# Reinforcement Learning TD1

Dorin Doncenco

November 2023

## 1 TD1

**Warmup**

**Q. 1.1 What is the goal of a reinforcement learning algorithm ? In this framework what is the goal of an agent ? How can you formalize it mathematically ?**

The goal of reinforcement learning is to study the problem of how an agent should act in an environment; what rewards it maximises, and how to find the best policy for the agent. The goal of the agent is to maximise its reward.

With $\pi$ as the optimum policy, $\gamma$ as the discount factor, $r_t$ as the reward $r$ at timestep $t$ , the policy which maximises the expected reward value is written as:

$$\pi = argmax_{\pi}\mathbf{E}[\sum_{t=0}^{\infty}\gamma^t r_t] \tag{1}$$

**Q. 1.2 What make reinforcement learning different from supervised learning ? What makes reinforcement lear- ning hard ?**

In conventional learning, we have a lot of data; in supervised learning, we also have ground truth labels. With reinforcement learning we have neither of those. The data is obtained by "playing through an episode", after which a certain label is provided depending on how the reward function is formulated and how well the agent has performed.

**Q. 1.3 Give the formal definition of a Markov Decision Process (MDP). Give the meaning of it components.**

The MDP is a system defined by 4 components (S, A, P, R).

- $S$ is the state space (set of possible states)

- $A$ is the action space (set of possible actions)

- $P$ is the transition probability that in a given state $s$, a certain action $a$ will lead to a certain state $s'$: $P_a(s|s') = P(s_{t+1} = s'|s_t = s, a_t = a)$

- $R$ is the reward for transitioning from state $s$ to state $s'$ using action $a$.

**Q. 1.4 What is the Markov property ?**

The Markov property is that the future state $s_{t+1}$ only depends on the current state $s_t$ and action $a_t$; the way that the current state was achieved is irrelevant for determining the future state.

**Q. 1.5 What does the discount factor $\gamma$ mean or represent ?**

$\gamma$ represents the speed of decay of the reward as time steps go by. With $\gamma$ approaching 1, the reward function tends to $\infty$; with $\gamma$ approaching 0, the reward function cares about the immediate reward at timestep 0 i.e. low $\gamma$ means only the first reward matters, high $\gamma$ means that the system is rewarded for prolonging the amount of time steps in the episode.

**Q. 1.6 Can you give an example of setup with non deterministic transitions ?**

Assuming that opponent actions are determined, betting in poker before the river is revealed is a non-deterministic transition.

**Q. 1.7 What is the difference between a Deterministic and Stochastic Policy ? Give an example for both**

A deterministic policy has a certain action to be taken, for a given a state. A stochastic policy has a probability distribution of actions to take, for a given state.

Chess / stochastic policy: A Gaussian distribution policy for the values $[1, 8]$ where the number picked decides which pawn to move 2 steps forward in chess as the first move; this would be a stochastic policy for the first move.

Chess / deterministic policy: Employing an endgame tablebase to decide the next moves to lead to the end of the game is a deterministic policy.

**Q. 1.8 What is the difference of an episodic setup and a continuous one ? Give an example of each.**

An episodic setup is an environment which has an end which is the agent's goal. A continuous example has no end (except if the agent gets stuck and needs to be reset)

A continuous setup is a robot trying to keep a bike going without falling over; there is no goal end, and the episode technically "ends" when the bike falls down, but it could go forever, with no actions being taken.

An episodic setup is anything that has a clear end; in a poker game, the episode is from the entry to the table to the point where all players have lost their treasury and there is a winner.

**Q. 1.9 What does V function means or represents ?**

The V function determines the (expected) value of a certain next state; this helps us assign a score and determine the best action to take to achieve the most valuable state i.e.

$$V_\pi(S) = E(\sum_{t=0}^{\infty} \gamma^t r_t) \tag{2}$$

**Harder questions**

**Q. 2.1 What is Full Observability ?**

Full observality means that the agent has all the information necessary about the current state to take the optimal action; such that providing any more information about the world would not give the agent any more information than it is required. Chess is an example of fully observable environments; the configuration of all the pieces is available to the agent; more information such as the color of the board or its localisation in the world is not relevant to its decision.

**Q. 2.2 What is the Reward Hypothesis ?**

An RL agent has the goal of maximising the expected value of the reward it will receive, i.e. finding the policy:

$\pi = argmax_{\pi}\mathbf{E}[\sum_{t=0}^{\infty}\gamma^t r_t]$

**Q. 2.3 Suppose that an agent have 2 goals, climbing as high as possible and going as far as possible north. Can the reward for a step of the associated MDP problem be $Rt = (0, -1)$? Explain**

Assuming a 3-dimensional world where you can climb higher without going north, and you can go north without climbing higher, yes, a reward for an action can be $(0, -1)$. This would happen if, given the coordinate system $(x, y, z)$ with x being the north/south axis, y being irrelevant, and z being the altitude axis, the goal is to maximize $x$ and $z$. If the agent climbs down a ladder, lowering $z$ , without changing $x$, we can have the associated reward for this action of $(0, -1)$.

