EL2805 Reinforcement Learning

Lab 1

The Great Escape

Pratima Rao A., Abgeiba Isunza Navarro

1 Problem 1 : The Maze and the Random Minotaur

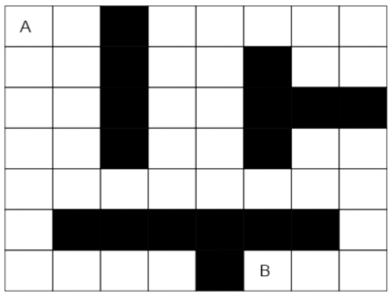


Figure 1: Maze

1.1 Formulating as MDP

State space
The state space is model as all possible positions in the maze without considering the walls.

$$\{S_p = (i, j) : \text{such that i,j is not a wall, } i\epsilon[0,6], j\epsilon[0,7]\} \qquad (1)$$

For the Minotaur we consider the state space as all possible positions in the maze, since the Minotaur does not consider any wall as an obstacle. That is :

$$\{S_m = (i, j) : i\epsilon[0,6], j\epsilon[0,7]\} \qquad (2)$$

Actions
The player has 5 different moves:

$$\mathscr{A}_p = \{\text{up, down, left, right, stay}\} \qquad (3)$$

where in *stay* the player remains in the same state. For the Minotaur, the action space is:

$$\mathscr{A}_m = \{\text{up, down, left, right}\} \qquad (4)$$

Transition probabilities

- Given that the player is at position s_p and takes action a_p that leads to another state $s'_p \epsilon S_p$, then $\mathbb{P}(s'_p \mid s_p, a_p) = 1$.
- Given that the player is at position s_p and takes action a_p that leads to a wall, the player will stay in state s, then $\mathbb{P}(s_p \mid s_p, a_p) = 1$.
- If the Minotaur and the player are at the same state $s_p = s_m$ at the same time the game ends and $\mathbb{P}(s'_p \mid s_p, a_p) = 0$.
- The Minotaur moves according to the Z available moves it has at each position. For example, if the Minotaur is in a corner, there would be Z=2 available movements. Then $\mathbb{P}(s'_m \mid s_m, a_m) = \frac{1}{Z}$.

Rewards The objective of the player is to arrive to the goal point (marked as B in Figure 1) without being caught by the Minotaur.

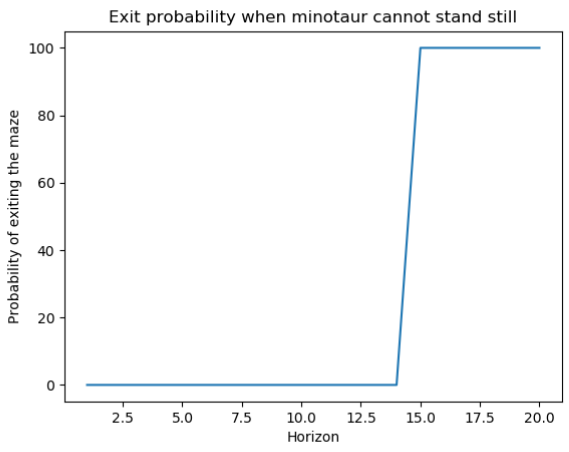
- If at state s_p, action a_p leads the player to a wall, the reward is $r(s_p, a_p) = -\infty$.
- If at state s_p, action a_p leads the player to another state which is not a wall, nor the goal or the Minotaur, the reward is $r(s_p, a_p) = 0$.
- If at state s_p, action a_p leads the player to the same position as the Minotaur, the reward is $r(s_p, a_p) = -20$.
- If at state s_p, action a_p leads the player to the goal, the reward is $r(s_p, a_p) = 20$.

