Reinforcement Learning Tutorial 7, Week 8
— with solutions —
Reward expectation, Prioritisation, and Uncertainty*

Michael Herrmann

March 2024

Problem 1 – R-learning

R-learning[1] is similar to $\mathcal{Q}$-learning, in particular for non-discounted, non-episodic problems. It is based on the average reward $\rho = \lim_{n\to\infty} \frac{1}{n}\sum_{t=1}^{n} E\left[r_t\right]$, and considers the current rewards in comparison to this accumulating reward average towards the value:

$$V\left(s_t\right) = \sum_{k=1}^{\infty} E\left[r_{t+k} - \rho | s_t = s\right]$$

$$\mathcal{Q}\left(s_t, a_t\right) = \sum_{k=1}^{\infty} E\left[r_{t+k} - \rho | s_t = s, a_t = a\right]$$

In this relative value function (relative to the average), ρ is slowly adapted as a measure of success. In this way a different concept of optimality is implied in particular for non-episodic tasks. As an algorithm, R-learning works as follows

1. Initialise ρ and $\mathcal{Q}\left(s, a\right)$
2. Observe s_t and choose a_t (e.g. ε-greedy), execute a_t
3. Observe r_{t+1} and s_{t+1}
4. Update
$$\mathcal{Q}_{t+1}(s_t, a_t) = (1-\eta)\,\mathcal{Q}_t(s_t, a_t) + \eta\Big(r_{t+1} - \rho_t + \max_a \mathcal{Q}_t(s_{t+1}, a)\Big)$$
5. If $\mathcal{Q}\left(s_t, a_t\right) = \max_a \mathcal{Q}\left(s_t, a\right)$ then
$$\rho_{t+1} = (1-\alpha)\,\rho_t + \alpha\Big(r_{t+1} + \max_a \mathcal{Q}_t(s_{t+1}, a) - \max_a \mathcal{Q}_{t+1}(s_t, a)\Big)$$

*with special thanks to Adam Jelley

[1] A. Schwartz (1993) A reinforcement learning method for maximizing undiscounted rewards. 10th ICML. (You don't need to know R-learning for the exam.)

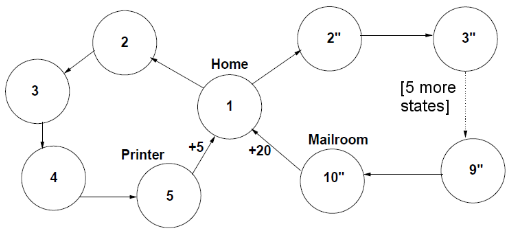


Figure 1: An illustrative example for R-learning (Schwartz, ibid.).

[*We would choose* $\eta \gg \alpha$, *because, otherwise, for* $r = 0$, $\mathcal{Q}$*-value may cease to change and the agent may get trapped in a suboptimal limit cycle.*]

Compare R-learning and $\mathcal{Q}$-learning in the following simple example, where only one decision needs to be taken: The agent moves either to nearby printer ("o.k.") or to distant mail room ("good").

Is it possible that R learning finds the optimal solution quicker than $\mathcal{Q}$-learning? How does the result for $\mathcal{Q}$-learning depend on the discount factor γ?

Answer The problem is quite similar to a two-armed bandit, but waiting times differ for the "arms": Q-learning with low γ favours the nearby goal, while its learning times get longer for larger γ, see Fig. 2. R-learning identifies the better choice quickly based on trajectory based reward averages. Note that results may depend on parameters.

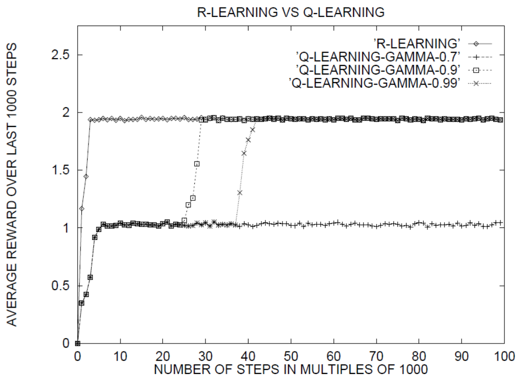


Figure 2: R-learning (Schwartz, ibid.) can solve certain problems much more quickly than for example Q-learning. Also note that Q-learning does prefer the shorter branch for small γ, while it has longer learning times for γ close to 1 (Schwartz, ibid.).

Problem 2 – Prioritised sweeping[2]

While Dyna[3] agents select state-action pairs uniformly at random from the previously experienced pairs, it might be more efficient to use a non-uniform probability distribution. Why? Which state-action pairs should be preferred? Discuss the role of goal states in this context.

Answer

Once having bee successful in a maze task (see Fig. 3), during the next episode, only state–action pairs that are are part of the trajectory towards the goal has a positive value, while all other values are still zero, so there is no need to update the corresponding states. In a more general sense, however, the idea is that "relevance" is not only about goals but about all changes (i.e. new information) in the reward/value structure.