**Q. 2.4 How would you design the reward of an agent playing chess ? Does MDP seems to be a satisfactory framework for chess ? Explain**

The MDP is a satisfactory framework for optimizing a chess policy; the next state depends only on the current state, regardless of what the previous actions leading to the current state have been.

A reward function for an agent playing chess would be:

$$r(s) = (1 \text{ if } s = \text{win}), (-1 \text{ if } s = \text{loss}), ([-0.1, 0.1] \text{ if } s = \text{draw}), (0 \text{ otherwise}), \tag{3}$$

with the draw reward being decided on how important is a draw compared to the other conditions; a higher draw reward encourages the agent to play more conservative, a lower draw reward encourages riskier plays.

An alternative option is to reward the agent for taking opponent pieces and punish for losing pieces in according to their expected value; however, such a system is vulnerable to giving up an advantageous position for taking an opponent's piece.

**Q. 2.5 Let us consider the CartPole problem where on wants to maintain a stick vertically. The stick is fixed at the bottom to a cartpole that can move. Suggest a reward structure which would likely induce the desired behavior**

A strict reward function, with $\theta$ the angle between the stick and the ground:

$r(s_\theta) = 1$ if $\cos(\theta) = 0, 0$ otherwise,

However, such a reward function would take a long time to converge, a better function would be:

$r(s_\theta) = 1 - |cos(\theta)|$,

Such a reward function would encourage the agent to keep the bar as vertical as possible, but suboptimal solutions would still be possible. The $\gamma$ factor is not considered, since there is no preference over achieving verticality now vs later, and a system that eventually stabilises is better than a system that quickly reaches verticality but is unstable.

**A simple Maze**

**Q. 3.1 What is the space of states ?**

$S = \{s_1, s_2, ..., s_{27}\}$, where $s_i$ represents a box in the labyrinth.

We can also represent the state space as a set of 3 elements, $S = \{$ entrance, inside, outside $\}$

**Q. 3.2 What is the space of actions ?**

The space of actions is $A = \{$ north, east, west, south $\}$.

**Q. 3.3 Why do we give a negative reward when the agent does not reach the goal ?**

The negative reward might discourage the agent from exploring, as exploration leads to punishent? Theoretically, I don't see why the behaviour should be different compared to rewarding the agent only when the goal is reached, without punishment.

**Q. 3.4 Give the Bellman equation followed by V**

$$V(S_{t=0}) = R(S_{t=0}) + \sum_{S_{t+1} \in states} \gamma^{t+1=1} P(S_{t+1}|S_t, a)V(S_{t+1}), \tag{4}$$

or in other words: The expected value of the current state is equal to the reward of this state $S_t$, plus the expected value of all the next possible states $S_{t+1} \in states$, in proportion to the probability $P$ of that state being achieved (by the policy / action taken), discounted by the factor $\gamma$.

**Q. 3.5 Deduce what is V for each state for the optimal policy (the one where the agent takes the best decision at each step)**

We can compute the V function output for each state; starting with the transition T and the state next to the end goal as the value $V(S) = R(S) + \gamma T(goal|S, forward) = -1 + 0.9 * 0 = -1$, we can "backtrack" iteratively applying the value function all the way to the start state (Figure 1).

Knowing the amount of steps required to take, we can also generalize equation 4 as such:

$$V(S) = \sum_{i=0}^{steps} 0.9^i * -1 \tag{5}$$

**Crossing a river**

**Q. 4.1 Compute the value function V for a constant policy,** $\forall s \in S, \pi(s) = right$

$V(S_N) = 100 = \sum_{i=0}^{\infty} \gamma^i r_i | S_0 = S_N) = \gamma_0 r_0 = r_N * 1$

For the first step before the goal:

$$V(S_{N-1}) = \gamma^1 r_N + \sum_{i=0}^{0} \gamma^i r_{N-1+i} \tag{6}$$

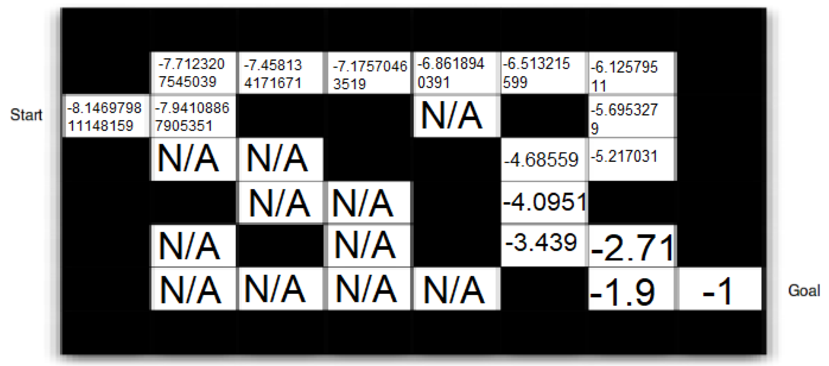

Figure 1: The amusing task of computing the value of each state! Since we consider only the optimal policy considering the starting point of "Start", some states which will never be achieved have the value set to N/A (or $-\infty$ or $-99999$)

