EL282805 - Reinforcement Learning - Lab 1 - Report

Adrian Chmielewski-Anders, Leo Zeitler
9512115537, 9509289634
`{amca3, llze}@kth.se`

1 The Maze and the Random Minotaur

1.1 Modelling the MDP

The state space is encoded as two tuples, the (x, y) coordinates of the player a and the minotaur m with the origin being in the top left. The player can be only in spaces with no walls, and the minotaur can be in any state (i.e. any (x, y) coordinate). Of course both must be within the bounds of the grid, so in this case $0 \leq x < 8$ and $0 \leq y < 7$. Formally we could write this as

$$\mathcal{S} = \{(p, m) \mid p = (x_1, y_1), m = (x_2, y_2), x_i < 8, y_i < 7 \forall i, j \notin W\} \tag{1}$$

where W is the set of coordinates of all the walls.

The available actions for the player in particular state s is a subset $\mathcal{A}_s$ of $\mathcal{A}$ that is defined as

$$\mathcal{A} = \{\text{up}, \text{down}, \text{left}, \text{right}, \text{stay}\} \tag{2}$$

which are the valid actions of the player being in state s. If there is a wall to the right, for example, then right is not a valid action. Thus, the actions depend on the present state. Formally, an action is valid if applying it results in a state which is a member of $\mathcal{S}$.

The reward is 1 upon entering the goal state, 0 everywhere else, except in the case that the player is in the same position as the minotaur at the same time step, in which case it is -1. In the event the player enters the goal state as soon as the minotaur does, for eas of programming, we adopt the convention that this is a win, and so the reward is 1. Formally, this can be written

$$r_t(s_{t-1}, a) = \begin{cases} -1 & p_t = m_t \\ 1 & p_t = g \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where s_t is the result of applying a to s_{t-1} and contains both the positions of the player p_t and minotaur m_t. The position of the goal is g.

The state transition probabilities are certain in p but random in m since the player moves with certainty, but the minotaur moves at random. Formally,

$$p((p_{t+1}, m_{t+1}) \mid p_t, m_t, a) = \frac{1}{A} \tag{4}$$

provided $\|m_{t+1} - m_t\|_M = 1$ where $\| \cdot \|_M$ denotes Manhattan distance, and a is the action applied to legally move from p_t to p_{t+1}, such that $\|p_{t+1} - p_t\|_M = 1$. Also, A is the number of legal moves that the minotaur can make which satisfies $2 \leq A \leq 4$ and dependent upon only its state. In one case in part (b) the upper bound is 5. The minotaur, however may move within walls, but not outside the boundaries of the map. Also the minotaur cannot stay in the main version of the exercise but can in one part of part (b).

1.2 Finite Time Horizon

To solve a model with a finite time horizon, it is necessary to set the objective to an undiscounted sum over the expected rewards. To be precise, let $R(.)$ bet the function that returns a reward for a given sequence of state-action pairs under a particular policy π $((s_1, a_1), ..., (s_T^\pi, a_T^\pi))$. The objective can be mathematically expressed as

$$\mathbb{E}[R(((s_1, a_1), ..., (s_T^\pi, a_T^\pi)))] = \mathbb{E}[\sum_i^T r_i(s_i^\pi, a_i^\pi)] \tag{5}$$

This can be solved using the value function $V_t(s)$ that defines the expected reward for a given state at a particular timestep t, where s denotes a state, and the Bellman's equation which is defined as the maximum of the value function over the possible actions a

$$u_t^B(s_t) = \max_{a \in A(s_t)} \left[r(s_t, a) + \sum_{j \in S} p(j|s_t, a) u_{t+1}(j) \right] \tag{6}$$

We discovered that the optimal policy, if the minotaur is not allowed to stand still, does not have strong incentives to go to the goal as quickly as possible, but rather just get there in general. An example of the moves taken can be seen in figure 1.

Whenever the time horizon was set to $T \geq 16$, the chased agent A won with probability $p = 1.0$. However, this was different if the minotaur could take "standing still" as a possible move. Then it became harder for the agent to win and to exit the maze in the given time.

