10-703 Deep RL and Controls
Homework 1
Spring 2017

Submitted By:
Karthik Vijayakumar (kvijaya1)
Sameer Bardapurkar (sbardapu)

February 14, 2017

Due February 17, 2017

Problem 1

Consider an environment in which our agent requires caffeine to function[1]. Because caffeine is so important to our agent, we would like the agent to find a policy that will always lead it to the shortest path to coffee. Once the agent reaches coffee, it will stick around and enjoy it.

In order to apply optimal control techniques such as value iteration and policy iteration, we first need to model this scenario as an MDP. Recall that an MDP is defined as tuple (S, A, P, R, γ), where:

S: The (finite) set of all possible states.

A: The (finite) set of all possible actions.

P: The transition function $P : S \times S \times A \to [0, 1]$, which maps (s', s, a) to $P(s'|s, a)$, i.e., the probability of transitioning to state $s' \in S$ when taking action $a \in A$ in state $s \in S$. Note that $\sum_{s' \in S} P(s'|s, a) = 1$ for all $s \in S, a \in A$.

R: The reward function $R : S \times A \times S \to \mathbb{R}$, which maps (s, a, s') to $R(s, a, s')$, i.e., the reward obtained when taking action $a \in A$ in state $s \in S$ and arriving at state $s' \in S$.

γ: The discount factor, which controls how important are rewards in the future. We have $\gamma \in [0, 1)$, where smaller values mean more discounting for future rewards.

[1] If it helps you can think of the agent as a graduate student.

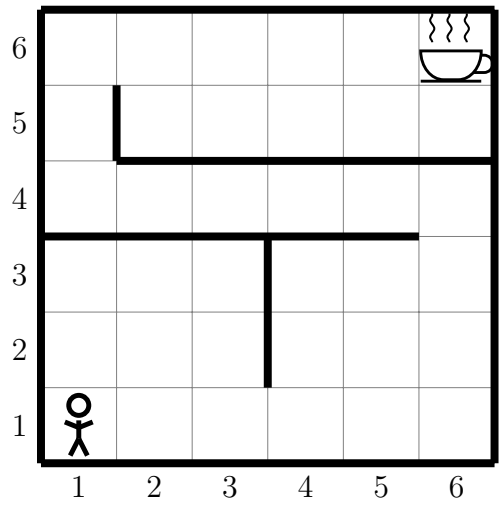


Figure 1: A particular instance of the shortest path problem. The goal is for the agent currently located in state $(1, 1)$ to have a policy that always leads it on the shortest path to the coffee in state $(6, 6)$.

In order to encode this problem as an MDP, we need to define each of the components of the tuple for our particular problem. Note that there may be many different possible encodings.

For the questions, in this section, consider the instance shown in Figure 1. In the figure, the agent is at $(1, 1)$, but it can start at any of the grid cells. The goal, displayed as a coffee cup, is located at $(6, 6)$. The agent is able to move one square up, down, left and right (deterministically). Walls are represented by thick black lines. The agent cannot move through walls. All actions are available in all states. If the agent attempts to move through a wall, it will remain in the same state.

When the agent reaches the coffee cup, the episode ends. Another way to think of this, is that every action in the coffee cup state, keeps the agent in the coffee cup state.

Part a (10pt)

For this section, assume we are modeling the MDP as an infinite horizon MDP. The coffee cup is still an absorbing state.

Using the above problem description answer the following questions:

a) Number of states is 36 (6×6 grid).

b) Number of actions is 4 - Up, Down, Left, Right.

c) Dimension of the Probability matrix $P(s, a, s')$ is $36 \times 4 \times 36$

d) Probabilities of the transition function are indicated in the table below

		s'				
s	**a**	(1,2)	(1,1)	(1,4)	(1,3)	(5, 6)
(1,1)	up	1.0	0.0	0.0	0.0	0.0
(1,1)	down	0.0	1.0	0.0	0.0	0.0
(1,3)	up	0.0	0.0	0.0	1.0	0.0
(6,6)	left	0.0	0.0	0.0	0.0	0.0

e) To obtain an optimal path to the coffee cup, the reward function R and discount γ can be defined as follows

