MATH4406 (Control Theory), HW3

Trent Skorka - s42897282

September 16, 2014

1. **Implement the backward induction algorithm by hand for problem data of your choice, with time horizon $N = 5$.**

   We define the parameters for the model: $N = 5, M = 2, c(u) = u, g(u) = 0, f(u) = 12u, K = 3, h(u) = 3u$

   $$p(j) = \begin{cases} 1/2 & j = 0 \\ 1/6 & j = 1 \\ 1/3 & j = 2 \end{cases}$$

   Thus we find

   $$O(a) = \begin{cases} 3 + a & a > 0 \\ 0 & a = 0 \end{cases}$$

   And, using the definitions given in Section 3.2.1 of [Puterman]

   $$F(u) = \begin{cases} 0 & u = 0 \\ 6 & u = 1 \\ 10 & u = 2 \end{cases}$$

   Rewards:

	a		
s	**0**	**1**	**2**
0	0	-1	-1
1	3	0	X
2	4	X	X

   Here, 'X' denotes an impossible action (as the action would result in current stock exceeding the maximum capacity, 2, of the problem).

   Transition probabilities:

	a		
s	**0**	**1**	**2**
0	1	0	0
1	1/2	1/2	0
2	1/3	1/6	1/2

We then implement the backward induction algoritm:

1. Set $t = N = 5$ and $r_5(s) = u_5^*(s) = 0$ because $g(s) = 0$.

2. Since $t \neq 1$, set $t = 4$

$$u_4^*(s) = \max_{a \in A_s} \{r(s,a) + \sum_{j=0}^{2} p(j|s,a) u_5^*(j)\} = \max_{a \in A_s} \{r(s,a)\}$$

s	$u_4^*(s)$	$A_{s,4}^*$
0	0	0
1	3	0
2	4	0

3. Since $t \neq 1$, set $t = 3$

$$u_3^*(s) = \max_{a \in A_s} \{r(s,a) + \sum_{j=0}^{2} p(j|s,a) u_4^*(j)\}$$

	$u_3^*(s,a)$				
s	$a=0$	$a=1$	$a=2$	$u_3^*(s)$	$A_{s,3}^*$
0	0	0..5	1.5	1.5	2
1	4.5	2.5	X	4.5	0
2	6.5	X	X	6.5	0

4.Since $t \neq 1$, set $t = 2$

	$u_2^*(s,a)$				
s	$a=0$	$a=1$	$a=2$	$u_2^*(s)$	$A_{s,2}^*$
0	1.5	2	3.5	3.5	2
1	6	4.5	X	6	0
2	8.5	X	X	8.5	0

5.Since $t \neq 1$, set $t = 1$

	$u_1^*(s,a)$				
s	$a=0$	$a=1$	$a=2$	$u_1^*(s)$	$A_{s,1}^*$
0	3.5	3.75	$\frac{195}{36}$	$\frac{195}{36}$	2
1	7.75	$\frac{231}{36}$	X	7.75	1
2	$\frac{375}{36}$	X	X	$\frac{375}{36}$	0

6. Since $t = 1$, stop.

We thus find the expected total reward function v_5^* for the optimal policy, π^*

s	$d_1^*(s)$	$d_2^*(s)$	$d_3^*(s)$	$d_4^*(s)$	$v_5^*(s)$
0	2	2	2	0	$\frac{195}{36}$
1	0	0	0	0	7.75
2	0	0	0	0	$\frac{375}{36}$

This is a nonstationary (σ, Σ) policty, where $\sigma = 1$ and $\Sigma_n = 2$ for $n = 1, 2, 3$, and $\Sigma_4 = 0$.

2. (σ, Σ) **policies are known to be optimal when the ordering cost is**

$$O(u) = cu + 1\{u > 0\}K$$

Implement (in software) the backward induction algorithm for 10 different inventory control problems with a time horizon of $N = 15$. Verify that the resulting policy is indeed a threshold (σ, Σ) policy for the 10 problems.

