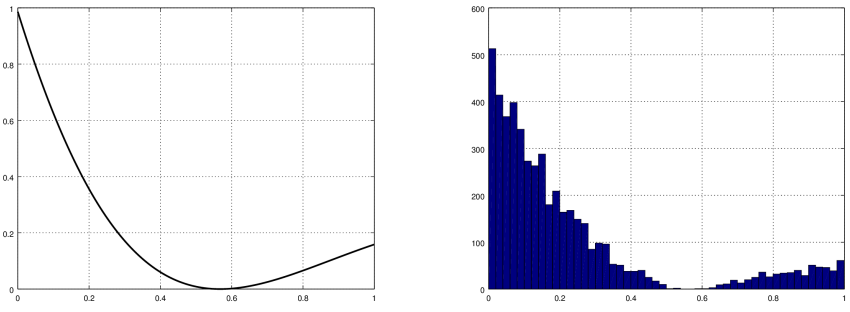
Esta abordagem faz uso de uma distribuição normal para determinar o valor da integral de $f(t)$. A implementação para este caso é apresentada a seguir:

Programa 1: Implementação para MH com núcleo normal

```
1 %Funcao definida no EP
2 function y = ep(t)
3 ···a = 1.97994808;
4 ···b = 0.88084997;
5 ···for i = 1:length(t)
6 ·····y(i) = 0.25*(2^(a*(1-t(i))))*(1 - sin(b*pi*t(i)));
7 ···endfor
8 endfunction
9
```

```
%Metropolis Hastings com nucleo normal
function [x1,a] = MHstepn(x0, sig)
···xp = rem(abs(normrnd(x0, sig)), 1);      %Determina candidato
···accprob = ep(xp)/ep(x0);                 %Probabilidade de aceitacao
···u = rem(abs(normrnd(0, sig)), 1);
···if u <= min(1, accprob)                  %Condicao de aceite
·····x1 = xp;
·····a = 1;
···else
·····x1 = x0;
·····a = 0;
···endif
endfunction

%Inicializacao de variaveis
n     = 5000;
xn    = 0;              %Ponto inicial normal
Xn    = zeros(1,n);     %Vetor para armazenar os pontos determinados pelo MH normla
t     = 0:0.0001:1;
sig   = 1;              %Sigma da gaussiana
accn  = [0  0];         %Vetor para determinar o ratio de aceitacao (normal)

for i = 1:n
···[xn,an] = MHstepn(xn,sig);
···accn    = accn + [an  1.0];
···Xn(i)   = xn;
endfor

In = sum(Xn)/n;
```

A figura 1 apresenta a densidade de probabilidade contraposta com a função $f(t)$, como esperado a densidade de probabilidade possui comportamento semelhante a $f(t)$.



(a) Função $f(t)$. (b) FDP gerada.

Figura 1: FDP contraposta a $f(t)$ com núcleo normal.

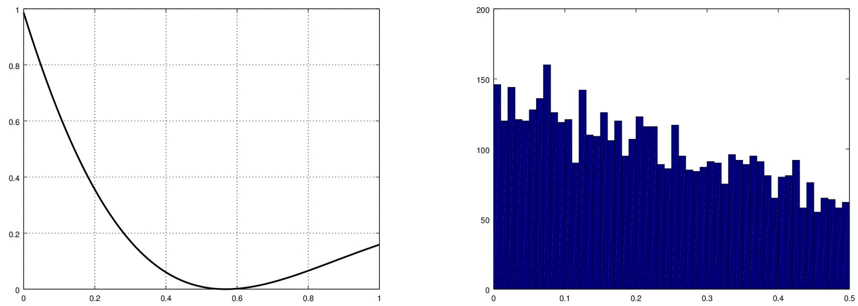
1.2 Núcleo Uniforme

Esta abordagem faz uso de uma distribuição uniforme para determinar o valor da integral de $f(t)$. A implementação para este caso é apresentada a seguir:

