MATH4406: Assignment 4

Ryan Heneghan // 41778535

September 25, 2014

Question 1

Consider an insecure parent who wishes to raise a well-balanced child, but more importantly wants to feel good about their parenting abilities. The parent believes that if the child is horribly naughty ($s = -5$) or sickeningly good ($s = 5$), they will alienate other children completely and never have healthy relationships. The parent will either discipline ($a = +1$), or encourage ($a = -1$) bad behaviour to attempt to minimise damage to their well-being. If the child is horribly naughty, the parent will only discipline, similarly if the child is sickeningly good, the parent will only encourage bad behaviour. If the child is more bad than good ($s = -1, \ldots, -4$) or more good than bad ($s = 1, \ldots, 4$), the cost of trying to move their behaviour in the opposite direction becomes increasingly costly ($r = s \times a$) to the parents belief in their parenting skills. Similiarly, if the child is more bad than good (or good than bad), they will enjoy being validated in their behaviour and the parent will feel good about their parenting abilities.

The parent knows that their efforts will not always have the desired effect; if the child is in the 'non-danger' behaviour states $s = -4, \ldots, -4$ the probability that discipline or encouragement will have the desired effect is 0.75. If the child is in the danger states ($s = -5, 5$), the probability that extreme discipline for horribly naughtiness ($s = -5$ and $a = 3$) or extreme encouragement for bad behaviour in a very good child ($s = 5$ and $a = -3$) will have the desired effect is only 0.50. The cost of extreme discipline or encouragement from these states is heavy on the parent's belief that they are a good parent. The parent will attempt to moderate the child's behaviour to maximise their belief that they are a good parent.

Question 2

The process and associated control decisions, coupled with the objective function appears to encourage making decisions that move the system away from state 0 toward states -5 and 5. However, there is a heavy penalty ($r = -15$) for each time-unit spent in these end states. So, the optimal policy will depend on whether the heavy penalty incurred by being in states -5,5 is out-weighed by the increasing rewards gained from moving towards those states.

An optimal policy, denoted π^* depends on what value the discount factor $\lambda \in (0, 1)$ takes. If $\lambda \approx 0$, then the long-term end point penalties in current value becomes quite small very quickly as $t \to \infty$. So, the optimal policy will be one that maximises the reward based on the current state: $\pi^* = (3, -1, -1, -1, -1, -1/1, 1, 1, 1, 1, -3)$. For $\lambda \approx 1$, the long-term end point penalties in current value would become small quite slowly. So the end point penalties would have a high current cost for longer, relative to the possible rewards gained from the other states. The optimal policy in this case would be one that keeps the system in state 0 - as far away from these costly end-points as possible: $\pi^8 = (3, 1, 1, 1, 1, -1/1, -1, -1, -1, -1, -3)$.

Question 3,4,5

The code for the brute force enumeration, value iteration and policy iteration algorithms is contained in Appendix 1. For the value iteration algorithm, an ϵ of 0.01 was sufficiently small to find the optimal policy. This was confirmed in question 7; the value iteration algorithm obtained the same optimal policies for all the 99 different values of λ that the brute force and policy iteration algorithm found, using $\epsilon = 0.01$.

Question 7

As the state space is symmetric, the decision made in state 0 is arbitrary. That is, the reward is the same if the decision maker decides to take action -1 or 1 from state 0. Consequently, state 0 has been omitted in the table below because any optimal policy could choose action -1 or 1 in state 0 and still be optimal. The code to generate and compare the 99 optimal policies for the three methods is contained in Appendix 1.

	States									
λ	-5	-4	-3	-2	-1	1	2	3	4	5
0.01 - 0.63	3	-1	-1	-1	-1	1	1	1	1	-3
0.64 - 0.85	3	1	-1	-1	-1	1	1	1	-1	-3
0.86 - 0.93	3	1	1	-1	-1	1	1	-1	-1	-3
0.94 - 0.96	3	1	1	1	-1	1	-1	-1	-1	-3
0.97 - 0.99	3	1	1	1	1	-1	-1	-1	-1	-3

Table 1: Optimal policies for $\lambda \in (0, 1)$ in steps of 0.01.

These results validate the hypothesis made about the relationship between the optimal decision policy and λ in question 2. As $\lambda \to 1$, the optimal policy becomes increasingly conservative. That is, the best policy becomes more about keeping the process away from the end points and their associated high costs, instead of being about maximising immediate rewards. This reflects the fact that as $\lambda \to 1$ the current-value costs of being in the end-points is larger than the possible rewards gained from moving toward them and maximising current rewards.

