Decision Making under Uncertainty
ASEN 6519-007
Maggie Jacoby **Homework 2** February 11, 2020

Conceptual Questions

Problem 1:

Give an example of an MDP with unique non-zero optimal value function, and multiple optimal policies.

Solution

A non-trivial example of this is an undiscounted grid world problem with three cells with the actions, rewards, and transitions described below.

States: This world has three adjacent cells, $\mathcal{S} = \{0, 1, 2\}$.

Actions: At each step you have three available actions: moving left, moving right, or staying where you are. This yields an action space of $\mathcal{A} = \{-1, 0, +1\}$.

Rewards: At each step, if you transition into 0 or 2 from the center cell, then you get a reward of $+10$. If you are in one of the edge cells and don't move, then you get a reward of $+5$. If you transition into the center cell from one of the edge cells, or are in the center cell and don't move, then you get a reward of 0. If you are on an edge, and move farther in that direction (ie you hit the wall), then you don't move and the reward is the same as if you had stayed in the cell ($+5$).

Transitions: Let the transition probabilities be explicitly stated. At each state you have a 50% chance of executing the desired action, and a 25% probability of executing each other action.

Policy While in the center state, a policy of moving left or moving right would be optimally good. While in an edge state, the policy of staying, or moving to the center will be optimally good. However the optimal value at all of these states is the same. We know that the optimal value functions are the same because there can not be distinct optimal value functions for an MDP. This can easily be seen by walking through the situations. If you are in the center cell, your immediate reward is the same if you move left or right ($+10$). If you are in an edge cell, the sum of your rewards over the next two steps will be the same if you stay for both steps ($+5$, $+5$), or if you move through the center to the other edge ($+0$, $+10$). The undiscounted set-up allows this to be easily seen.

Problem 2:

Given: Stationary infinite-horizon MDP with $\mathcal{S} = \{1, 2\}$, $\mathcal{R}(s, a) = s^2$, and $\gamma = 0.9$, and $V^\pi(1) = 37$.

Find: $V^*(2)$, the optimal value at state $s = 2$.

Solution

While we don't know the values of $\mathcal{T}$ explicitly, we know there are only two possible states, and so from either state there are only two possible actions: stay in the current state, or move to the other state. We also know that the sum of the probability of these two actions must be one for both states, as shown below.

$$T(s' = 1|s = 1) + T(s' = 2|s = 1) = 1$$
$$T(s' = 1|s = 2) + T(s' = 2|s = 2) = 1$$

We can let $T(s' = 1|s = 1) = T_1$ and $T(s' = 1|s = 2) = T_2$, thus $T(s' = 2|s = 1) = 1 - T_1$ and $T(s' = 2|s = 2) = 1 - T_2$.

We know that for an infinite horizon problem with discount γ, the optimal value, $V^*(s)$ satisfies the Bellman equation below:

$$V^*(s) = \max_{a \in A} \left[R(s, a) + \gamma \sum_{s'} T(s'|s, a) V^*(s') \right]$$

Given $V^*(s')$, we know that

$$\begin{aligned} V^*(2) &= \max_{a \in A} \left[R(2, a) + \gamma \sum_{s'} T(s'|2, a) V^*(s') \right] \\ &= \max_{a \in A} \left[2^2 + 0.9 * \Big(T_2 * V^*(s') + (1 - T_2) * V^*(s') \Big) \right] \\ &= 4 + 0.9 * V^*(s') \end{aligned}$$

To find the maximum that $V(2)$ can be, we need to look infinitely out. We know that the largest R(s,a) can be is 2^2, and given a discount factor of γ:

$$V^*(2) \le \sum_t^\infty \gamma^t R(s, a) = \sum_t^\infty (0.9)^t * 4$$

$\sum_t^\infty (0.9)^t * 4$ is a geometric sequence, the infinite sum of which is given by $\sum_t^\infty (r)^t * a = \frac{a}{1-r}$. Thus,

$$\boldsymbol{V^*(2) = \frac{4}{1 - 0.9} = 40}$$

Decision Making under Uncertainty
ASEN 6519-007
Maggie Jacoby **Homework 2** February 11, 2020

Exercises

Problem 3: Grid World with a Key

Given:

- Nine-by-one cell grid world
- Can only move left (-1) or right (+1)
- 90% of the time you move the direction you want, 10% of the time the opposite
- Cell 8 is the exit, cell 2 has the key needed to exit
- +10 reward for exiting
- Discount factor $\gamma = 0.95$

Find: The value of being in cell 9 **without** the key.

