Assignment 4

Zaineb Amor
Erik Engelhardt

May 2019

1 Introduction

In this assignment, we complete the backward function of the MarkovChain class. The @MarkovChain/backward method has to work for both finite and infinite duration markov chains.

2 Backward algorithm

We implemented the backward algorithm in @MarkovChain. The function is included in the zip-file but here is the code as well.

```
1  betaHat=backward(mc,pX,c)
2      T=size(pX,2);%Number of observations
3      N= mc.nStates; %Number of possible states
4      A = mc.TransitionProb;
5      finite = finiteDuration(mc);
6
7      %Init
8      if finite
9          betaHat = A(:, end)/(c(T)*c(T + 1));
10     else
11         betaHat = 1/c(T);
12     end
13
14     %Backward step
15     for t = T - 1 : -1 : 1
16         beta_temp = A(:, 1 : N)*(pX(:, t + 1).*betaHat(:,
               1))/c(t);
17         betaHat = [beta_temp betaHat]
18     end
19 end
```

2.1 Results of the backward algorithm

To check the backward algorithm we used a markov chain defined with q = [1; 0] and A = [0.9, 0.1, 0; 0, 0.9, 0.1].
pX used here was generated using the prob method of GaussD as pX = prob(B, x) where B is the out output distriution wich is a scalar gaussian with mean 1 = 0 and standard deviation 1 = 1 for the first state, and with 2 =3 and 2 =2 for the second state and x = [-0.2, 2.6, 1.3].
Finaly c= [1.0000 0.1625 0.8266 0.0581].

The code to verify the sanity of the backward algorithm is the following:

```
%addpath ('@MarkovChain')
%addpath ('@GaussD')
%addpath ('@HMM')
q = [1; 0];
A = [0.9, 0.1, 0; 0, 0.9, 0.1];
b1 = GaussD('Mean', 0, 'StDev', 1);
b2 = GaussD('Mean', 3, 'StDev', 2);
B = [b1, b2];
MC = MarkovChain(q, A);
x = [-0.2, 2.6, 1.3];
[pX, scale] = prob(B, x);
c = [1, 0.1625, 0.8266, 0.0581];
betaHat = backward(MC, pX, c)
```

The result was coherent with the instructions:
betaHat =
[1.0003 1.0393 0
8.4182 9.3536 2.0822]