Exercício 03

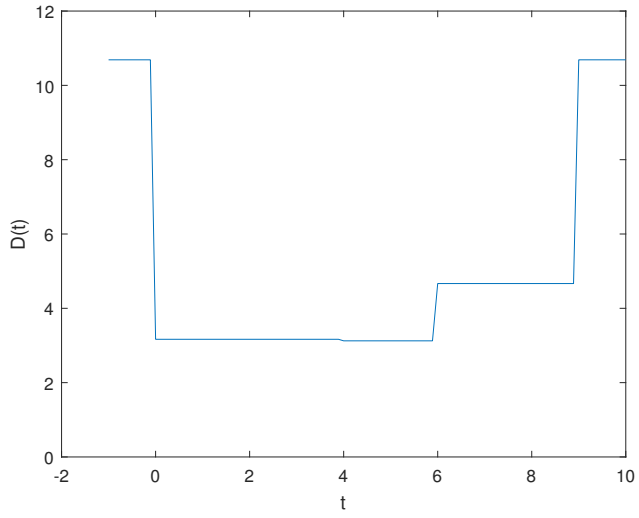
Renan Salles de Freitas
CPE 723 - Otimização Natural

2 de abril de 2018

Exercício 1.a. O código MatLab para gerar o gráfico está abaixo:

```
X = [0,4,6,9];
t = linspace(-1,10,100);
D = zeros(1,length(t));

for i = 1:length(t)
    ti = t(i);
    X1 = X(X <= ti);
    X2 = X(X > ti);
    m1 = mean(X1);
    m2 = mean(X2);
    D(i) = 0.25 * ( sum((X1 - m1).^2) + sum((X2 - m2).^2) );
end
```



Exercício 1.b. No algoritmo *Deterministic Annealing*, temos que a condição de partição é dada pela fórmula:

$$p_{y|x} = \frac{e^{-d_{xy}/T}}{\mu_x}$$

Onde d_{xy} é a distância quadrática entre o centróide y e o dado x:

$$d_{xy} = \begin{bmatrix} (x_1 - y_1)^2 & (x_2 - y_1)^2 & (x_3 - y_1)^2 & (x_4 - y_1)^2 \\ (x_1 - y_2)^2 & (x_2 - y_2)^2 & (x_3 - y_2)^2 & (x_4 - y_2)^2 \end{bmatrix} = \begin{bmatrix} 9 & 1 & 9 & 36 \\ 11.56 & 0.36 & 6.76 & 31.36 \end{bmatrix}$$

E $\mu_x = \sum_y e^{-d_{xy}/T}$.
Dessa forma, podemos calcular a matriz de probabilidade conjunta:

$$p_{y|x} = \begin{bmatrix} 0.9282 & 0.3452 & 0.0962 & 0.0096 \\ 0.0718 & 0.6548 & 0.9038 & 0.9904 \end{bmatrix}$$

```
X = [0,4,6,9];
Y = [3 3.4]';
T = 1.0;

d_xy = bsxfun(@minus, X, Y).^2;

e = exp(-d_xy/T);
mu = sum(e);
p_xy = bsxfun(@rdivide, e, mu);


% p_xy =
%
%     0.9282    0.3452    0.0962    0.0096
%     0.0718    0.6548    0.9038    0.9904
```

Exercício 1.c. Podemos calcular D a partir do somatório:

$$\begin{aligned} D &= \sum_x p(x) \sum_y p(y|x) d(x,y) \\ &= \frac{1}{4}(p_{1|1}d(1,1) + p_{2|1}d(1,2) + p_{2|1}d(2,1) + p_{2|2}d(2,2) + \\ &+ p_{3|1}d(3,1) + p_{3|2}d(3,2) + p_{4|1}d(4,1) + p_{4|2}d(4,2)) \\ &= 12.0361 \end{aligned}$$

```
X = [0,4,6,9];
Y = [3 3.4]';
T = 1.0;

d_xy = bsxfun(@minus, X, Y).^2;

e = exp(-d_xy/T);
mu = sum(e);
p_xy = bsxfun(@rdivide, e, mu);


D = 0.25 * sum(sum(p_xy.*d_xy));
```