We assume that the difference between when the minotaur can and cannot stand still occurs due to the dissimilar outcome of the situation if the minotaur and the agent are right next to each other. While the agent could safely move to the position of the minotaur when it was not allowed to stand still, we need to move actively around it if it has the possibility of staying available in every state. That makes it more difficult to get to the goal state in time. When the agent had more time, the probability that it would win grew larger and larger. This can be seen in Figure 2 where we plot the empirical probability of winning for various values of T averaged over 1000 trials. This is complimented by the plot in Figure 3 where we plot the value function at the initial position for various values of T. We note that the value function evaluated at the starting position for various learned values of T resembles the empirical results in Figure 2.

1.3 Infinite Time Horizon

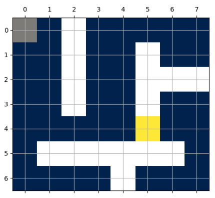
In contrast to the previous problem we assume that the time horizon T is now randomly taken from a geometric distribution. We changed our model such that we apply an objective with an infinite time horizon. To discount the influence of potential future rewards we set a discount factor which is determined by the mean of a geometric distribution. It was given by the task that the duration of the agent's life is geometrically distributed with mean $\mathbb{E}[T] = \mu = 30$. The mean of a geometric distribution and the discount factor of a geometric series λ can be set into relation via

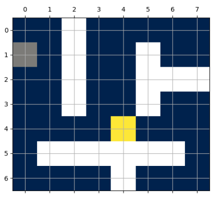
$$\begin{aligned} \mathbb{E}[T] = \mu &= \frac{1}{1-\lambda} \\ \iff \lambda &= \frac{1-\mu}{\mu} \\ \implies \lambda &= \frac{29}{30} \end{aligned} \tag{7}$$

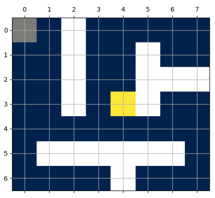
We can now motivate our modelling choice through the following equation:

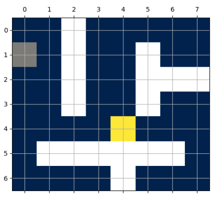
$$\mathbb{E}[\sum_t^T r(s_t, a_t)] = \mathbb{E}[\sum_t^\infty \lambda^t r(s_t, a_t)] \tag{8}$$

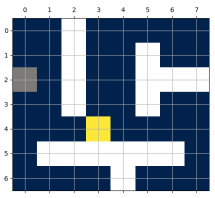
where T is sampled from a geometric distribution and λ derived via equation 7. However, this makes it difficult to evaluate the value function and to apply Bellman's equation implemented as a

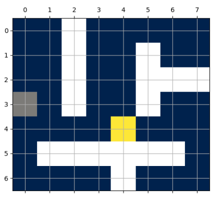

(a) With t=1

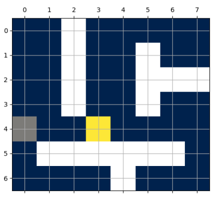

(b) With t=2

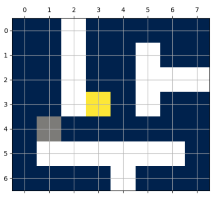

(c) With t=3

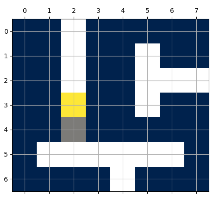

(d) With t=4

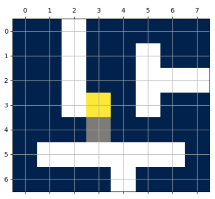

(e) With t=5

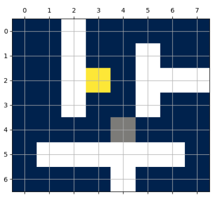

(f) With t=6

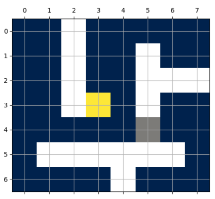

(g) With t=7

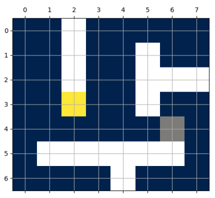

(h) With t=8

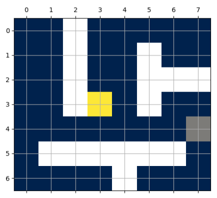

(i) With t=9

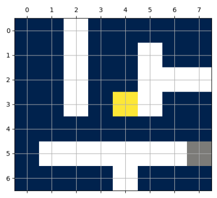

(j) With t=10

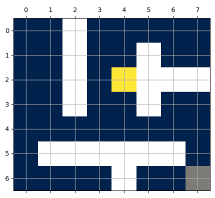

(k) With t=11

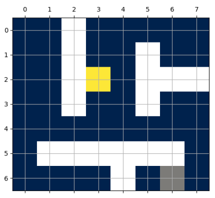

(l) With t=12


(m) With t=13


(n) With t=14, a winning state.