We consider 10 different cases of the Inventory Control Problem. In each of these cases, the time horizon is $N = 15$, and we set $g(u) = 0$. The variables are presented in the table below, as well as the values describing the optimal policy. Here, the optimal policy is

$$\pi^* = (d_1^*(s), \ldots, d_{13}^*, d_{14}^*)$$

$$\text{Where } d_1^* = d_2^* = \cdots = d_{13}^* = \begin{cases} \Sigma_t - s & s < \sigma_t \\ 0 & s > \sigma_t \end{cases}$$

$$\text{and } d_{14}^* = \begin{cases} \Sigma_{14} - s & s < \sigma_{14} \\ 0 & s > \sigma_{14} \end{cases}$$

So π^* is a threshold (σ, Σ) policy. The values found in the table below for M, K, c, f, h and the probabilities, were randomly generated and the code was then implemented to find $\sigma_t, \Sigma_t, \sigma_{14}, \Sigma_{14}$. The Matlab code used to find the optimal policy is presented in *Appendix 1* .

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
M	3	2	4	3	6	4	4	4	2	2
K	2	4	4	1	4	5	2	1	4	3
c	1	2	2	3	3	1	2	2	3	1
f	10	12	20	11	10	7	12	6	28	8
h	1	2	2	1	2	2	3	1	2	3
$\mathbf{p}(j=0)$	0.3	0.4	0.2	0.1	0.2	0.1	0.2	0.3	0.4	0.2
$\mathbf{p}(j=1)$	0.1	0.3	0.1	0.3	0.2	0.3	0.4	0.2	0.4	0.4
$\mathbf{p}(j=2)$	0.4	0.2	0.3	0.4	0.1	0.4	0.2	0.2	0.2	0.3
$\mathbf{p}(j=3)$	0.2	0.1	0.4	0.2	0.2	0.2	0.1	0.1	0	0.1
$\mathbf{p}(j=4)$	0	0	0	0	0.3	0	0.1	0.2	0	0
σ_t	2	1	3	2	2	1	1	2	2	1
Σ_t	3	2	3	3	4	3	2	3	2	2
σ_{14}	2	1	2	2	1	1	1	1	1	1
Σ_{14}	2	1	3	2	2	0	1	2	2	2

It is interesting to note that all but the 10^{th} problem were nonstationary threshold policies. Nevertheless, it can be seen that all 10 problems chosen do indeed staisfy a threshold (σ, Σ) policy.

Find some ordering cost $O(.)$, different from the above, that yields a non threshold policy. Demonstrate that the output of your backward induction algorithm yields a non-threshold policy.

Consider the case where $O(u) = 4 + e^{2u} - u$ for $u > 0$. Take $M = 4, f(u) = 20u, g(u) = 0, h(u) = 2u$, and

$$p(j) = \begin{cases} 0.2 & j = 0 \\ 0.1 & j = 1 \\ 0.3 & j = 2 \\ 0.4 & j = 3 \end{cases}$$

Implementing the algorithm using the matlab code, using simply $N = 5$, we find

s	$d_1^*(s)$	$d_2^*(s)$	$d_3^*(s)$	$d_4^*(s)$	$v_3^*(s)$
0	1	1	1	1	23.0785
1	1	1	1	1	39.4370
2	1	1	1	0	53.6680
3	1	0	0	0	65.2037
4	0	0	0	0	75.5927

This is obviously not a threshold (σ, Σ) policy. Cosider the first decision epoch; $a = 1$ for $s < 4$ and $a = 0$ for $s = 4$. The action does not ensure the stock after ordering is Σ if the stock before ordering is less than some other σ.

3. **Look at Theorem 4.4.2 on page 89 together with part (a) or Proposition 4.4.3 on the next page. Package these into a single theorem that states that when $\mathcal{S}$ is finite or countable and $\mathcal{A}_s$ are all finite then there exists a deterministic Markovian policy that is optimal. Write out the proof. Be precise and neat.**

Theorem: If $\mathcal{S}$ is finite or countable and $\mathcal{A}_s$ are all finite then there exists a deterministic Markovian policy that is optimal.

Proof: Suppose there exists an $a' \in \Pi^{MD}$ which satisfies Therorem 4.4.1 of [Puterman]. That is, it satisfies

$$r_t(s_t, a') + \sum_{j \in S} p_t(j|s_t a) u_{t+1}^*(j) = \max_{a \in A_{s_t}} \left\{ r_t(s_t, a) + \sum_{j \in S} p_t(j|s_t, a) u_{t+1}^*(j) \right\} \tag{1}$$