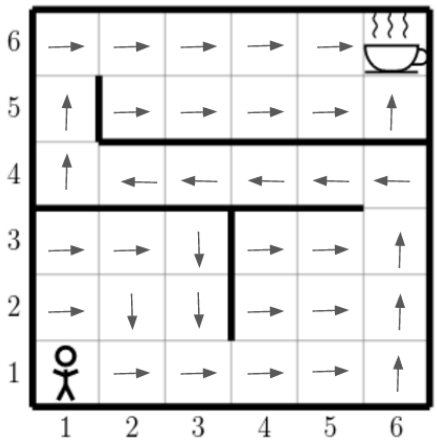
$$R(s) = \begin{cases} 100 & \forall s' \in (6,6), s \notin (6,6) \\ \text{-1} & \forall s' \notin (6,6), s \notin (6,6) \\ 0 & \forall s \in (6,6) \end{cases}$$

$$\gamma = 0.9$$

f) As long as $\gamma \in (0, 1)$, the optimal policy will not be affected. This is because the bellman backup would be a contraction converging to an unique fixed point. While changing γ might change the rate of propagation of rewards across the state-space, the bellman backup ensures that we converge to the same point, and hence we would derive the same optimal policy out of the value function.

g) Since we have 36 states and 4 actions per state, we would have a total of 4^{36} policies.

h) The optimal policy is described below



i) Deterministic policy.

j) Since we are aware of the reward function in our example, we would converge to the exact value function after a few iterations. Hence there would be no advantage of having stochastic policy in our case. A stochastic policy would be helpful in an adversarial environment, where the rewards are not known beforehand.

Part b (2pt)

Now consider that our agent often goes the wrong direction because of how tired it is. Now each action has a 10% chance of going perpendicular to the left of the direction chosen and 10% chance of going perpendicular to the right of the direction chosen. Given this change answer the following questions:

a) Probabilities of the transition function are indicated in the table below

		s'		
s	**a**	(1,3)	(3,2)	(1,4)
(2,2)	up	0.0	0.1	0.0

b) The optimal policy does not change from the previous deterministic case. This is because the optimal actions we previously had would still lead us to the maximum reward with a probability of 0.8. Hence, the agent would still select the same action to achieve maximum return.

c) The values of states are now subject to change. This is because of the stochastic actions, which will lead to a different backed up value to each state. In our case of maximum reward on reaching goal, we would essentially end up with lower values of each state, since the other possibilities from the same action are not highly rewarding.

Part c (4pt)

Now consider a deterministic MDP, like in Part a. But this time, the agent has a meeting with their adviser in 5 minutes. So they need a policy that optimizes getting the coffee in that time limit. We will model this as an episodic MDP.

Consider the case where each step takes 1 minute to execute.

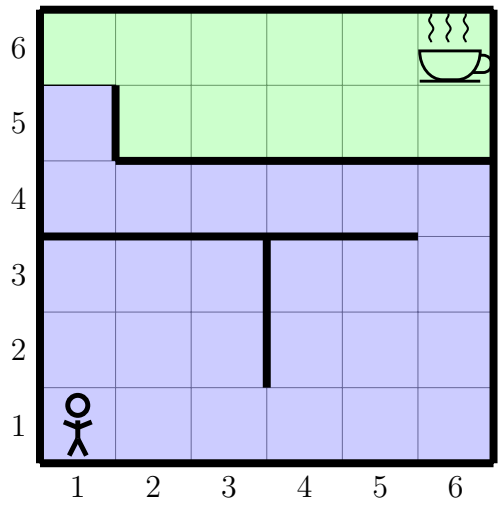


Figure 2: MDP for problem 1c.

a) A similar reward function as in Part a will lead to an optimal path around the coffee cup

$$R(s) = \begin{cases} 100 & \forall s' \in (6,6), s \in s_{green} - \{(6,6)\} \\ \text{-1} & \forall s' \notin (6,6), s \in s_{green} - \{(6,6)\} \\ 0 & \forall s \in (6,6) \end{cases}$$

b) No. The policy will be the same in the green region as in Part a. Both cases are trying to get to goal along the shortest path from any state.

c) We would have $V_{\pi_a} = V_{\pi_b}$. This is because of the fact that the episodes of the MDP will be only 5 timesteps long, none of the episodes will reach the goal. Since all reachable states are empty states which get the same reward, the policies might be same.

d) $V_{\pi_a} = V_{\pi_b}$, since the same policy in s_{green} would lead to the same value, and all policies in s_{blue} would all also lead to a negative value as they can reach only empty states.