1.2 Solving for different values of horizon

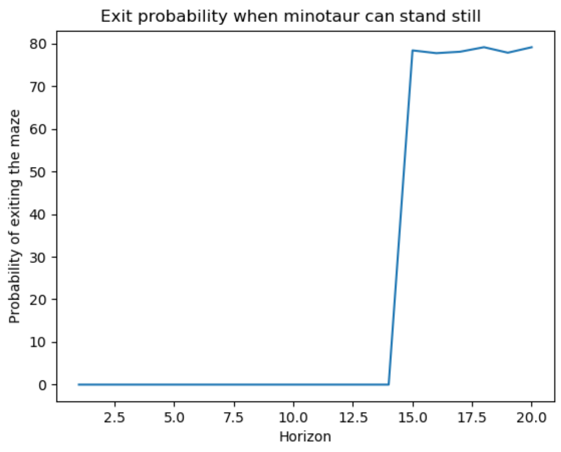
We simulated the MDP problem with a finite horizon T. Where our objective is

$$\mathbb{E}\left[\sum_{t=0}^{T} r(s_t, a_t)\right]$$

We solve this problem by doing a backward recursion over the Bellman equation. In figure 2 we observe the maximal probability of exiting the maze as a function of T when the Minotaur's actions are $\mathbb{A}_{m_1}$ = {up, down, left, right} and $\mathbb{A}_{m_2}$ = {up, down, left, right, stay}. For this test we run 10,000 experiments for each horizon. As seen in both plots, to reach the goal, the player needs at least $T = 15$. However, if the Minotaur is allowed 5 movements $\mathbb{A}_{m_2}$ the probability of the player getting to the goal lowers to around 80% compared to the 100% when the Minotaur has $\mathbb{A}_{m_1}$ for $15 \leq T \leq 21$. Giving an extra movement to the Minotaur changes its transition probabilities and requires more movements from the player to avoid it and to get to the goal. By increasing the horizon the player will increment its probabilities of getting to the goal when the Minotaur has 5 movements.



(a)



(b)

Figure 2: Maximal probability of exiting the maze as a function of T a) when the Minotaur cannot stay in the same position, b) when the Minotaur has the possibility of staying in the same position

1.3 Infinite horizon

In this problem, the life of the player is geometrically distributed with mean 30. In this case we model the MDP as a infinite horizon where $\mathbb{E}[T] = \frac{1}{1-\lambda} = 30$, so the discounted factor is $\lambda = \frac{29}{30}$. We solve the discounted infinite horizon MDP using value iteration. The objective here is to minimize the horizon objective by a stationary policy π with the discount factor λ.

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \lambda^t r(s_t, \pi(s_t))\right]$$

For this case we used $\mathbb{A}_{m_1}$ for the Minotaur and established a precision $\epsilon = 0.01$. We run the game 10000 times after the value iteration converges and calculated the probability of getting out alive by counting the number of successful games. The resulting probability was $0.999 \sim 1$.

2 Problem 3 : Bank Robbing (Reloaded)

Figure 3 shows the city map used for this problem. Position A is the starting point of the robber, position B is the bank and the police moves randomly starting from the corner diagonally opposite to the robber.

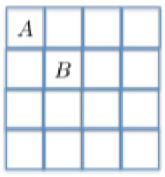


Figure 3: Unknown city

State space
The state space $\mathscr{S}$ comprises of all cell values in the city i.e. from 0 to 15. Each cell value corresponds to a tuple with positional arguments.

$$\mathscr{S} = \{k : \text{where k is from 0 to 15, and each k} = (i, j) \text{ where i,j vary from 0 to 3}\}$$

Actions
The robber has five different moves available at each step.

$$\mathscr{A}_r = \{\text{up, down, left, right, stay}\}$$

The police, however, only has four moves available - up, down, left and right.

$$\mathscr{A}_p = \{\text{up, down, left, right}\}$$

Rewards
The rewards for this are as follows:

- If at state s, an action leads the robber to reach position B, $r(s,a) = 1$
- If at state s, an action leads the robber to reach the same position as the police, $r(s,a) = -10$
- In every other case, $r(s,a) = 0$

