Reinforcement Learning Tutorial 7, Week 8
— with solutions —
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.)

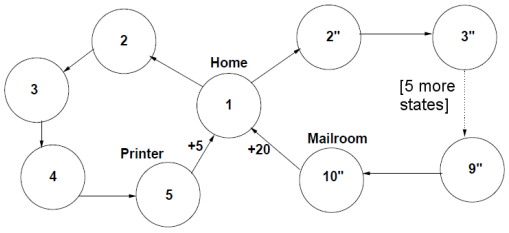

Figure 1: An illustrative example for R-learning (Schwartz, ibid.).

[*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.

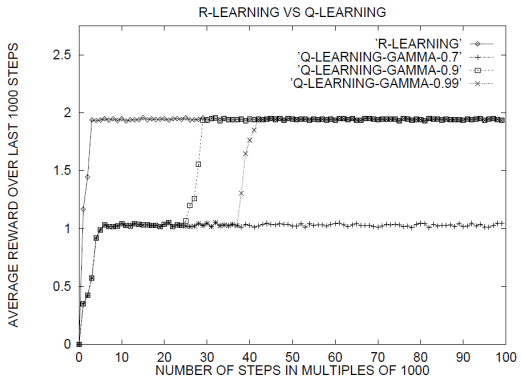

Figure 2: R-learning (Schwartz, ibid.) can solve certain problems much more quickly than for example Q-learning. Also note that Q-learning does prefer the shorter branch for small γ, while it has longer learning times for γ close to 1 (Schwartz, ibid.).

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.

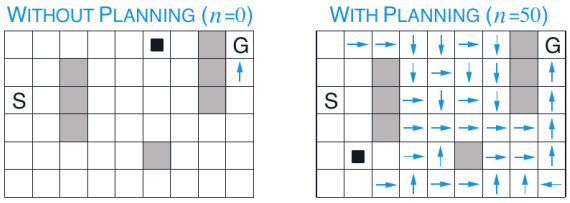

Figure 3: A simple maze task (Sutton and Barto, 2020, Fig. 8.3)

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 2 - 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?

Answer:

The learning rate η is usually not critical for simple deterministic problems, and relatively large values are often useful to reduce the time to convergence. However, if the initial TD-type errors are large compared to the relevant range of values, then it may be necessary to limit the learning rate from the beginning. If the rewards (or actions) are stochastic, then it is necessary to reduce the learning rate according to the Robbins-Monro conditions (see lecture 2, slide 12), although in practice a quicker decay of the learning rate is usually preferable, i.e. the learning rate may need to be decreased so that it reaches zero in finite time. See also momentum techniques (for example Sarigül and Avci [2018]). Also note that for RL methods based on function approximation, the Robbins-Monro conditions are not sufficient for convergence.

Difficulties can arise, when several learning rates (or the learning rate and the exploration rate) interact, e.g. if a sliding average with its own learning rate enters the updates, if a training algorithm is used to approximate the policy or value function etc. In this cases, some consideration of the relative "speed" of the respective learning processes as well as some experimentation may be necessary.

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 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.

Answer:

The answer is, arguably, both. The discount factor is occasionally included in the tuple defining Markov Decision Processes (MDP) Silver [accessed 2020], but is also frequently omitted and is not present in the original definition Bellman [1957]. That being said, the discount factor determines the relative importance of rewards, based on how close in time they are received. This affects what the optimal policy for the problem would be. If we then were to adopt the view that the discount factor is not part of the definition of an MDP, then an MDP would not in-and-of-itself completely define our control problem (we would not have fully defined our cost function).

There is work indicating that we can control the discount factor during training in a way that would improve the learnt policy, in terms of performance when deployed Jiang et al. [2015]. Specifically, this work defines a new "evaluation" discount factor γ_{eval}, which is smaller than the discount factor we have defined for our problem γ). This is then used to train a policy for a long horizon problem (as defined by γ) using short horizon samples (as defined by γ_{eval}). This means that the discount factor is used here as a training hyperparameter to trade off bias and variance (see also e.g. Schulman et al. [2015]). In particular, the authors show that you can see increasing the discount factor (and therefore the effective horizon) as increasing the complexity of the learnt model. And as you will recall from Machine Learning, we need to match the complexity of our model to the task at hand.

Part c

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

Answer:

Different algorithms have different parameters, so, if you read or hear about a new algorithm, try to get an idea whether their particular parameters are sensitive, need online adjustment, or depend strongly on the problem. In other words, own experience in using an algorithm is useful. Here, we restrict ourselves to the following parameters:

- The exploration rate ϵ has been discussed in the context of MAB and many of the features observed there, carry over also the general case, although this parameter needs to be checked in any particular case and to be adapted in many cases.
- Resolution parameters determine how many discretisation steps are made available for the description of the state of a problem. Likewise, dimensioning parameters decide the complexity of function approximation methods, e.g. the number of neuron is a network or the number of samples. Given

the problem complexity, these parameters are determined by the required efficiency of the algorithm and the available resources.

- Initialisation values can include any prior knowledge on the problem, but should otherwise not introduce an unwanted bias. E.g., optimistic initialisation, if done in a suitable way, introduces a bias towards exploration that is often desirable. In other cases, small random initial values are a good choice.
- The number of time steps should be large enough to reach a certain quality of a solution, i.e., so that given the exploration rate, the learning rate(s) and the complexity of the problem the solution can actually be found. It is often helpful to make a rough calculation to get an idea how long you'll have to wait for a solution or whether you should reconsider the choice of the parameters.

References

Richard Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957. ISSN 00959057, 19435274. URL http://www.jstor.org/stable/24900506.

Nan Jiang, Alex Kulesza, Satinder Singh, and Richard Lewis. The dependence of effective planning horizon on model accuracy. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, page 1181–1189, Richland, SC, 2015. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450334136.

Mehmet Sarigül and Mutlu Avci. Performance comparison of different momentum techniques on deep reinforcement learning. *Journal of Information and Telecommunication*, 2(2):205–216, 2018.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

David Silver. Applications of reinforcement learning in real world. https://www.davidsilver.uk/wp-content/uploads/2020/03/MDP.pdf, accessed 2020.