Reinforcement Learning Tutorial 7, Week 8

—

Reward expectation, Prioritisation, and Hyperparameters*

Pavlos Andreadis, Michael Herrmann

March 2025

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

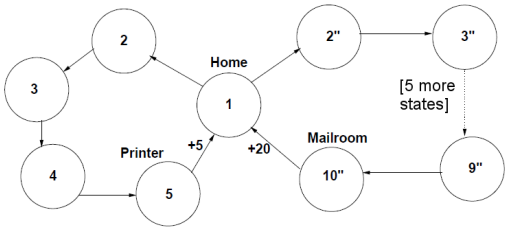
1. Initialise ρ and $\mathcal{Q}\left(s, a\right)$
2. Observe s_t and choose a_t (e.g. ε-greedy), execute a_t
3. Observe r_{t+1} and s_{t+1}
4. Update
$$\mathcal{Q}_{t+1}(s_t, a_t) = (1 - \eta)\, \mathcal{Q}_t(s_t, a_t) + \eta\Big(r_{t+1} - \rho_t + \max_a \mathcal{Q}_t(s_{t+1}, a)\Big)$$
5. If $\mathcal{Q}\left(s_t,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

[*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?

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.

Problem 3 - Discussion

Part a

Considering a Reinforcement Learning algorithm in general, what is the overall effect of increasing the learning rate? What happens when you set it too high? What happens when you set it too low?

Part b

Is the discount factor γ:

1. Part of the definition of a Markov Decision Process? That is, a part of the definition of the problem to be solved; or
2. Is it external to the problem? That is, a hyperparameter for training the model.

When a discount factor is close to 1, we end up with a long horizon problem. That is, we plan for long-term gains. Assume we were training a Reinforcement

[2] Sutton & Barto, Sect. 8.4
[3] The Dyna-Q algorithm is one example of Dyna, see Lecture 7, Slides 7ff

Learning agent for a long-horizon problem. Could you think of a reason for which a method using short-horizon targets (cut-off at some horizon h) might outperform a method using long-term horizon targets on this problem? To aid in answering, consider searching online for "planning horizon reinforcement learning model accuracy" and look for related work.

Part c

What other parameters are relevant in Reinforcement Learning, and what considerations would help to determine their values?