For the second step before the goal:

$$V(S_{N-2}) = \gamma^2 r_N + \sum_{i=0}^{1} \gamma^i r_{N-1+i} \tag{7}$$

For the nth step before the goal:

$$V(S_{N-n}) = \gamma^n r_N + \sum_{i=0}^{n-1} \gamma^i r_{N-n+i} \tag{8}$$

Alternatively, it can be written as a recursive expression:

$$V(S_{N-n}) = \gamma * V(S_{N-n+1}) + r_{S_{N-n}} \tag{9}$$

**Q. 4.2 Same question with** $p(s_i, right, s_{i+1}) = .9; p(s, right, s) = .1.$. Denoting $p(success)$ and $p(fail)$ as the transition to the right being successful (0.9) and failing (0.1),

$V(S_N) = r_N = 100$

$V(S_{N-1}) = 81 + 0.09 * V(S_{N-1}) = \gamma p(success) V(S_N) + \gamma p(fail) V(S_{N-1}) = 0.81 * V(S_N) + 0.09 * V(S_{N-1}) = 0.9 * 0.9 * 100 + 0.9 * 0.1 * V(S_{N-1})$

We can solve for $V(S_{N-1})$ :

$0.91 * V(S_{N-1}) = 81 \leftrightarrow V(S_{N-1}) \approx 89.01$

Now we can compute the generalized value function for any state:

$$V(S_{N-n}) = p(success) * \gamma * V(S_{N-n+1}) + p(fail) * \gamma * V(S_{N-n}) = 0.8901 * V(S_{N-n+1}) \tag{10}$$

or simpler, we rediscover a different discount factor:

$$V(S_{N-n}) = 0.8901^n * V(S_N) \tag{11}$$

**Computer store management**

**Q. 5.1 Is it an episodic or a continuous problem ? Give a justification .**

This is a continuous problem; each week, we have leftover computers from previous weeks, thus we cannot divide the episodes by weeks. If we had no storage, this would become an episodic problem; in such a scenario, there would be no information passed onto the next week from the previous ones.

**Q. 5.2 What would be the state space ? Is it discrete or continuous ?**

The state space is S(stock). It is discrete, as we cannot have a float number of computers in stock. We do not need to have the amount of computers ordered nor sold in the current state, unless we want to form some sort of memory in our system.

**Q. 5.3 What would be the action space ? Is it discrete or continuous ? Is it stochastic or deterministic ?**

The action space $a_t$ is discrete, and it represents the amount $t$ of computers to buy at the start of the week. The assumption is that the selling of computers is automatic $min(\text{demand}, a_t + \text{stock})$.

The action space is stochastic: If an action is taken, the state will be affected by the demand which is randomly determined throughout the week; thus the transition state is not certain.

**Q. 5.4 Hard question As the owner, you want to maximise your balance each week. Write the reward as a function of the actions and the state for each week t. rt(at, st)**

With $a_t$ the number of computers ordered, $s_t$ the number of computers in stock (from state at week t), $M$ the capacity of the warehouse, $h$ the maintenance cost, $c$ the order cost per unit, $c_0$ the base order cost if an order happens, $(a_t > 0)$ as a boolean statement which returns 1 if an order happens, 0 otherwise, $p$ the price for which a unit is sold, $D_t$ the demand for the week, and the assumption that computers over storage limit M get stolen at the end of the week, we can write the reward function as:

$$r_t(a_t, s_t) = -h(min(M, max(0, (a_t + s_t - D_t)))) - ca_t - c_0 * (a_t > 0) + pD_t \quad (12)$$

**Golf**

We can describe the value of the decision $q_\pi(s, a)$ as the sum between reward for reaching the next state and the value of the next state.

$$q_\pi(s, a) = R(s, a) + \gamma V(s') \quad (13)$$

We can describe the value of the next state as the sum of all future $i$ amount of $s'$ states $s' \in states$, and the reward for reaching those steps by taking the action determined by the policy in that state, discounted by the respective gamma at step i.

$$q_\pi(s,a) = R(s,a) + \sum_{\{i,s'\}\in\{steps,states\}} \gamma^i R(s', \pi(a'|s')) \tag{14}$$