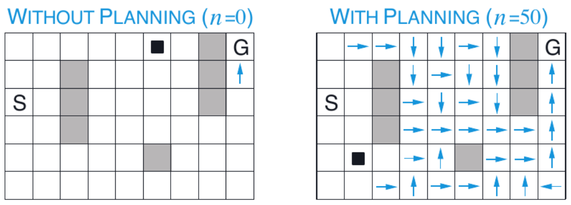


Figure 3: A simple maze task (Sutton and Barto, 2020, Fig. 8.3)

Although it may seem that goal states are critical, we should be reminded that not all problems have goals states. In fact, any state where information enters is interesting, i.e. rather than work backwards from goal states only, we should consider any state where a change of value has occurred.

This is an iterative procedure which can sweep through the recent trajectories, where different streams of information may simply be added up.

If not real trajectories are used, but a planner produces the respective preceding, then the backwards frontier can grow exponentially, so that other criteria can be used to limit the computational effort, such as the size of the updater.

Also Example 8.4 in Sutton & Barto (2020) shows, that prioritised sweeping does not avoid the exponential complexity of the search problem in a maze, but can achieve for example an order of magnitude of improvement.

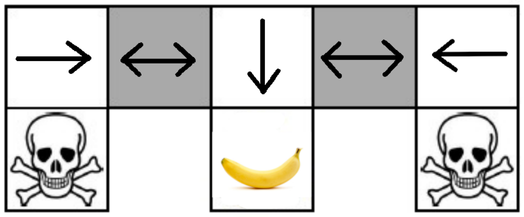
[*Is there any downside to this?*]

[2]Sutton & Barto, Sect. 8.4

[3]The Dyna-Q algorithm is one example of Dyna, see Lecture 7, Slides 7ff

Problem 3 – Ambiguous state information

Discuss the aliased gridworld example[4], where the agent, which is here, as always, a monkey, cannot distinguish between the two densely overgrown swampy parts of the environment (shown here as grey grid cells). This means that for the two states that are shown in grey the same entry of the policy $\pi(\cdot, \text{grey})$ has to be used. Actions are: N, W, S, E. Rewards and states as shown in the figure, where we naturally assume that $r(\textit{skull}) \ll r(\textit{banana})$.



Compare the optimal deterministic policy for the example with the optimal stochastic policy. How could an algorithm find the stochastic policy?

Answer

A deterministic solution needs to decide which unique action to assign to both of the two grey states. For example, if both are given the "left" action, then at least in half of the cases the banana can be found, while in the other cases the monkey is either stuck or worse, i.e. the deterministic scheme gets stuck in a portion of all cases. Can we do any better in this problem?

Although this was not intended, the original figure could be construed as suggesting that the monkey is starting always in the left upper cell. Then a solution could be easy unless the nature of the grey state is that of tunneling the monkey to the other grey field in some cases.

Also, a non-Markovian solution is imaginable: Keep going until a backwards arrow is encountered or until a southwards arrow indicates that the goal is almost reached.

The optimal solution (in terms of discounted reward averaged over many episodes) is to use a probability of 0.5 of each of left and right in the two grey fields and actions towards the inside in the fields next to the skulls, and obviously a south pointer in the center, see the figure. So the trajectory can be long but has finite average as the following analysis shows.

Assuming that leftwards and rightwards probabilities are not necessarily the same, and the monkey starts in one of the grey states, then the return is

$$R \quad = \quad r(banana)\,\gamma^2 \left(P(right) \sum_{k=0}^{\infty} \gamma^{2k} P(left)^k + P(left) \sum_{k=0}^{\infty} \gamma^{2k} P(right)^k \right)$$

[4]adapted from David Silver's lecture 7

$$\begin{aligned} &= r(banana)\,\gamma^2 \left(P(right)\frac{1}{1-\gamma^2 P(left)} + P(left)\frac{1}{1-\gamma^2 P(right)} \right) \\ &= r(banana)\,\gamma^2 \left(\frac{1-P(left)}{1-\gamma^2 P(left)} + \frac{P(left)}{1-\gamma^2(1-P(left))} \right) \end{aligned}$$

which is maximal for $P(right) = P(left) = 0.5$, i.e. introducing any asymmetry into the otherwise symmetric problem causes longer trajectories towards one side which are not compensated by gains at the other side. To prove, we can take the derivative of the last expression w.r.t. $P(left)$, set it to zero, and solve for $P(left)$. If $\gamma = 1$, then it does not matter how long it takes on either side, so any action probabilities are fine as long as $P(right) > 0$ and $P(left) > 0$.

Finally, we could also consider here as stochastic environment, where any action can be wrong by 90deg with some probability, so that an upward action would be best for the corner states, and upward or downward action of the grey states and a downwardly directed action for the middle state would be best, but if errors are rare, than trajectories can become very long, so that more control over the policy may be preferable.

This is a very interesting example, as it shows that stochastic policies can indeed be better than deterministic ones although this is implied only if state information is missing.