For this to exist, we require $u_t^*(h_t) = u_t^*(s_t)$. This can be shown by induction. Fist note $u_N^*(h_N) = u_N^*(h_{N-1}, a_{N-1}, s) = r_N(s)$. Assume this is valid for $n = t+1, \ldots, N$. We find

$$\begin{aligned} u_t^*(h_t) &= \sup_{a \in A_{s_t}} \left\{ r_t(s_t, a) + \sum_{j \in S} p_t(j|s_t, a) u_{t+1}^*(h_t, a, j) \right\} \\ &= \sup_{a \in A_{s_t}} \left\{ r_t(s_t, a) + \sum_{j \in S} p_t(j|s_t, a) u_{t+1}^*(j) \right\} \text{ by the induction hypothesis} \end{aligned}$$

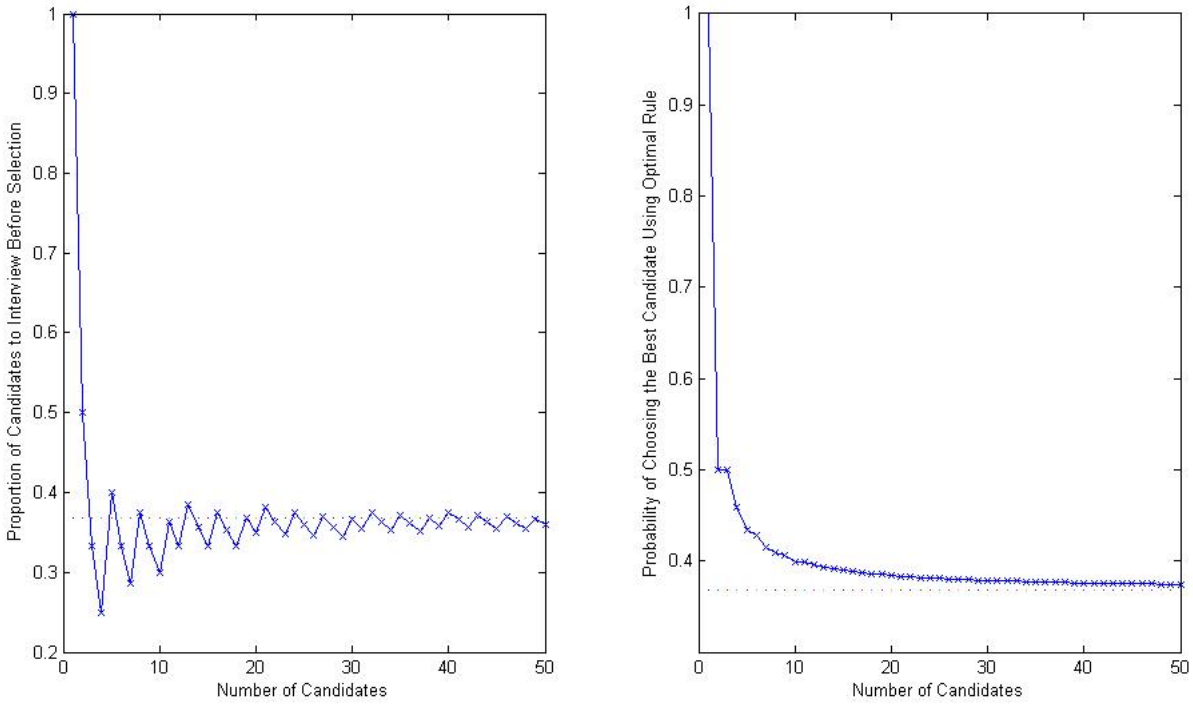
This is clearly dependent on h_t only through s_t, so we have $u_t^*(h_t) = u_t^*(s_{t-1})$. Thus, by Theorem 4.4.1b and Theorem 4.3.3b, (1) is indeed true and a' is an optimal policy which exists, as $\mathcal{A}_s$ is finite. Note that this is no longer a history-dependent policy, as it was in Therorem 4.4.1.

Thus, $a' \in \Pi^{MD}$ is an optimal policy.

4. **Read the analysis of "The Secretary Problem" in Section 4.6.4.**

(a) **Reproduce Figure 4.6.2 (supply your code).**

The Matlab used for this diagram is shown in *Appendix 2.*



The dotted line is placed at e^{-1}, so it can be seen that both graphs converge to e^{-1}, as in Figure 4.6.2 from [Puterman].

The code itself requires setting $\tau(1) = 1$, and $u_t^*(1)|_{N=1} = 1$. This is explained in [Puterman], as the formulas are only valid for $N > 2$.

Note that $u_t^*(0) = u_t^*(1) = u_\tau^*(1)$ from (4.6.10), which is what was used in the code.