Programa 2: Implementação para MH com núcleo uniforme

```
%Funcao definida no EP
function y = ep(t)
···a = 1.97994808;
···b = 0.88084997;
···for i = 1:length(t)
·····y(i) = 0.25*(2^(a*(1-t(i))))*(1 - sin(b*pi*t(i)));
···endfor
endfunction

%Metropolis Hastings com nucleo uniforme
function [x1,a] = MHstepu(x0)
···xp = rem(unifrnd(0, 1), 1);                %Determina candidato
···accprob = ep(xp)/ep(x0);                   %Probabilidade de aceitacao
···u = normrnd(0, 1);
···if u <= min(1, accprob)                    %Condicao de aceite
·····x1 = xp;
·····a = 1;
···else
·····x1 = x0;
·····a = 0;
···endif
endfunction

%Inicializacao de variaveis
n     = 5000;
xu    = 0;              %Ponto inicial uniforme
Xu    = zeros(1,n);     %Vetor para armazenar os pontos determinados pelo MH uniforme
t     = 0:0.0001:1;
sig   = 1;              %Sigma da gaussiana
accu = [0 0];           %Vetor para determinar o ratio de aceitacao (uniforme)

for i = 1:n
···[xu,au] = MHstepu(xu);
···accu    = accu + [au 1.0];
···Xu(i)   = xu;
endfor

Iu = sum(Xu)/n;
```

A figura 2 apresenta a densidade de probabilidade contraposta com a função $f(t)$, como esperado a densidade de probabilidade possui comportamento semelhante a $f(t)$.



(a) Função $f(t)$.

(b) FDP gerada.

Figura 2: FDP contraposta a $f(t)$ com núcleo uniforme.

2 Determinação de integrais a partir da distribuição em $f(x)$

Foram realizadas, ainda, duas integrações a partir da distribuição de $f(x)$ com núcleo normal, são elas as seguinte integrais:

$$\int_0^1 \frac{f(x)}{c} \sin(x) \mathrm{d}x \tag{2}$$

$$\int_0^1 r(\sin(x), s) \frac{f(x)}{c} \mathrm{d}x \tag{3}$$

onde $c = \int_0^1 f(x)\mathrm{d}x$, s é um valor qualquer em $[0, 1]$ e $r(y, x)$ é definido por:

$$r(y,x) = \begin{cases} y & se \quad y \geq x; \\ 0 & se \quad y < x; \end{cases}$$

3 Implementação

Segue a implementação completa para a determinação da integral de $f(t)$ através do método de Monte Carlo e o algoritimo de Metropolis-Hastings para dois núcleos distintos: distribuição normal e uniforme.

Programa 3: Implementação

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Nome: Caio Vinicius Dadauto í            Exerccio de programa 4
% Nusp: 7994808
% Curso: Laboratorio de Programacao e Simulacao
% Turma: Noturno
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
7
8  #!/usr/bin/octave -qf
9
10 1;
11 ignore_function_time_stamp("all");
12 clear all;
13
14 %Funcao definida no EP
15 function y = ep(t)
16 ···a = 1.97994808;
17 ···b = 0.88084997;
18 ···for i = 1:length(t)
19 ·····y(i) = 0.25*(2^(a*(1-t(i))))*(1 - sin(b*pi*t(i)));
20 ···endfor
21 endfunction
22
23 %Metropolis Hastings com nucleo normal
24 function [x1,a] = MHstepn(x0, sig)
25 ···xp = rem(abs(normrnd(x0, sig)), 1);      %Determina candidato
26 ···accprob = ep(xp)/ep(x0);                 %Probabilidade de aceitacao
27 ···u = rem(abs(normrnd(0, sig)), 1);
28 ···if u <= min(1, accprob)                  %Condicao de aceite
29 ·····x1 = xp;
30 ·····a = 1;
31 ···else
32 ·····x1 = x0;
33 ·····a = 0;
34 ···endif
35 endfunction
36
37 %Metropolis Hastings com nucleo uniforme - 0.5
38 function [x1,a] = MHstepu(x0)
39 ···uniform = unifrnd(0, 1);
40 ···xp = rem(abs(uniform - 0.5), 1);         %Determina candidato
41 ···accprob = ep(xp)/ep(x0);                 %Probabilidade de aceitacao
42 ···u = normrnd(0, 1);
43 ···if u <= min(1, accprob)                  %Condicao de aceite
44 ·····x1 = xp;
45 ·····a = 1;
46 ···else
47 ·····x1 = x0;
48 ·····a = 0;
49 ···endif
50 endfunction
51
52 %%                %%
53 %   PARTE A e B   %
54 %%                %%
55 n    = 5000;
56 xn   = 0;              %Ponto inicial normal
57 xu   = 0;              %Ponto inicial uniforme
58 Xn   = zeros(1,n);     %Vetor para armazenar os pontos determinados pelo MH normal
59 Xu   = zeros(1,n);     %Vetor para armazenar os pontos determinados pelo MH uniforme
60 t    = 0:0.0001:1;
```

