Lab - MDPs

This document contains two sections.
Section 01 - some notes related to the main material covered in the lecture on MDPs as a refresher for the lab assignment.
Section 02 - contains three exercises, for which you should submit the answers

Submissions:

- One PDF containing the answers for exercises 01, 02 and 03 (except python code)
- One python file containing the code for exercise 03.

Section 01 - Notes

Markov Decision Processes

A Markov Decision Process is defined by:

- Initial State: S_O
- Transition Model; T (s, a, s′) (Note: We also called this P (s′|s, a) in lecture)
- Reward Function R(s) or R(s, a) or R(s, a, s′)

In this particular formulation we will be using the simplest reward function, R(s).
A policy π specifies which action an agent should take given its current position. In other
words, it provides a mapping from states to actions. The goal of an agent is to learn what policy maximizes its expected sum of discounted reward where the discount factor is specified by γ.
We can express this as:

$$\pi^* = \arg\max_{\pi} E[\sum_{t=0}^{\inf} \gamma^t R(s_t)|\pi] \qquad (1)$$

Utility

The utility of a state is simply the immediate reward associated with the state, plus the future
utility associated with the states that the agent will visit after this state until it stops acting.
A policy π can be specified by choosing for each state s the action that maximizes the expected utility of the following state s′:

$$\pi^*(s) = \arg\max_a \sum_{s'} T(s, a, s')U(s') \tag{2}$$

The utility of a state s can therefore be expressed as function of the immediate reward received s and the future best discounted expected utility of taking an action and transitioning to the set of states it can reach by one action:

$$U^\pi(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s')U(s') \tag{3}$$

Note that γ specifies the degree to which the agent weights distant rewards vs immediate rewards in terms of the utility of a state. If it is set to 0, then the agent only considers the immediate reward associated with the state it is in now in order to compute that state's utility. If it is set to close to 1, then the immediate reward, and the expected utility of the next state are equally important in specifying the current state's utility.
The above equation is called the Bellman equation.

Value iteration

Note that this formulation is essentially identical to the Russel and Norvig formulation: this way you will become both familiar with performing value iteration on Q(s, a) (as formulated in lecture) and this alternative specification.
For each state there is 1 Bellman equation. Therefore for n states there are n unknown utilities. However, since the equations involve taking the maximum, these equations cannot be simultaneously solved. Instead we can solve for the state utilities in an iterative manner:

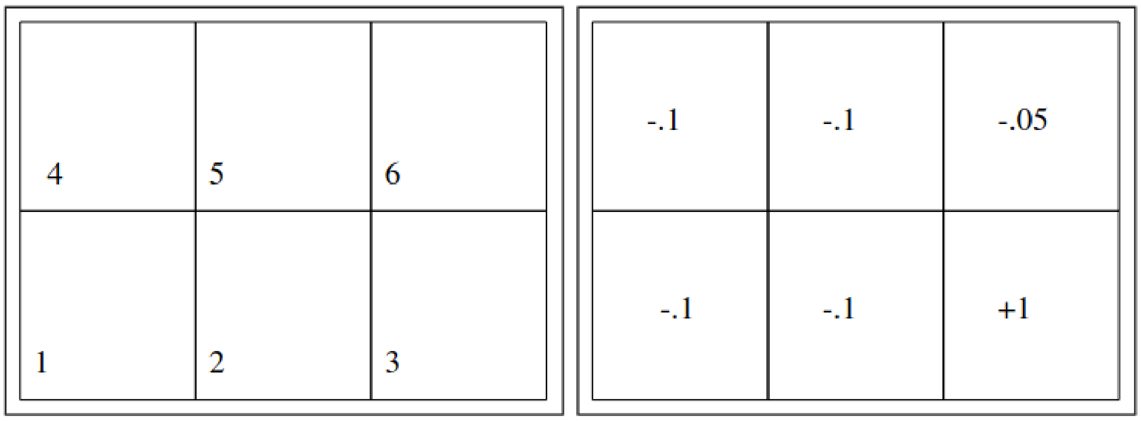
```
1. Initialize the utilities: for $i = 1 : n$, $U(i) = \text{Random-Value}$
2. $\delta = 0$
3. while $\delta > \epsilon \frac{1-\gamma}{\gamma}$
   (a) for each state $s_i$
       i. $U'(s_i i) = R(s_i) + \gamma \max_a \sum_{s'} T(s_i, a, s')U(s')$
       ii. if $|U(s_i) - U(s_i)| > \delta$ then $\delta = |U(s_i) - U(s_i)|$
   (b) $U = U'$, $\delta = 0$
4. Return $U$
```

Epsilon Є is a small value (e.g. 0.1) which you can set as a parameter.

Section 02 - Lab assignment

Pretend an agent is trying to plan how to act in a 3x2 world.
Figure 1 shows the world, the rewards associated with each state, and the dynamics.
There are 5 possible actions: north, east, south, west and stay still. The first 4 actions succeed with probability .9 and go to a right angle of the desired direction with probability .05: see figure 2 for an illustration of this. The fifth action, “do nothing,” succeeds with probability 1.
The rewards associated with each state are
R(1 : 6) = [−0.1 − 0.1 + 1 − 0.1 − 0.1 − 0.05] (4) and are also shown in figure 1.

State 3 is the only terminal state.



(a) States of 3x2 World (b) Rewards of 3x2 World

Figure 1: 3x2 World Specification

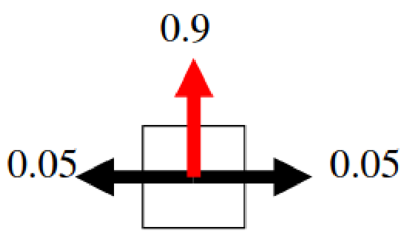


Figure 2: Transition model given a desired movement in red

Exercise 01: Value Iteration

[5 points]
Here we ask you to perform 3 rounds (aka 3 updates) of value iteration. Initialize your utility vector to be 0 for all the states. Please fill in the table with the appropriate values. Use $\gamma = 0.999$ as your discount factor.

State	Expected utility for taking each action					Best Action	Expected Utility for Best Action	Updated Utility
	North	East	South	West	Nothing			
1								
2								
3								
4								
5								
6								
1								
2								
3								
4								
5								
6								
1								
2								
3								
4								
5								
6								

Exercise 02: Best Policy

[2 points]
What is the best policy after 3 iterations? You should be able to read this directly off the table.

Exercise 03: Coding

[3 points]
Write a python function to compute the final results of the state utilities after they have converged, using an epsilon Є of .01.

- Function name: get_state_utilities
- Inputs: epsilon, rewards
- Outputs: utilities at convergence

What is the best policy at the end?
On what iteration does the policy converge?
How many iterations does it take the utilities to converge?