Problem 2

Coding

The codes have been implemented in the appropriate templates.

Part a - `Deterministic-4x4-FrozenLake-v0`

a) Policy iteration took 0.00139 seconds. It carried out 7 policy improvement steps and 35 policy evaluations

b) The optimal policy for this map found using policy iteration is shown in Figure 3.

```
DRDL
DLDL
RDDL
LRRL
```

Figure 3: Optimal policy for `Deterministic-4x4-FrozenLake-v0` found using policy iteration

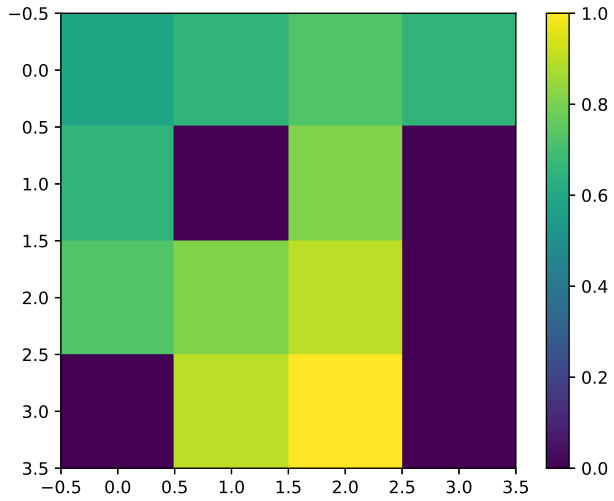


Figure 4: Value function for `Deterministic-4x4-FrozenLake-v0` found using policy iteration

c) The value function for the policy found using policy iteration is shown in Figure 4.

d) Value function found by value iteration took 0.00108 seconds and required 7 iterations.

e) The value function found using value iteration is shown in Figure 5.

f) Value iteration was faster. Both took same number of iterations on a macro level, but policy iterations took multiple value evaluations per iterations and thus took more iterations in total.

g) We did not find any differences in the value functions.

h) The optimal policy for this map found using value iteration is shown in Figure 6.

i) The total cumulative discounted reward obtained by the agent was 0.5905, this value matches the value computed for the starting state.

Part a - `Deterministic-8x8-FrozenLake-v0`

a) Policy iteration took 0.017 seconds. It carried out 15 policy improvement steps and 135 policy evaluations

b) The optimal policy for this map found using policy iteration is shown in Figure **??**.

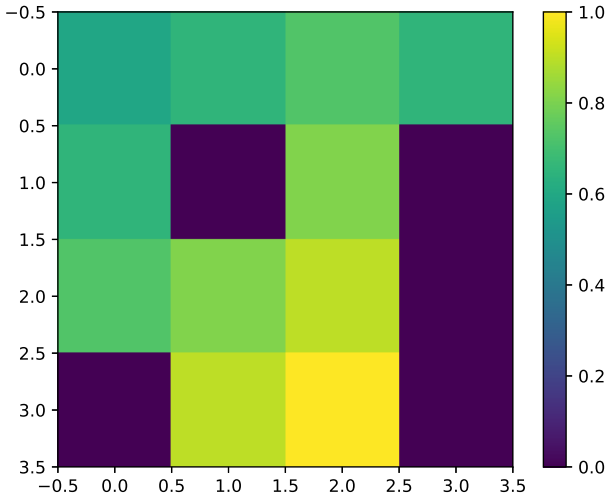


Figure 5: Value function for `Deterministic-4x4-FrozenLake-v0` found using value iteration

```
DRDL
DLDL
RDDL
LRRL
```

Figure 6: Optimal policy for `Deterministic-4x4-FrozenLake-v0` found using value iteration

c) The value function for the policy found using policy iteration is shown in Figure 8.

d) Value function found by value iteration took 0.00965 seconds and required 15 iterations.

e) The value function found using value iteration is shown in Figure 9.

f) Value iteration was faster. Value iteration took more number of iterations on a macro level, but policy iterations took multiple value evaluations per iterations and thus took more iterations in total.

g) We did not find any differences in the value functions.

h) The optimal policy for this map found using value iteration is shown in Figure 10.

i) The total cumulative discounted reward obtained by the agent was 0.2542, this value matches the value computed for the starting state.