(b) **Fill in any missing details in the analysis on page 102, yielding the Ne^{-1} rule. This is a good "rule of thumb" for life. How would you use it elsewhere?**

We begin with equation (4.6.12). From here, we evaluate the integral

$$\begin{aligned}\int_{\tau(N)}^{N} \frac{1}{x}dx &= \log(x)|_{\tau(N)}^{N} \\ &= \log(N) - \log(\tau(N)) \\ &= \log\left(\frac{N}{\tau(N)}\right) \approx 1 \text{ By (4.6.12)}\end{aligned}$$

This can be manipulated to show

$$\begin{aligned}\log\left(\frac{N}{\tau(N)}\right) &\approx 1 \\ \frac{N}{\tau(N)} &\approx e \\ \frac{\tau(N)}{N} &\approx e^{-1}\end{aligned}$$

Taking the limit of both sides yields

$$\begin{aligned}\lim_{N\to\infty} \frac{\tau(N)}{N} &\approx \lim_{N\to\infty} e^{-1} \\ &= e^{-1}\end{aligned}$$

Recall (4.6.10) : $u_t^*(0) = u_t^*(1) = \cdots = u_\tau^*(0) = u_\tau^*(1)$ and $u_t^*(0) = \frac{t}{N}\left[\frac{1}{t} + \frac{1}{t+1} + \cdots + \frac{1}{N-1}\right]$. Thus,

$$\begin{aligned}u_t^*(0) = u_t^*(1) = u_{\tau(N)}^*(0) &= \frac{\tau(N)}{N}\left[\frac{1}{t} + \frac{1}{t+1} + \cdots + \frac{1}{N-1}\right] \\ &= \frac{\tau(N)}{N}\log\left(\frac{N}{\tau(N)}\right) \\ &= \frac{\tau(N)}{N}\log(e) \\ &= \frac{\tau(N)}{N} \to e^{-1}\end{aligned}$$

An example of how this Ne^{-1} rule could be used in real life is in a dating service. This is quite analogous to hiring an employee - consider the first 36.8% of potential partners, and then choose the next most compatable.

(c) **Think (or look up) generalisations or modifications of the Secretary Problem (there has been much research on this). Briefly (in one paragraph) present one such modified problem).**

A generalisation of the Secretary Problem presented by Shoou-Ren Hsiau and Jiing-Ru Yang, of the National Changhua University of Education, was on in which applicants were interviewed in groups. The applicants were randomly divided into these N groups, with the ith group containing l_i members. Analogous with the standard model, the interviewer must decide whether to select an employee from the current group, or to dismiss the group and interview the next one. They cannot recall previous applicants, and they do know the relative ranks of previous applicants. Assuming an optimal strategy of rejecting the first $r-1$ applicants (for some r) and selecting the next best applicant, Hsiau and Yang found the threshold r is given by

$$r = \min\left\{n| \sum_{k=n+1}^{N} \frac{l_k}{b_{k-1}} \leq 1\right\}, \text{ where } b_k = \sum_{i=1}^{k} l_i$$

APPENDIX 1

Matlab Code for Question 2

```
clear all
clc
%Define variables
Epochs=15;        %Number of time steps/decision epochs
Max_Stock=3;     %Maximum Stock Capacity
K=2;             %Ordering cost
Cost=1;          %Ordering cost of u units
Reward=10;       %Reward for sale of u units
g=0;             %Reward at last time step
Hold_cost=1;     %Holding cost of u units
Probs= zeros(Max_Stock+1,1); %Probabilities
Probs(1) = 0.3;              %j=0
Probs(2) = 0.1;              %j=1
Probs(3) = 0.4;              %j=2
Probs(4) = 0.2;              %j=3
Probs(5) = 0;                %j=4
Probs        %View probabilites
sum(Probs') %Check probabilities add to 1

%Finding expected reward with stock of u units
for i=0:Max_Stock
    for j=0:max(0,i-1)
        F1(j+1)=Reward*j*Probs(j+1);
    end
    q=zeros(1,Max_Stock+1);
    for j=i:Max_Stock
        q(j+1)=Probs(j+1);
    end
    F(i+1)=sum(F1)+sum(q)*Reward*i;
end


%Calculate reward matrix
for s=0:Max_Stock %Stock
    for a=0:Max_Stock %Actions
        if s+a>Max_Stock
            r(s+1,a+1)=-100; %Set arbitrarily large negative as it is
                             %not an acceptable action
        elseif a==0
            r(s+1,a+1)=F(s+a+1)-Hold_cost*(s+a);
        else
            r(s+1,a+1)=F(s+a+1)-(K+Cost*a+Hold_cost*(s+a));
        end
    end
end