Solution

The value of being in state 9 without the key is: **4.429**. See the code below for my formulation of the problem, and a sample trajectory. I chose to represent the state space with 19 states. States 1-9 are the cells when you don't have the key. States 10-18 are the same cells, but represent states when you do have the key. State 19 is the terminal state.

CODE OUTPUT: Example of a non-direct path

```
"| | | | | | | | |s|"
"| | | | | | | |s| |"
"| | | | | | |s| | |"
"| | | | | |s| | | |"
"| | | | |s| | | | |"
"| | | |s| | | | | |"
"| | | | |s| | | | |"
"| | | |s| | | | | |"
"| | |s| | | | | | |"
"| |s| | | | | | | |"
"| | |k| | | | | | |"
"| | | |k| | | | | |"
"| | | | |k| | | | |"
"| | | | | |k| | | |"
"| | | | | | |k| | |"
"| | | | | |k| | | |"
"| | | | |k| | | | |"
"| | | | | |k| | | |"
"| | | | | | |k| | |"
"| | | | | | | |k| |"
"| | | | | | | | | |"
Value at state 9: 4.429079667177839
```

```
using POMDPs
using QuickPOMDPs
using POMDPModelTools
using POMDPSimulators
using POMDPPolicies
using Random
using Plots
using LinearAlgebra
using DiscreteValueIteration

S = 1:19 #1:9 without key, 10:18 with key, 19 is the terminal state
A = [-1, 1]
γ = 0.95

function T_m(s, a, sp) ## there are probably moree compact ways to do this...
    if s == 2 # transition from non-key to key state (2 to 11)
        if sp == 12
            return 0.9
        elseif sp == 10
            return 0.1
        else
            return 0.0
        end
    elseif s == 9 #hitting the wall in state 9 (right side)
        if sp == 8
            return 0.9
        elseif sp == 9
            return 0.1
        else
            return 0.0
        end
    elseif s == 10 #hitting the wall in state 10 (left side)
        if sp == 11
            return 0.9
        elseif sp == 10
            return 0.1
        else
            return 0
        end
    elseif s == 17 # terminate
        if sp == 19
            return 1.0
        else
            return 0.0
        end
    else
        if sp == clamp(s + a, 1, 18)
            return 0.9
        elseif sp == clamp(s - a, 1, 18)
            return 0.1
        else
            return 0.0
        end
    end
end

function R_m(s,a)
    if s == 17
        return 10.0
    else
        return 0.0
    end
end
```

```
function fp(s)
    right = union([1,2],collect(10:17))
    left = union(collect(3:9),[18,19])
    if in(s,right)
        return 1
    elseif in(s,left)
        return -1
    end
end

ds = DisplaySimulator(max_steps=20, max_fps=100)
policy = FunctionPolicy(s->fp(s));

#init = Uniform(1:9)
init = Uniform(9)
term = Set(19)
gw = DiscreteExplicitMDP(S,A,T_m,R_m,γ, init, terminals=term);


function POMDPModelTools.render(m::typeof(gw), step) # modified from Zach's code
    str = "|"
    for s in states(m)
        if s <= 9
            if s == step.s
                str *= "s|"
            else
                str *= " |"
            end
        elseif s < 19                          # jump to the key states
            if s == step.s
                str *= "k|"
            else
                str *= " |"
            end
        end
    end
    if step.s <= 9
        return SubString(str,1,19)             # without key
    else
        return SubString(str,19,37)            # with key
    end
end

simulate(ds, gw, policy)

# Solving
solver = ValueIterationSolver(max_iterations=100, belres=1e-6, verbose=false); # creates the solver
curr_policy = solve(solver, gw)

S = 9
v = value(curr_policy, S)
print("Value at state ", S, ": ", v)
```

