Introduction to Reinforcement Learning

Chao Tao

Jan. 17, 2020

1 Learning Setup

Reinforcement Learning (RL) has many real-world applications. Recently, famous applications include

- Games: AlphaGo ([7]), AlphaStar ([9]),
- Robotics: solve rubik's cube with a robot hand ([1]),
- AutoML: neural architecture search ([8]),

to name a few.

Intuitively, RL is about learning to collect the maximum rewards through interactions with poorly understood environments. The environment is usually modeled as a Markov Decision Process (MDP). A MDP is made up of four elements i.e., state $\mathcal{S}$, action $\mathcal{A}$, transition probability $\mathbb{P}$ and reward function R, where $\mathbb{P}(s' \mid s, a)$ and $R(s, a)$ represent the probability of transitioning to state s' and the reward at state s when action a is taken respectively. Note that the definition implies that in every state $s \in \mathcal{S}$, there are $|\mathcal{A}|$ available actions to take. After the definition of the environment, we can use the following diagram to describe a RL problem.

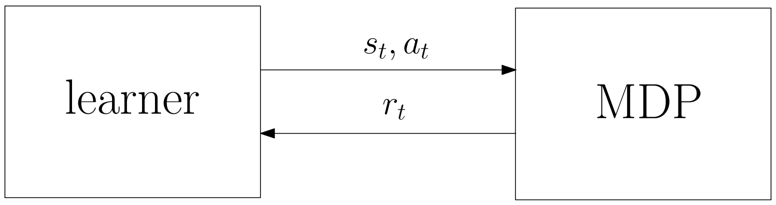

Figure 1: Learner-Environment Interface

Usually, in a RL problem, a learner interacts with the MDP in steps. In each time step t, the learner observes the state s_t of the MDP and then takes an action a_t after which he/she will receive an immediate reward r_t.

For simplicity, we only consider the case when both of $|\mathcal{S}|$ and $|\mathcal{A}|$ are *finite* (or *tabular* in other words). There are some works considering extending tabular MDP to the infinite case, which is beyond the scope of our discussion. Apart from that, we also assume the reward function R and the initial state s_1 are *deterministic*.

According to the mathematical formulation of the goal, we divide RL problems into three different categories.

1.1 Infinite Discounted MDP

An instance of this kind of problems can be represented by a 6-tuple $(\mathcal{S}, \mathcal{A}, \mathbb{P}, R, \gamma, s_1)$ where $\mathcal{S}, \mathcal{A}, \mathbb{P}$ and R share the same definition as a MDP, $\gamma \in (0, 1)$ is the discount factor and s_1 is the initial state.

The goal is to find an algorithm or policy π to be equipped by the learner such that the expected total rewards

$$\mathbb{E}\left[\sum_{t=1}^{+\infty} \gamma^{t-1} R^{\pi}(s_t, a_t)\right]$$

is maximized, where the superscript π is used to denote that the action is taken according to policy π. When it clear from the context, we usually omit the superscript.

1.2 Infinite Undiscounted MDP

An instance of this kind of problems can be represented by a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathbb{P}, R, s_1)$ where both of the notations share the same meaning as before.

The goal is to find an algorithm or policy π to be equipped by the learner such that the average rewards

$$\mathbb{E}\left[\liminf_{T \to +\infty} \frac{1}{T} \sum_{t=1}^{T} R^{\pi}(s_t, a_t)\right]$$

is maximized.

1.3 Episodic MDP

An instance of this kind of problems can be represented by a 6-tuple $(\mathcal{S}, \mathcal{A}, \mathbb{P}, R, H, s_1)$ where H means after each H time steps the MDP will restart from the initial state s_1 and other notations share the same meanings as before.

The goal is to find an algorithm or policy π to be equipped by the learner such that after $T = KH$ time steps, the total rewards

$$\mathbb{E}\left[\sum_{k=1}^{K} \sum_{h=1}^{H} R^{\pi}(s_{k,h}, a_{k,h})\right]$$

is maximized, where $s_{k,h}$ and $a_{k,h}$ denote the state and the action at tth time step. In episodic MPD, we usually add a redundant state $s_{k,H+1}$ to denote the next state after action $a_{k,H}$.

2 Optimality

Take an infinite discounted MDP as an example, in general, the optimal action a_{t+1} may depend on the whole history i.e., $\mathcal{H}_t \overset{\text{def}}{=} (s_1, a_1, \dots, s_t, a_t, s_{t+1})$ and/or some external randomness. Fortunately, there is no harm to the final goal if we only consider some restricted policy.

Let us first introduce some concepts. We call a policy is *Markov* if it only considers the last state. A policy is called *stationary* if the policy does not change with time. A *deterministic* policy is such that the action to be taken is deterministic.

After the knowledgement of the aforementioned concepts, we are ready to introduce the following theorems.