(o) With t=15, a winning state.


(p) With t=16, a winning state.


(q) With t=17, a winning state.

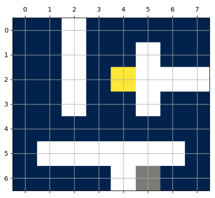

(r) With t=18, a winning state.

Figure 1: The path of A through the maze when the minotaur is not allowed to stay still and a finite time horizon objective with $T = 20$.

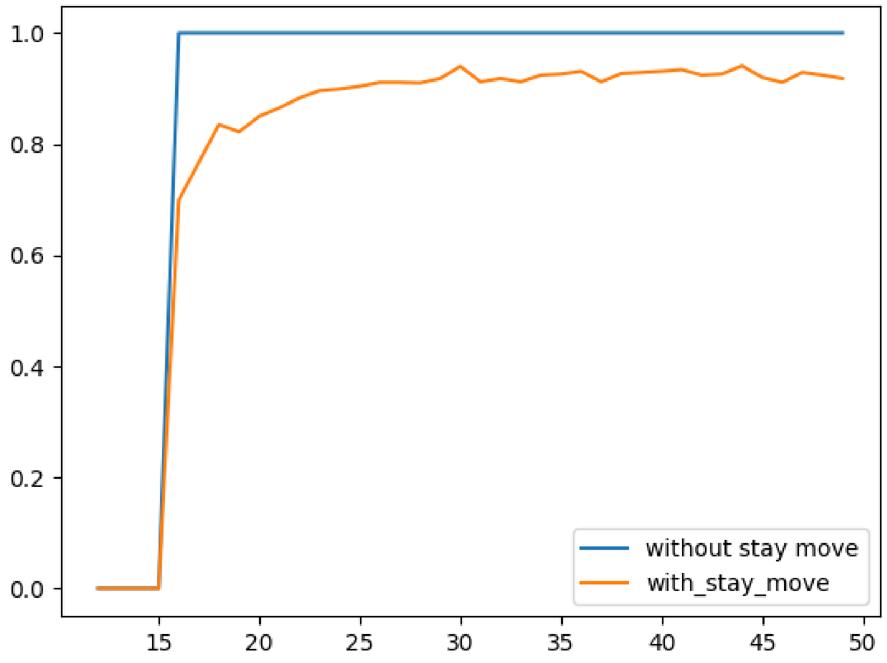


Figure 2: The empirical probability as a function of the time horizon T where we count the number of wins over $N = 1000$ trials.

Dynamic Programming (DP) approach. Instead, we make use of an iterative approach, called value iteration (VI), where we improve the value function until the update steps become smaller than a given threshold. All other modelling assumptions stay the same. Running the optimal policy with a geometrically distributed time horizon, the probability of exiting the maze alive is $p = 0.623400$, what is inline with our expectations. One would expect that we would not have enough time when $P(T < 15)$ which, since $T \in \text{Geom}(\mu)$ where $\mu = 30$, is just an evaluation of the CDF of a Geometric distribution, namely $p \approx 1 - F(15) = 1 - 1 + (1 - \frac{1}{30})^{15}$.