Appendix 1

Brute force enumeration algorithm

```
function [d3] = Q3(lambda)
S = [-5:-1,0:5];
reward = zeros(11,512);

% This creates all 512 policies.
n = 9; d = (dec2bin(0:(2^n)-1)=='1');
d_bottom = 3*ones(512,1);
d_top = -3*ones(512,1);
d = [d_bottom d d_top];
d(d==0) = -1;

% This creates an all positive and all negative transition matrix.
d1 = 0.75*ones(1,8); d2 = 0.25*ones(1,8);

Pminus = diag(d1,-1) + diag(d2,1); Pplus = diag(d2,-1) + diag(d1,1);

Pminus_init = [0.75; zeros(8,1)]; Pminus_final = [zeros(8,1); 0.25];
Pplus_init = [0.25; zeros(8,1)]; Pplus_final = [zeros(8,1); 0.75];

P_top = [1/2, 1/2, zeros(1,9)]; P_bottom = [zeros(1,9), 1/2, 1/2];

Pplus = [Pplus_init Pplus Pplus_final]; Pplus = [P_top; Pplus; P_bottom];
Pminus = [Pminus_init Pminus Pminus_final]; Pminus = [P_top; Pminus; P_bottom];

% This for loop creates a specic transition matrix for the current policy.
P_spec = zeros(9,11); P_spec = [P_top; P_spec; P_bottom];

for i = 1:length(d)
   d_curr = d(i,:);
   r_curr = transpose(S.*d_curr);

    for t = 2:10
       if d_curr(t)== 1
          P_spec(t,:) = Pplus(t,:);
       elseif d_curr(t) == -1
          P_spec(t,:) = Pminus(t,:);
       end
    end

    reward(:,i) = (inv(eye(11)-lambda*P_spec))*r_curr;
end

mean_reward = mean(reward,1);
[val idx] = max(mean_reward);
d3 = d(idx,:);
end
```

Value iteration algorithm

```
function [d4] = Q4(lambda)
S = [-5:-1,0:5];
epsilon = 0.01;

newV = zeros(size(S))';
oldV = zeros(size(S))'+(epsilon*(1-lambda)/(2*lambda))+1;
d = zeros(size(S))';
d(1) = 3;
d(end) = -3;

while norm(newV-oldV)>(epsilon*(1-lambda)/(2*lambda))

    oldV = newV;

    %boundary space
    newV(1) = -15+(lambda*0.5*(oldV(1)+oldV(2)));
    newV(end) = -15+(lambda*0.5*(oldV(end)+oldV(end-1)));

    for s = 2:length(S)-1

        %transition probs
        Psminus = zeros(1, length(S));
        Psminus(s-1) = 0.75;
        Psminus(s+1) = 0.25;

        Psplus = zeros(1, length(S));
        Psplus(s-1) = 0.25;
        Psplus(s+1) = 0.75;

        [newV(s), d(s)] = max([-S(s)+lambda*Psminus*oldV, S(s)+lambda*Psplus*oldV]);
    end
end

d(d==1) = -1;
d(d==2) = 1;

d4=transpose(d);
end
```

Policy iteration algorithm

```
function [d5] = Q5(lambda)
S = [-5:-1,0:5];
act = [-1,1];

v_curr = zeros(11,1);

% Initial policy
n = 9;
d = (dec2bin(0:(2^n)-1)=='1');
n0 = ceil(rand*512);

d0 = d(n0,:); d0 = [3 d0 -3]; d0(d0==0) = -1;

dold = [3 zeros(1,9) -3]; dnew = d0;

% This creates an all positive and all negative transition matrix.
d1 = 0.75*ones(1,8); d2 = 0.25*ones(1,8);

Pminus = diag(d1,-1) + diag(d2,1); Pplus = diag(d2,-1) + diag(d1,1);

Pminus_init = [0.75; zeros(8,1)]; Pminus_final = [zeros(8,1); 0.25];

Pplus_init = [0.25; zeros(8,1)]; Pplus_final = [zeros(8,1); 0.75];

P_top = [1/2, 1/2, zeros(1,9)]; P_bottom = [zeros(1,9), 1/2, 1/2];

Pplus = [Pplus_init Pplus Pplus_final]; Pplus = [P_top; Pplus; P_bottom];

Pminus = [Pminus_init Pminus Pminus_final]; Pminus = [P_top; Pminus; P_bottom];

% This for loop creates a specic transition matrix for the current policy.
P_spec = zeros(9,11); P_spec = [P_top; P_spec; P_bottom];

while isequal(dold(1,[1:5,7:11]), dnew(1,[1:5,7:11])) == 0 % While d_n ~= d_n+1
   dold = dnew; % Set prior d_n+1 to current d_n

   c_d = zeros(1,9);
   d_curr = dold;
   r_curr = transpose(S.*d_curr);

    for t = 2:10
       if d_curr(t)== 1
          P_spec(t,:) = Pplus(t,:);
       elseif d_curr(t) == -1
          P_spec(t,:) = Pminus(t,:);
       end
    end
```

```
    v_curr = (inv(eye(11)-lambda*P_spec))*r_curr; % Current policy reward

    for i = 2:10 % Policy improvement step.
        c_state = S(i);

        a_minus = -c_state+lambda*(0.75*v_curr(i-1)+0.25*v_curr(i+1));
        a_pos =  c_state+lambda*(0.25*v_curr(i-1)+0.75*v_curr(i+1));

        [~, idx] = max([a_minus, a_pos]);

        idx(idx==1) = -1;
        idx(idx==2) = 1;
        c_d(i-1) = idx;
    end

    dnew = [3 c_d -3];
end

d5 = dnew;
end
```

Comparison code (Question 7)

```
d_agree = zeros(99,10);
count = 0;

for i = 1:99

    lambda = i*0.01;

    d3 = Q3(lambda);
    d4 = Q4(lambda);
    d5 = Q5(lambda);

    d3 = d3(1,[1:5,7:11]);
    d4 = d4(1,[1:5,7:11]);
    d5 = d5(1,[1:5,7:11]);

    if isequal(d3,d4,d5)
       d_agree(i,:) = d3;
    end
    count = count+1;
end

d_agree
```