Reinforcement Learning Tutorial 8, Week 9
— with solutions —
Revision: MDPs & Function Approximation

Pavlos Andreadis

March 2025

Overview: The following tutorial questions relate to material taught in weeks 1 to 6 of the 2024-25 Reinforcement Learning course. They aim at encouraging engagement with the course material and facilitating a deeper understanding.

This week we stop to take a look back at some things we have done through the course till now. Specifically focusing on modelling Markov Decision Processes (MDPs), and function approximation. We will do so by looking even further back to the past, and into the 2017-18 Reinforcement Learning exam.

Problem 1 - Revision: MDP Modelling

[Adapted from RL exams in 2017-18]

Pamp the sailor was in a shipwreck and has been left stranded on an island. Though this island is now "home", it has no resources and Pamp occasionally sets out on a raft to scavenge for resources from the surrounding islands. Upon arriving on an island, Pamp accumulates a specific, known, amount of resources (except for the "home" island which never has any resources). Each island has a fixed amount of resources, which are replenished after each visit. Pamp can attempt to move from any island to any *other* island, but some times the sea currents will move the raft randomly to one of the islands (even the one Pamp is on at the moment).

1. Consider the control problem where the current state is specified by the current island Pamp is on, and the actions Pamp can take are to attempt a move towards another island. Assume that there are 2 other islands, except for "home" which is always the starting island (so 3 islands in total). Moreover, assume that Pamp has full knowledge of the amount of resources on each island, and that there is always a 90% chance of Pamp

transitioning towards the intended island, with the rest of the probability uniformly distributed across the remaining islands.

(a) Formulate a Markov Decision Process (MDP) for the problem of controlling Pamp's actions in order to maximise the accumulation of resources during an episode/trip. (Give the transition and reward functions in tabular format, or give the transition graph with rewards).

Answer:

If we define a_0 as the action for moving to the other island with the smallest index, and a_1 as the action for moving to the other island with the largest index, then the transition and reward function can be written as:

a_0	s_0/"home"	s_1	s_2
s_0 / "home"	$0.05, 0$	$0.90, r_1$	$0.05, r_2$
s_1	$0.90, 0$	$0.05, r_1$	$0.05, r_2$
s_2	$0.90, 0$	$0.05, r_1$	$0.05, r_2$

a_1	s_0/"home"	s_1	s_2
s_0 / "home"	$0.05, 0$	$0.05, r_1$	$0.90, r_2$
s_1	$0.05, 0$	$0.05, r_1$	$0.90, r_2$
s_2	$0.05, 0$	$0.90, r_1$	$0.05, r_2$

(b) If Pamp's trip ends upon returning to the "home" island, how would you modify the above MDP? (Similarly, "How would an MDP for this modified problem differ from the MDP for the above question?").

Answer:

Returning "home" should now put us in an absorbing state and the task becomes episodic. We can set the discount factor as $\gamma = 1$ (or remove the discount factor).

(c) Consider the discounted return from the state "home" for a single episode. For which of the models above in i) and ii) could this number be an accurate representation of the sum of resources gathered during that episode? (Assuming your rewards have been defined to represent the quantity of resources gathered when visiting each island).

Answer:

For the model defined for the episodic task in ii) (the assumption being that the discount factor was set to 1).

2. In the example at the beginning of this question, Pamp has access to a Transition and Reward function.

   (a) Assuming no access to the Reward and Transition functions, would Pamp be able to compute an optimal policy without leaving "home", and why? Considering a Reinforcement Learning algorithm in general, what is the property of not needing these two functions as input called?

Answer:

No, Pamp requires samples to compute the optimal policy (if Pamp is made to somehow have access to a simulator, then Yes). Model-free.

   (b) Consider any of the MDPs defined above, focusing on that your states are defined as the island Pamp is currently on. Which basic assumption of MDPs would be violated if the transition probabilities from one island to another also depended on the number of previously visited islands? If this assumption was violated, but you were asked to evaluate a plan for moving from island to island, which algorithm would you choose and why?

Answer:

The Markov Assumption/Property. Any Monte Carlo Policy Evaluation, because it is less affected by violations of the Markov property (or because it doesn't bootstrap).

   (c) Is the algorithm you chose well defined for continuing (non-episodic) tasks?

Answer:

Monte Carlo is not.

[*Are TD Learning and Dynamic Programming?*]

Problem 2 - Revision: Function Approximation

[Adapted from RL exams in 2017-18]

Consider the problem with Pamp the sailor in Problem 2, but with an infinite number of islands. Moreover, assume that Pamp can only ever see and choose between 2 different islands (*left* and *right*) to move towards and that Pamp can observe an estimate of the amount of resources on each of those 2 islands. If you were to formulate the control problem as an MDP:

1. What would be a good representation of state if Pamp had no memory of previously visited islands?

   ### Answer:

   Use the estimation of the reward on the left and right island, as well as the reward signal from arriving to this island.

2. What would be a good representation of state if Pamp could remember the previous island (in addition to the current one)?

   ### Answer:

   As above, plus the same information from the previous island. (Transitions are still stochastic, so unless that is taken into account somehow, it would be wrong to omit some of this information as redundant).

3. Define the linear approximate state-value function for one of the above two cases.

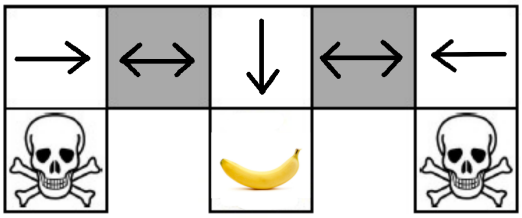
   ### Answer:

   $V_t = \boldsymbol{w}_t^T \, \boldsymbol{x}_s$, with, for example, $\boldsymbol{x}_s = \left(\tilde{r}_{\text{left}}, r_s, \tilde{r}_{\text{right}}\right)$

Problem 3 – Ambiguous state information

Discuss the aliased gridworld example[1], 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 $r(\text{skull}) \ll r(\text{banana})$.

[1] adapted from David Silver's lecture 7



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

$$\begin{aligned} R &= r(\text{banana})\,\gamma^2 \left(P(\text{right}) \sum_{k=0}^{\infty} \gamma^{2k} P(\text{left})^k + P(\text{left}) \sum_{k=0}^{\infty} \gamma^{2k} P(\text{right})^k \right) \\ &= r(\text{banana})\,\gamma^2 \left(P(\text{right}) \frac{1}{1-\gamma^2 P(\text{left})} + P(\text{left}) \frac{1}{1-\gamma^2 P(\text{right})} \right) \\ &= r(\text{banana})\,\gamma^2 \left(\frac{1-P(\text{left})}{1-\gamma^2 P(\text{left})} + \frac{P(\text{left})}{1-\gamma^2(1-P(\text{left}))} \right) \end{aligned}$$

which is maximal for $P(\text{right}) = P(\text{left}) = 0.5$, 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. $P(\text{left})$, set it to zero, and solve

for $P(\text{left})$. If $\gamma = 1$, then it does not matter how long it takes on either side, so any action probabilities are fine as long as $P(\text{right}) > 0$ and $P(\text{left}) > 0$.

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.