2 Robbing Banks

2.1 Formatting as an MDP

The state space is very similar to the first problem. It can be modeled as a length 4 tuple which contains the positions of the player and the police and they must be within the bounds of the grid world, formally

$$\mathcal{S} = \{(j,k) \mid j = (x_1, y_1), k = (x_2, y_2), x_i < 6, y_i < 3 \forall i, j\} \tag{9}$$

The available actions for the player in a particular state s is a subset $\mathcal{A}_s$ of $\mathcal{A}$, where

$$\mathcal{A} = \{\text{up}, \text{down}, \text{left}, \text{right}, \text{stay}\} \tag{10}$$

The subset $\mathcal{A}_s$ is full of moves a such that applying a to s leads to $s' \in \mathcal{S}$.

$$R_t(s_{t-1}, a) = \begin{cases} -50 & p_t = m_t \\ 10 & p_t = g \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

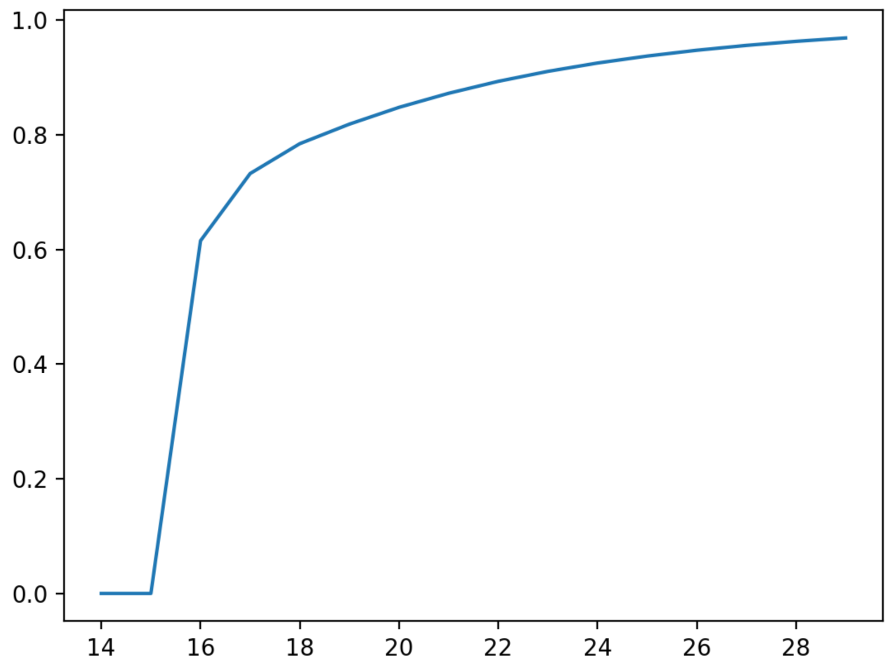


Figure 3: The value function for various values of T in the case where the minotaur is able to stand still.

where s_t is the result of applying a to s_{t-1} and contains both the positions of the the player p and the police m. Here, g is the position of any of the goals.

The state transition probabilities are only stochastic in the movements of the police. The notation is the same as in (4) except which moves are valid (i.e. considered in A) are only those which move "in the direction of" the player p as specified on the homework. Furthermore, if the player and police are in the same position, then there is only one state which is transitioned to with probability 1. This state is identical to the initial state.

2.2 Solving and Plotting

As required in task (b), we solved the MDP for different values of the discount factor λ. As it can be seen in the plot in figure 4, the values of the value function for the initial state start with an expected return of around 10 with a discount factor $\lambda = 0.1$ but grows exponentially with increasing λ and has its peak at $\lambda = 0.9$. This is expected since we are taking more future moves into account and the discount factor is incorporated exponentially into the infinite time horizon objective.

3 Bank Robbing (Reloaded)

When implementing exercise 3, the next state transition function incorporates the next state of the police, which moves at random. For both Q-Learning and SARSA, all trials were conducted over 10M iterations in one "episode." That is, there is no exit move. The value of $\lambda = 0.8$ is maintained throughout as specified in the description of the problem. This is an expected result because with increasing λ the values in the future are more important.

(a) In Figure 5 the value of the Q-function is plotted over time. This is what one would expect since the values of up and left are invalid moves, thus since the initial values of the Q

