26/05/2015

Exercício de Programa 3:

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

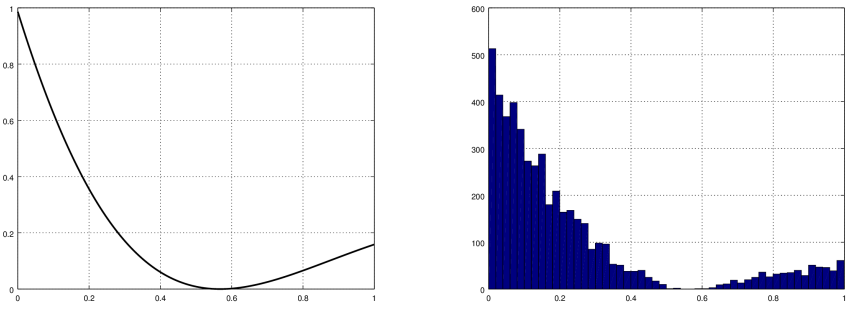
Esta abordagem faz uso de uma distribuição normal para determinar o valor da integral de $f(t)$. A implementação para este caso é apresentada a seguir:

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.

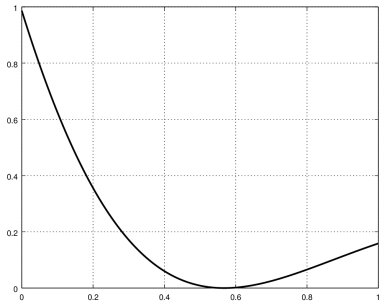
1.2 Núcleo Uniforme

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:

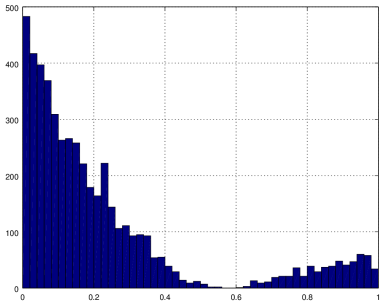
Programa 2: Implementação para MH com núcleo uniforme

```
%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()
···xp = unifrnd(0, 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] = MHstepn(xu,sig);
···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 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

```
1  #!/usr/bin/octave -qf
2
3  1;
4  ignore_function_time_stamp("all");
5  clear all;
6
7  %Funcao definida no EP
8  function y = ep(t)
9  ···a = 1.97994808;
10 ···b = 0.88084997;
11 ···for i = 1:length(t)
12 ·····y(i) = 0.25*(2^(a*(1-t(i))))*(1 - sin(b*pi*t(i)));
13 ···endfor
14 endfunction
15
16 %Metropolis Hastings com nucleo normal
17 function [x1,a] = MHstepn(x0, sig)
18 ···xp = rem(abs(normrnd(x0, sig)), 1);      %Determina candidato
19 ···accprob = ep(xp)/ep(x0);                 %Probabilidade de aceitacao
20 ···u = rem(abs(normrnd(0, sig)), 1);
21 ···if u <= min(1, accprob)                  %Condicao de aceite
22 ·····x1 = xp;
23 ·····a = 1;
24 ···else
25 ·····x1 = x0;
26 ·····a = 0;
27 ···endif
```

```
endfunction

%Metropolis Hastings com nucleo uniforme
function [x1,a] = MHstepu()
···xp = unifrnd(0, 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;
xn   = 0;            %Ponto inicial normal
xu   = 0;            %Ponto inicial uniforme
Xn   = zeros(1,n);   %Vetor para armazenar os pontos determinados pelo MH normla
Xu   = zeros(1,n);   %Vetor para armazenar os pontos determinados pelo MH uniforme
t    = 0:0.0001:1;
sig  = 1;            %Sigma da gaussiana
accn = [0 0];        %Vetor para determinar o ratio de aceitacao (normal)
accu = [0 0];        %Vetor para determinar o ratio de aceitacao (uniforme)

xn = input("Entre com o valor inicial entre 0 e 1 para o nucleo normal: ");
xu = input("Entre com o valor inicial entre 0 e 1 para o nucleo uniforme: ");
n  = input("Entre com o valor de iteracoes a serem realizadas: ");
for i = 1:n
···[xn,an] = MHstepn(xn,sig);
···[xu,au] = MHstepn(xu,sig);
···accn   = accn + [an 1.0];
···accu   = accu + [au 1.0];
···Xn(i)  = xn;
···Xu(i)  = xu;
endfor

%Plota figuras
figure;
plot(t, ep(t), "linewidth", 2, "color", 'k');
hold off;
figure;
hist(Xn, 50);
figure;
hist(Xu, 50);

In = sum(Xn)/n;
Iu = sum(Xu)/n;
printf('MH normal:\n\tValor -> %f\n\tErro -> %f%%\n\nRatio -> %f\n\n',
···In, 100*abs((In - 0.196064)/0.196064), accn(1)/accn(2));
printf('MH uniforme:\n\tValor -> %f\n\tErro -> %f%%\n\nRatio --> %f\n\n',
···Iu, 100*abs((Iu - 0.196064)/0.196064), accu(1)/accu(2));
```

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 inicial entre 0 e 1 para o nucleo normal: 0
Entre com o valor inicial entre 0 e 1 para o nucleo uniforme: 0
Entre com o valor de iteracoes a serem realizadas: 5000
MH normal:
     Valor -> 0.219046
     Erro -> 11.721825%
     Ratio -> 0.461600

MH uniforme:
     Valor -> 0.223490
     Erro -> 13.988296%
     Ratio --> 0.458400
```