Exercício 1.d. A condição de centróide é regida pela fórmula:

$$y_k = \frac{\sum_x p_{k|x} x}{\sum_x p_{k|x}}$$

Logo:

$$\begin{aligned} y_1 &= \frac{\sum_x p_{1|x} x}{\sum_x p_{1|x}} \\ &= \frac{p_{1|1}x_1 + p_{1|2}x_2 + p_{1|3}x_3 + p_{1|4}x_4}{p_{1|1} + p_{1|2} + p_{1|3} + p_{1|4}} \\ &= 1.4822 \end{aligned}$$

$$\begin{aligned} y_2 &= \frac{\sum_x p_{2|x} x}{\sum_x p_{2|x}} \\ &= \frac{p_{2|1}x_1 + p_{2|2}x_2 + p_{2|3}x_3 + p_{2|4}x_4}{p_{2|1} + p_{2|2} + p_{2|3} + p_{2|4}} \\ &= 6.4698 \end{aligned}$$

```
X = [0,4,6,9];
Y = [3 3.4]';
T = 1.0;

d_xy = bsxfun(@minus, X, Y).^2;

e = exp(-d_xy/T);
mu = sum(e);
p_xy = bsxfun(@rdivide, e, mu);


Y = sum(p_xy.*X,2)./sum(p_xy,2);
```

Exercício 1.e. Para este exercício, usamos o código de Matlab abaixo:

```
X = [0,4,6,9];
Y = [3 3.4]';
T = 0.1;

d_xy = bsxfun(@minus, X, Y).^2;

e = exp(-d_xy/T);
mu = sum(e);
p_xy = bsxfun(@rdivide, e, mu);

D = 0.25 * sum(sum(p_xy.*d_xy));

Y = sum(p_xy.*X,2)./sum(p_xy,2);
```

E obtemos:

$$p_{y|x} = \begin{bmatrix} 1 & 0.0017 & 0 & 0 \\ 0 & 0.9983 & 1 & 1 \end{bmatrix}$$

$$D = 11.8703$$

$$Y = \begin{bmatrix} 0.0066 \\ 6.3346 \end{bmatrix}$$

Exercício 1.f. Para este exercício, usamos o código de Matlab abaixo:

```
X = [0,4,6,9];
Y = [3 3.4]';
T = 50;

d_xy = bsxfun(@minus, X, Y).^2;

e = exp(-d_xy/T);
mu = sum(e);
p_xy = bsxfun(@rdivide, e, mu);

D = 0.25 * sum(sum(p_xy.*d_xy));

Y = sum(p_xy.*X,2)./sum(p_xy,2);
```

E obtemos:

$$p_{y|x} = \begin{bmatrix} 0.5128 & 0.4968 & 0.4888 & 0.4768 \\ 0.4872 & 0.5032 & 0.5112 & 0.5232 \end{bmatrix}$$

$$D = 13.0881$$

$$Y = \begin{bmatrix} 4.6635 \\ 4.8344 \end{bmatrix}$$

Exercício 1.g. Para temperaturas baixas ($T = 0.1$), o problema cai em uma situação de *hard-clustering*, isto é, a matriz de probabilidades conjuntas da condição de partição ($p_{y|x}$) possui apenas valores 0 e 1, não havendo a estocacidade do algoritmo e os centróides ficam em $y = [0 \quad 6.333]$, já que o algoritmo separa os dados mais próximos para os clusters: $y_1 = 0$ e $y_2 = (4 + 6 + 9)/3 = 6.333$.

Para temperaturas altas, o problema está próximo da máxima entropia, e os valores da matriz de probabilidades conjuntas ficam em torno de 0.5 para todo $y|x$. Isso pode ser

percebido pelo decaimento rápido das exponenciais. Neste caso, há a formação de apenas um cluster: $y_1 = y_2 = \sum_x X/4 = 4.75$.

Para o caso de temperaturas intermediárias $T = 1$,, podemos observar o comportamento de *soft-clustering* para o caso $p_{y|x_2}$, na matriz de probabilidade conjunta. Porém a temperatura se assemelha mais ao caso quando $T = 0.1$.