Part b - `Stochastic-4x4-FrozenLake-v0`

a) Value function required 0.00447 seconds and 23 iterations.

b) The value function for the policy found using value iteration is shown in Figure 11.

```
DDDDDDDD
DDDRDDDD
DDDLDRDD
RRRRDLDD
RRULDDRD
DLLRRDLD
DLRULDLD
RRULRRRL
```

Figure 7: Optimal policy for `Deterministic-8x8-FrozenLake-v0` found using policy iteration

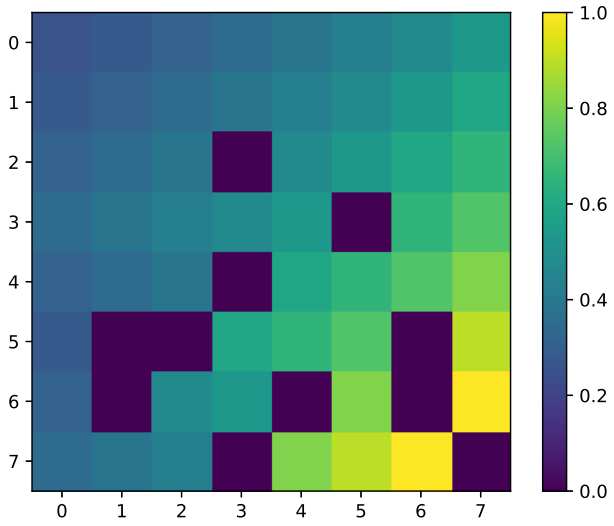


Figure 8: Value function for `Deterministic-8x8-FrozenLake-v0` found using policy iteration

c) The optimal policy for this map found using value iteration is shown in Figure 12.

d) The optimal policy differs from the deterministic case. Here we observe that every action leads to all directions except the direction opposite to it with a probability of 0.33 each. Thus, we can see that the policy is such that near the hole the agent tries to take actions which will minimize its chances to go into the hole. This can be seen in state 6.

e) The total cumulative discounted reward obtained by the agent was 0.0534, this value is quite close to the value computed for the starting state. We think that any deviation is because of the stochasticity of the environment.

Part b - `Stochastic-8x8-FrozenLake-v0`

a) Value function required 0.0196 seconds and 24 iterations.

b) The value function for the policy found using value iteration is shown in Figure 13.

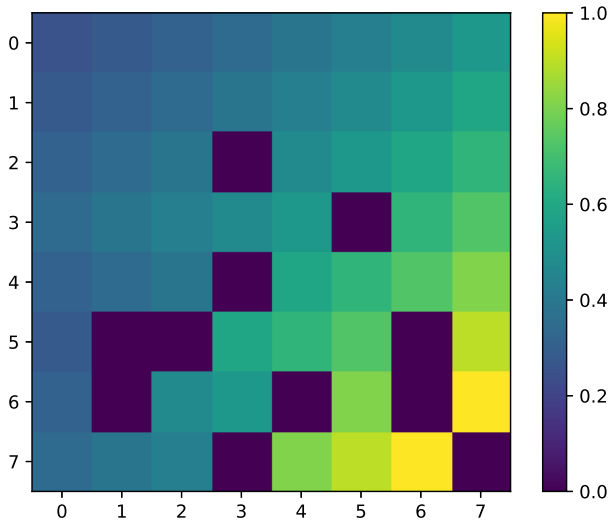


Figure 9: Value function for `Deterministic-8x8-FrozenLake-v0` found using value iteration

```
DDDDDDDD
DDDRDDDD
DDDLDRDD
RRRRDLDD
RRULDDRD
DLLRRDLD
DLRULDLD
RRULRRRL
```

Figure 10: Optimal policy for `Deterministic-8x8-FrozenLake-v0` found using value iteration

c) The optimal policy for this map found using value iteration is shown in Figure 14.

d) The optimal policy differs from the deterministic case. Here we observe that every action leads to all directions except the direction opposite to it with a probability of 0.33 each. Thus, we can see that the policy is such that near the hole the agent tries to take actions which will minimize its chances to go into the hole. This can be seen in state 53.

e) The total cumulative discounted reward obtained by the agent was 0.00487, this value is quite close to the value computed for the starting state. We think that any deviation is because of the stochasticity of the environment.