```

```
47
48      %Calculate transition probabilities
49 -    for j=0:Max_Stock %Stock at next time step
50 -        for S=0:Max_Stock %Current stock = s+a < M
51 -            if S<j
52 -                P(S+1,j+1)=0;
53 -            elseif j==0
54 -                ans=zeros(1,Max_Stock);
55 -                for n=S:Max_Stock
56 -                    ans(n+1)=Probs(n+1);
57 -                end
58 -                P(S+1,j+1)=sum(ans);
59 -            else
60 -                P(S+1,j+1)=Probs(S-j+1);
61 -            end
62 -        end
63 -    end
64
65
66      %Begin Algorithm
67      %u_N(s)=0, as g(s)=0
68
69      %Initiate algorithm for t=N-1
70 -    r1=max(r');
71 -    for n=1:Max_Stock+1
72 -        u_max(n)=r1(n);
73 -    end
74 -    [num idx]=max(r');
75 -    opt=ind2sub(size(r'),idx)-1; %Find optimal actions
76 -    for n=1:Max_Stock+1
77 -        A(n,Epochs-1)=opt(n);
78 -    end
79
80      %Implement algorithm for t<N-1
81 -     for i=2:Epochs-1
82 -         t=Epochs-i; %Decreasing t at each time step, stopping at t=1
83 -        for s=0:Max_Stock
84 -            for a=0:Max_Stock
85 -                if s+a>Max_Stock
86 -                   u(s+1,a+1)=-100; %Set as arbitrarily large negative as it
87                                       %is not a possible action
88 -                else
89 -                    sum1=0;
90 -                    for j=0:Max_Stock
91 -                        sum1=sum1+P(s+a+1,j+1)*u_max(j+1);
92 -                    end
93 -                    u(s+1,a+1)=r(s+1,a+1)+sum1;
```

```
                    u(s+1,a+1)=r(s+1,a+1)+sum1;
                end
            end
        end
        u_max = max(u');
        [num idx]=max(u');
        opt=ind2sub(size(u'),idx)-1; %Find optimal actions
        for n=1:Max_Stock+1
            A(n,t)=opt(n); %Store optimal actions in a matrix
        end
    end

    %Output results
    F       %Expected Reward given current stock of u units
    r       %Rewards r(s,a)
    P       %Transition Probabilities p(j|s,a)
    u_max   %Optimal rewards
    A       %Optimal actions

```

APPENDIX 2

Matlab Code for Question 4a

```
clear all
clc

N_max = 50; %50 Applicants
applicants = 1:N_max; %Number of applicants as vector

%Proportions
tau = zeros(1,N_max); %Initialise the vector
tau_N(1) = 1;
for N=2:N_max
    for t=1:N-1
        fracs = 1./(t:N-1);
        if sum(fracs) < 1
           cont(t)= 0; %0 => 'Stop'
        else
           cont(t)=1; %1 => 'Continue'
        end
        tau(N) = sum(cont); %Sum up 'Continue' to find number of time steps
                            %before choosing applicant
    end
end
%Calculate tau/N
for N=2:N_max
    tau_N(N) = tau(N)/N;
end

%Plot tau/N against N
subplot(1,2,1)
plot(applicants,tau_N, '-x')
hold on
plot(applicants,1/exp(1)) %Plot e^-1
xlabel('Number of Candidates')
ylabel('Proportion of Candidates to Interview Before Selection')

%Probabilities
u = zeros(1,N_max); %Initialise the vector
u_N(1) = 1;
for N=2:N_max
    fracs = tau(N)./(tau(N):N-1);
    u(N) = sum(fracs);
end
%Calculate u(1)=t/N[1/t + 1/(t+1) + ... + 1/(N-1)]
for N=2:N_max
    u_N(N) = u(N)/N;
end

```

```
46
47      %Plot u against N
48 -    subplot(1,2,2)
49 -    plot(applicants,u_N,'-x')
50 -    hold on
51 -    plot(applicants,1/exp(1)) %Plot e^-1
52 -    xlabel('Number of Candidates')
53 -    ylabel('Probability of Choosing the Best Candidate Using Optimal Rule')
```

REFERENCES *(For Question 4c)*

Hsiau, S., Yang, J. (2000) A Natural Variation of the Standard Secretary Problem. *Statistica Sinica* **10** 639-346. National Changhua University of Education.