Exercício 2. Para este exercício vamos utilizar a função Rosenbrock para N = 20:

$$f(\mathbf{x}) = \sum_{i=1}^{N-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$$

O mínimo está em $x_{min} = \mathbf{1}$ e $J_{min} = 0$. O código MatLab para SA está abaixo:

```
% Exercicio 2

clear all
clc

B = @(n,t) exp(-n/t);

number_of_variables = 20;
number_of_iteration = 30000;
number_of_temperature_iteration = 10;
epsilon = 0.05;
T0 = 1;
T = T0;
x0 = 1.5*ones(1, number_of_variables);
x = x0;
counter = 1;
counter_temperatures = 1;
Jmin = J(x(counter,:));
xmin = x(counter,:);
while counter_temperatures <= number_of_temperature_iteration
    for i = 1:number_of_iteration
        r = randn(1,number_of_variables);
        xk = x(counter,:) + epsilon * r;
        Jxk = J(xk);
        dJ = Jxk - J(x(counter,:));
        counter = counter + 1;
        if dJ < 0
            x = [x ; xk];
        else
            a = rand;
            if B(dJ,T) > a
                x = [x ; xk];
            else
                x = [x ; x(counter - 1,:)];
            end
```

```
        end
        if Jxk < Jmin
            Jmin = Jxk;
            xmin = xk;
        end
    end
    counter_temperatures = counter_temperatures + 1;
    T = T0 / log2(counter_temperatures + 1);
    number_of_iteration = number_of_iteration / log2(
       counter_temperatures + 1);
end
```

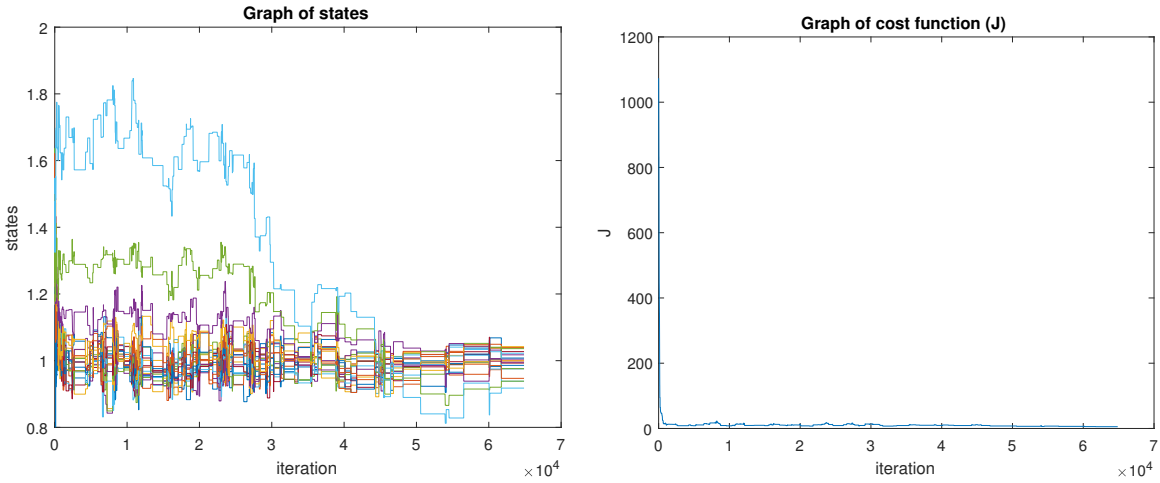
A função custo:

```
function y = J(x)
n = size(x);
y = zeros(n(1),1);
for i=1:n(2)-1
    y = y + 100*(x(:,i+1) - x(:,i).^2).^2 + (1 - x(:,i)).^2;
end
```

Encontramos:

$$x_{min} = \begin{bmatrix} 0.9757 \\ 0.9996 \\ 1.0407 \\ 1.0047 \\ 1.0362 \\ 1.0222 \\ 1.0272 \\ 1.0079 \\ 1.0393 \\ 1.0419 \\ 1.0175 \\ 0.9761 \\ 0.9383 \\ 0.9927 \\ 0.9859 \\ 0.9470 \\ 0.9995 \\ 0.9951 \\ 0.9385 \\ 0.9177 \end{bmatrix}$$

e $J_{min} = 5.4474$.



Verificou-se que o SA teve certa dificuldade em encontrar o mínimo global pelo grande número de mínimos locais. O algoritmo desenvolvido possui mais iterações nas temperaturas altas e menos nas temperaturas baixas. Dessa forma, o algoritmo realiza 30.000 iterações para $T_0 = 1$ e reduz oo número de iterações logaritmicamente, como a temperatura. Além disso, o algoritmo é sensível à perturbação, resfriamento, e condição inicial.