Part c (4 pt)

We have provided one more version of the frozen lake environment. Now the agent receives a -1 reward for landing on a frozen tile, 0 reward for landing on a hole, and $+1$ for landing on the goal.

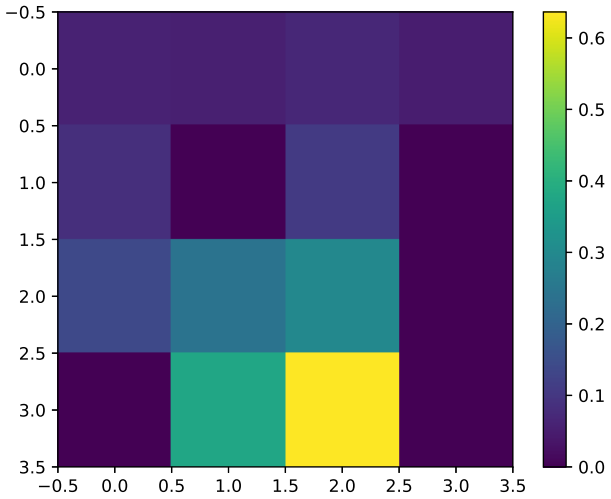


Figure 11: Value function for `Stochastic-4x4-FrozenLake-v0` found using value iteration

```
LULU
LLLL
UDLL
LRDL
```

Figure 12: Optimal policy for `Stochastic-4x4-FrozenLake-v0` found using value iteration

Answer these questions for map `Deterministic-4x4-neg-reward-FrozenLake-v0`.

a) The value function found using value iteration is shown in Figure 15.

b) Yes, the value function is different from the other deterministic map. This is because we have a negative reward for every frozen tile , while terminal hole states give no reward. The values of the grid are as follows

c) The optimal policy for this map found using policy iteration is shown in Figure 16.

d) Yes, the policy is different for the same deterministic map with different rewards. Consider the state at 4 adjacent to the hole - In Part a, the optimal policy was to go down to a frozen tile, since the frozen tile had 0 reward vs the hole which had negative reward. On the contrary, in the map with negative rewards, the optimal policy in the same state is to go right onto a hole since the rewards are switched and hence the agent should prefer to fall into hole if it wants to maximize its returns.

Problem 3 (10pt)

We have implemented the environment in `queue_envs.py`.

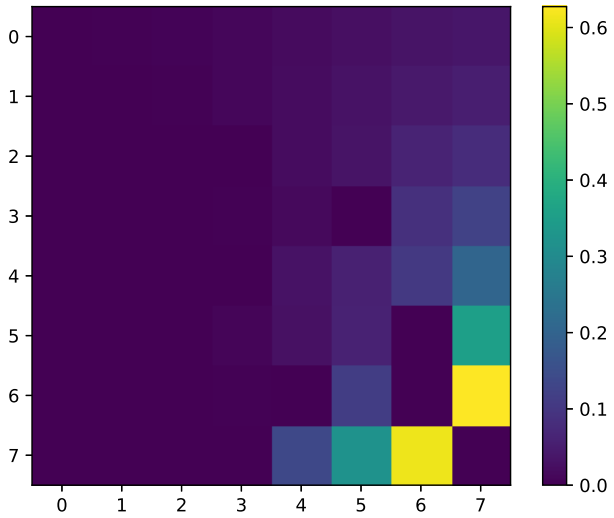


Figure 13: Value function for `Stochastic-8x8-FrozenLake-v0` found using value iteration

```
DRRRRRRR
URRURRRD
URLLRURD
UUUDLLRD
UULLRDUR
LLLDULLR
LLDLLLLR
UDLLDDDL
```