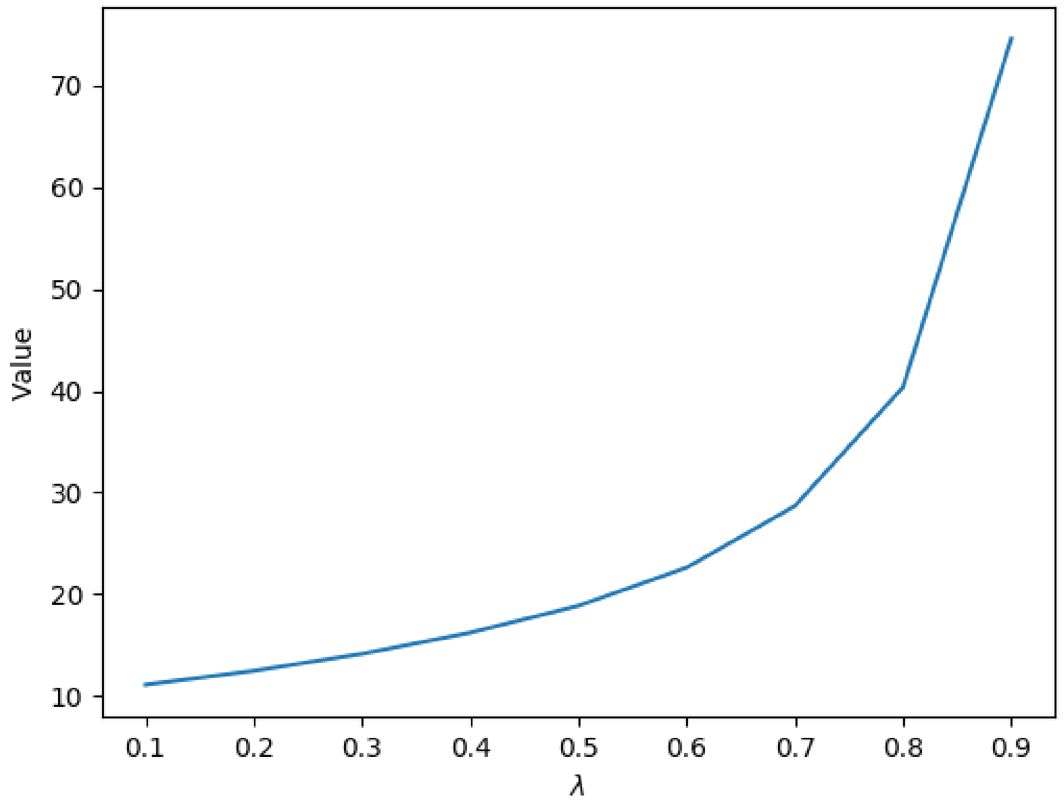


Figure 4: Dependence of the optimal value of the initial state on the discount factor λ

function are zeros, they should stay zero. The largest values are right and down, which are the two steps needed to get to the bank. Stay is the only other valid move which is somewhere in the middle. The learning is quite quick and does not change much after 100K steps. There is no ϵ-greedy policy used for exploring but rather each move is selected at random (i.e. uniformly, each valid move with equal probability).

(b) Figure 6 shows the same plots as in Figure 5 but for the SARSA algorithm. One can see that this results in much more noisy values of the Q-function. This is because it is an on policy method and we are using the Q-function actively during training. Furthermore, note that the case where $\epsilon = 0.1$ it takes up until around 6M steps for the proper value for the down action to be learned which is because the Q-function is used actively. Furthermore, the values of the Q-function are more noisy in general since the policy learns not the best Q but the "ϵ-optimal" Q. This behavior can be seen throughout increasing the value for ϵ. The interesting case of the harmonically decreasing ϵ since the values of Q are still quite small, though the ordering is correct. One can hypothesize that the values of ϵ get too small too quick, and then there is not much exploration which is done, leading to not much further change of the Q-function.

It is worth noting that for all four strategies tried, all learn the general policy of going to the bank, and staying there until the police are one move away, then running away, and then repeating. That is, to get back to the bank without getting caught and repeat.