Problem 4: Continuous state MDP

Given:

- $\mathcal{S} = \mathbb{R}$
- $\mathcal{A} = \mathbb{R}$
- $R(s,a) = -2s^2 - a^2$
- $s' = G(s,a,w) = 2s + a + w$, $w \sim \mathcal{N}(0,1)$
- $\gamma = 1$
- $\pi^*(s) = -k\,s$ where $k \in \mathbb{R}$

Find: k for the optimal policy to a precision of 0.02.

Solution: The best I could find is that $\mathbf{k} = \mathbf{2.30}$. I found this by exploring the potential space for k, and progressively narrowing in on a value for it, as the change in my optimal value function from one iteration to the next decreased. See my code below.

```
using POMDPs
using QuickPOMDPs
using Distributions
using POMDPPolicies
using POMDPSimulators

m = QuickMDP(
    function G(s, a, d=Normal())
        sp = 2*s + a + rand(d)
        r = -2*s^2-a^2
        return (sp=sp, r=r)
    end,
    initialstate_distribution = Normal(),
    actiontype = Float64
);

function eval_k(K)
    function pfunc(s)
        return -K*s
    end
    policy = FunctionPolicy(pfunc)
    sim = RolloutSimulator(max_steps=100)
    r = simulate(sim, m, policy)
    return r
end

function MonteCarloPolicyEval(k)
    u = []
    for i in 1:1000
        curr_r = eval_k(k)
        append!(u,curr_r)
    end
    return mean(u)
end

function FullSearch(k_start, e, mult)
    k = k_start
    u_max = -Inf
    diff = Inf
    best_k = k_start
    while e*mult > 0.000001
        while diff > e*mult
            K_test = k
            println("in: ", k, " : ", u_max)
            k, u = NeighborSearch(mult, k, u_max)
            println("out: ", k, " : ", u)
            if u > u_max
                best_k, u_max = k, u
            end
            println(best_k, " : ", u_max)
            diff = abs(K_test-k)
        end
        mult = mult*.1
    end
    return k
end

function NeighborSearch(m, curr_k, curr_u)
    for j in 1:20
        new_k = curr_k + m*rand(Normal())
        u_x = MonteCarloPolicyEval(new_k)
        if u_x > curr_u
            curr_u = u_x
            curr_k = new_k
        end
    end
    return curr_k, curr_u
end

k_start = 2
eps = 0.1
mult = 1.5
x = FullSearch(k_start, eps, mult)
```

Challenge Problem

Problem 5: Value iteration for ACAS given $\mathcal{T}$ and $\mathcal{R}$

Score submitted to thee leaderboard. See my code below.

```
using DMUStudent
using DMUStudent.HW2
using LinearAlgebra
using Profile

const g = 0.99
const e = 0.01

function ValueIteration(size_s, R, T)
    U = zeros(Float64, size_s);
    error = 100
    first = true
    while error > 0.1
        Q = zeros(Float64, (3,size_s))
        U_k = zeros(Float64, size_s);
        for a in 1:3
            Q[a,:]= R[a] + g*(T[a]*U)
        end
        for s in 1:size_s
            U_k[s] = maximum(Q[:,s])
        end
        error = maximum(abs.(U-U_k));
        println(">  ", error)
        U = U_k
    end
    return U
end

function RunEvaluation(d)
    n = d
    m = UnresponsiveACASMDP(n)
    T = transition_matrices(m, sparse=true)
    R = reward_vectors(m)
    size_s = length(R[1])

    T_arr = [T[i] for i in 1:length(T)]
    R_arr = [R[i] for i in 1:length(R)]
    println("Discretization: ", n)
    v = @time ValueIteration(size_s, R_arr, T_arr);
    return v
end

v7 = @time RunEvaluation(7)
```