Figure 14: Optimal policy for `Stochastic-8x8-FrozenLake-v0` found using value iteration

Problem 4 (10pt)

In order to answer the questions in this problem, we first need to prove that the Bellman optimality operator ($\mathcal{F}^* : \mathbb{R}^S \to \mathbb{R}^S$) is a contraction in $\mathbb{R}^S$. Consider $V_1, V_2 \in \mathbb{R}^S$ and $\gamma < 1$.

$$\begin{aligned}\left\|\mathcal{F}^*V_1(s) - \mathcal{F}^*V_2(s)\right\|_\infty = \Bigg\| &\max_{a\in A}\left(\sum_{s'\in S} P(s'|s,a)\left(R(s,a,s') + \gamma V_1(s')\right)\right) \\ &- \max_{a\in A}\left(\sum_{s'\in S} P(s'|s,a)\left(R(s,a,s') + \gamma V_2(s')\right)\right)\Bigg\|_\infty \end{aligned} \tag{1}$$

$$\left\|\mathcal{F}^*V_1(s) - \mathcal{F}^*V_2(s)\right\|_\infty \leq \gamma\left\|\max_{a\in A}\sum_{s'\in S} P(s'|s,a)(V_1(s') - V_2(s'))\right\|_\infty$$

$$\left\|\mathcal{F}^*V_1(s) - \mathcal{F}^*V_2(s)\right\|_\infty \leq \gamma \max_{s\in S}\left(V_1(s) - V_2(s)\right)$$

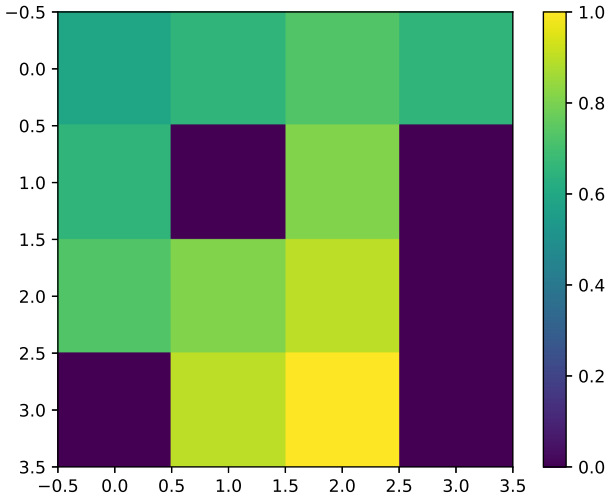


Figure 15: Value function for `Deterministic-4x4-neg-reward-FrozenLake-v0` found using value iteration

```
-1  0 -1  0
 0  0  0  0
 0  0  0  0
 0  0  1  1
```

Thus, we know that the Bellman optimality operator is a contraction in $\mathbb{R}^S$.

a) Since we know that the Bellman operator is a contraction in $\mathbb{R}^S$, we know that V^* is a unique fixed point of $\mathcal{F}^*$ by the Banach fixed point theorem.

b) We know that, if V^* is the optimal value function, then,

$$\mathcal{F}^* V^* = V^*$$

Let V be an arbitrarily initialized value function and let $\gamma < 1$, then, if we apply $\mathcal{F}^*$ to V, we get,

$$\|\mathcal{F}^* V - \mathcal{F}^* V^*\|_\infty \leq \gamma \|V - \mathcal{V}^*\|$$

Also, let V_k be the result of applying $\mathcal{F}^*$ to V, k times, then we can see that,

$$\|V_k - V^*\|_\infty \leq \gamma^k \|V - \mathcal{V}^*\|$$

Thus, if $\gamma < 1$ and as $k \to \infty$, we can see that,

$$\|V_k - V^*\|_\infty \to 0$$

c) The optimal policy can be recovered as follows:

$$\pi^*(s) = \arg\max_{a \in A} \left(\sum_{s' \in S} P(s'|s,a) \left(R(s,a,s') + \gamma V^*(s') \right) \right).$$

```
DDLD
RLLL
DURL
LLRL
```

Figure 16: Optimal policy for `Deterministic-4x4-neg-reward-FrozenLake-v0` found using policy iteration