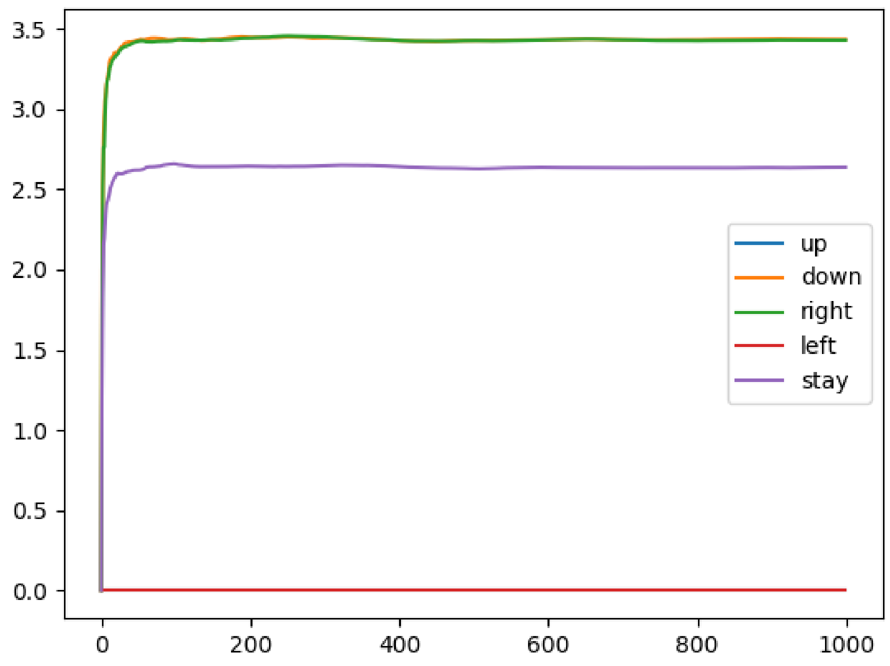
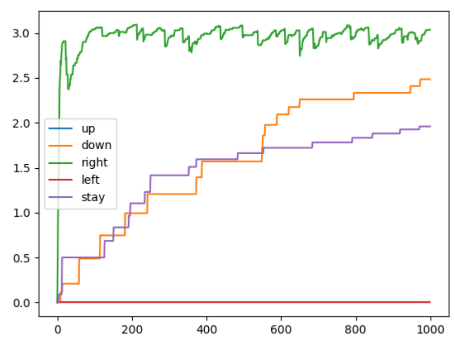
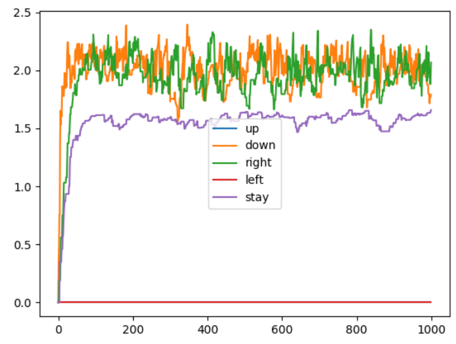


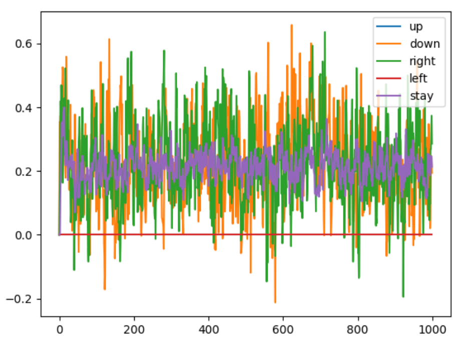
Figure 5: The value of Q-function over time (i.e. x-axis is time, and y-axis is the Q-function value) for the initial state. The is recorded over 10M steps though the values of the Q-function are recorded every 10K steps.



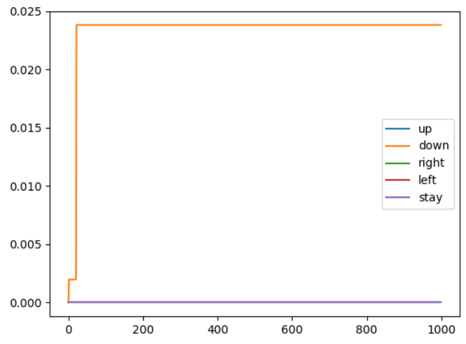
(a) With $\epsilon = 0.1$



(b) With $\epsilon = 0.3$



(c) With $\epsilon = 0.7$



(d) With $\epsilon = \frac{1}{n(s,a)}$

Figure 6: The same plots of the value of the Q-function for the initial starting point as in Figure 5 only with SARSA. The training was done over 10M steps and values recorded every 10K steps. The value of ϵ was varied within 0.1, 0.3, 0.7 as well as an adaptive ϵ equal to the reciprocal of the number of updates to this state action pair.