2.1 Q-Learning

Q-learning is a learning algorithm which aims to find the best action to take given the current state by taking random actions. In order to do this, we first initialise a Q-table of size $16 \times 16 \times 5$ to 0s. The robber's position is initialised to $(0,0)$, and the police's position to $(3,3)$. Following this, for each step an action is chosen randomly, and the Q-table and current state are updated using the formula:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_a Q\left(s',a\right) - Q(s,a) \right]$$

$$s \leftarrow s'$$

Q-learning was implemented with $\epsilon = 0.25$, $\gamma = 0.8$ and for 1e6 iterations. As can be

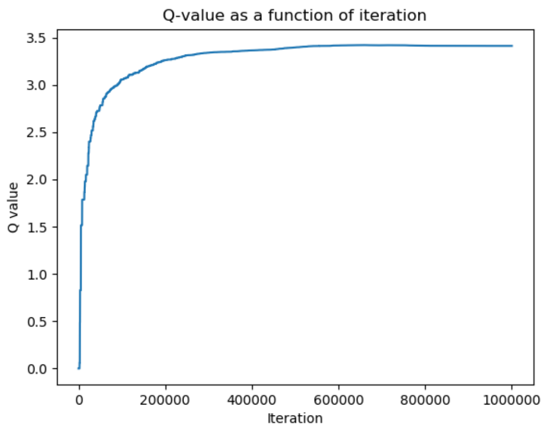


Figure 4: The Q-value for state 0 (with robber and police at their starting positions) as a function of of number of iterations.

seen in figure 4, the Q-value appears to converge around 600,000 iterations.

2.2 SARSA

SARSA (State - action - reward - state - action) is a learning algorithm which follows an approach similar to Q-learning. The initial values are the same as for Q-learning. At each stage, however, instead of choosing an action randomly, an ϵ -greedy exploration is utilised. At each step, a random number r is generated.

- if $r < \epsilon$, an action is chosen randomly
- if $r >= \epsilon$ an action is chosen from the Q values (which yields the maximum utility)

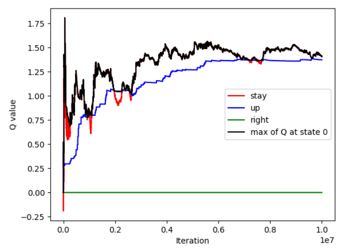
After the action is chosen, the Q-table, action and state are updated using the formula:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma_a Q\left(s', a\right) - Q(s,a) \right]$$

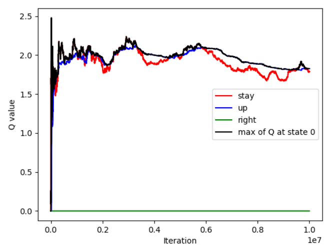
$$s \leftarrow s'$$

$$a \leftarrow a'$$

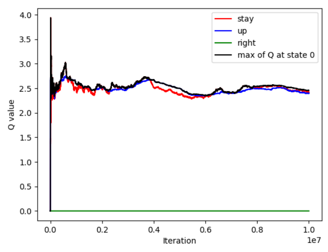
SARSA was implemented using ϵ greedy exploration, with $\epsilon = 0.1, 0.3, 0.5, 0.7$ and 0.9, $\gamma = 0.8$ and for 1e7 iterations in each case.



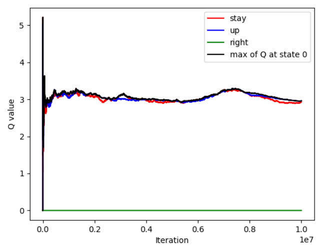
(a) $\epsilon = 0.1$



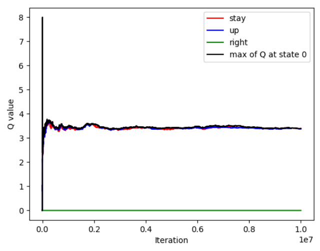
(b) $\epsilon = 0.3$



(c) $\epsilon = 0.5$



(d) $\epsilon = 0.7$



(e) $\epsilon = 0.9$

Figure 5: SARSA with different values of ϵ. The Q-value for state 0 (with robber and police at their starting positions) with different starting moves is plotted.