```
61  sig  = 1;                %Sigma da gaussiana
62  accn = [0  0];           %Vetor para determinar o ratio de aceitacao (normal)
63  accu = [0  0];           %Vetor para determinar o ratio de aceitacao (uniforme)
64  xn   = 0;
65  xu   = 0;
66
67  n  = input("Entre com o valor de iteracoes a serem realizadas: ");
68  printf('\n\n');
69
70  for i = 1:n
71  ···[xn,an] = MHstepn(xn,sig);
72  ···[xu,au] = MHstepu(xu);
73  ···accn    = accn + [an 1.0];
74  ···accu    = accu + [au 1.0];
75  ···Xn(i)   = xn;
76  ···Xu(i)   = xu;
77  endfor
78
79  %Plota figuras
80  figure;
81  plot(t, ep(t), "linewidth", 2, "color", 'k');
82  hold off;
83  figure;
84  hist(Xn, 50);
85  figure;
86  hist(Xu, 50);
87
88  In = sum(Xn)/n;
89  Iu = sum(Xu)/n;
90  printf('------------------> Funcao definida no EP <----------------------\n');
91  printf('MH normal:\n\tValor -> %f\n\tErro -> %f%%\n\tRatio de aceite -> %f\n\n',
92  ···In, 100*abs((In - 0.196064)/0.196064), accn(1)/accn(2));
93  printf('MH uniforme:\n\tValor -> %f\n\tErro -> %f%%\n\tRatio de aceite -> %f\n\n',
94  ···Iu, 100*abs((Iu - 0.196064)/0.196064), accu(1)/accu(2));
95
96  %%                %%
97  %     PARTE C     %
98  %%                %%
99  In = sum(sin(Xn))/n;
100 printf('------------------> Funcao g(x) * sen(x) <----------------------\n')
101 printf('MH normal:\n\tValor -> %f\n\tErro -> %f%%\n\n',
102 ···In, 100*abs((In - 0.196064)/0.196064));
103
104 %%                %%
105 %     PARTE D     %
106 %%                %%
107 s = 0;
108 I = 0;
109
110 printf('-----------------> Funcao r(h(x), s) * g(x)<--------------------\nMH normal:\n');
111 while s < 1
112 ···for i = 1:n
113 ·····h = sin(Xn(i));
114 ·····if h >= s
```

```
115 ·········I += h;
116 ·····endif
117 ···endfor
118 ···I /= n;
119 ···printf('s = %f:\n\tValor -> %f\n\tErro -> %f%%\n',
120 ·····s, I, 100*abs((I - 0.196064)/0.196064));
121 ···s += 0.1;
122 endwhile
```

Neste programa é solicitado ao usuário que entre com o número de iterações e, ainda, com os valores iniciais para o algoritimo de Metropolis-Hasting com núcleo normal e uniforme. Um exemplo da saida do programa é apresentada a seguir:

```
Entre com o valor de iteracoes a serem realizadas: 5000


------------------> Funcao definida no EP <----------------------
MH normal:
     Valor -> 0.222752
     Erro -> 13.611951%
     Ratio de aceite -> 0.470800

MH uniforme:
     Valor -> 0.218472
     Erro -> 11.428888%
     Ratio de aceite -> 0.721000


-------------------> Funcao g(x) * sen(x) <----------------------
MH normal:
     Valor -> 0.208848
     Erro -> 6.520137%


-----------------> Funcao r(h(x), s) * g(x)<---------------------
MH normal:
s = 0.000000:
     Valor -> 0.208848
     Erro -> 6.520137%
```

```
s = 0.100000:
     Valor -> 0.190244
     Erro -> 2.968340%
s = 0.200000:
     Valor -> 0.155718
     Erro -> 20.577778%
s = 0.300000:
     Valor -> 0.119315
     Erro -> 39.144695%
s = 0.400000:
     Valor -> 0.097789
     Erro -> 50.123888%
s = 0.500000:
     Valor -> 0.089458
     Erro -> 54.373133%
s = 0.600000:
     Valor -> 0.088200
     Erro -> 55.014923%
s = 0.700000:
     Valor -> 0.075920
     Erro -> 61.278121%
s = 0.800000:
     Valor -> 0.028158
     Erro -> 85.638134%
s = 0.900000:
     Valor -> 0.000006
     Erro -> 99.997128%
s = 1.000000:
     Valor -> 0.000000
     Erro -> 99.999999%
```