Theorem 1. *For infinite discounted/undiscounted MDPs, there always exists a stationary, deterministic and Markov policy that is optimal for all starting states simultaneously*

Proof. For the infinite discounted MDP case, please refer to Theorem 6.2.7 of [6] (reference due to Shipra Agrawal). For the undiscounted case, please refer to the optimality criteria section in [3]. □

Theorem 2 ([2])**.** *For episodic MDPs, there always exists a deterministic and Markov policy that is optimal for all starting states simultaneously*

Remark 3. *This theorem implies that the optimal policy may change within one episode.*

3 Bellman Equations

From now on, we are assuming the problem we are facing is an infinite discounted MDP and the optimal policy is Markov, stationary and deterministic. For other settings, the corresponding results are similar.

3.1 Value Functions

Given a policy π, we call $V^\pi(\cdot) : \mathcal{S} \to \mathbb{R}$ the state-value function where $V^\pi(s)$ denotes the expected rewards at state s when the learner follows policy π. We also call $Q^\pi(\cdot, \cdot) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the state-action-value or Q-value function where $Q^\pi(s, a)$ denotes the expected rewards at state s when the learner first takes action a and follows policy π afterwards. We use π^* to denote the optimal policy. And we use $V^*(\cdot)$ and $Q^*(\cdot, \cdot)$ to denote its state-value function and Q-value function respectively.

3.2 Bellman Expectation Equations

For any policy π, we have the following *Bellman Expectation Equations* holds

$$V^\pi(s) = R(s, \pi(s)) + \gamma \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}(s' \mid s, \pi(s)) V^\pi(s'),$$

and

$$Q^\pi(s, a) = R(s, a) + \gamma \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}(s' \mid s, a) Q^\pi(s, \pi(s')).$$

3.3 Bellman Optimality Equations

For the optimal policy π^*, we have the following *Bellman Optimality Equations* holds

$$V^*(s) = \max_{a \in \mathcal{A}} \left\{ R(s, a) + \gamma \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}(s' \mid s, a) V^*(s') \right\},$$

and

$$Q^*(s, a) = R(s, a) + \gamma \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}(s' \mid s, a) \max_{b \in \mathcal{A}} Q^*(s', b).$$

4 State Occupancy

Let V^π be the column vector $[V^\pi(s)]^T_{s\in\mathcal{S}}$, R^π be the column vector $[R(s,\pi(s))]^T_{s\in\mathcal{S}}$ and $\mathbb{P}^\pi$ be the matrix $[\mathbb{P}(s'|s,\pi(s))]_{s\in\mathcal{S},s'\in\mathcal{S}}$. Then according to Bellman Equation, we have

$$V^\pi = R^\pi + \gamma \mathbb{P}^\pi V^\pi. \tag{1}$$

Lemma 4. *There always exists a solution i.e., V^π to* (1).

Proof. Rewrite (1), we obtain $(I-\gamma\mathbb{P}^\pi)V^\pi = R^\pi$. It suffices to prove matrix $I-\gamma\mathbb{P}^\pi$ is invertible. Let x be an arbitrary vector. Note that

$$\begin{aligned}
&(I-\gamma\mathbb{P}^\pi)x = 0\\
\Leftrightarrow\ & x = \gamma\mathbb{P}^\pi x\\
\Rightarrow\ & \|x\|_\infty = \|\gamma\mathbb{P}^\pi x\|_\infty \le \gamma\|x\|_\infty\\
\Rightarrow\ & (1-\gamma)\|x\|_\infty \le 0\\
\Rightarrow\ & x = 0,
\end{aligned}$$

which completes the proof. □

Hence $V^\pi = (I-\gamma\mathbb{P}^\pi)^{-1}R^\pi = (I+\sum_{t=1}^{+\infty}(\gamma\mathbb{P}^\pi)^t)R^\pi$, from which we derive that

$$\begin{aligned}
V^\pi(s) &= \sum_{s'\in\mathcal{S}}\left(\sum_{t=1}^{+\infty}\gamma^{t-1}\,\mathbf{Pr}(s_t=s'\mid s_1=s)\right)R(s',\pi(s')),\\
&= \sum_{s'\in\mathcal{S}} d_s^\pi(s')R(s',\pi(s')),
\end{aligned}$$

where we have defined $d_s^\pi \overset{\text{def}}{=} [\sum_{t=1}^{+\infty}\gamma^{t-1}\,\mathbf{Pr}(s_t=s'\mid s_1=s)]^T_{s'\in\mathcal{S}}$, which is called *discounted state occupancy* vector. And we also define $\eta_s^\pi = (1-\gamma)d_s^\pi$ and call it *state occupancy* vector. Both d_s^π and η_s^π notations come from Lecture 1 of [4].

5 Q-learning

Suppose the transition probability $\mathbb{P}$ is not known beforehand, how should the learner behave and what is the best he/she can do? The high level idea is to estimate Q^*-value function, and then behave greedily according to the estimated $\widetilde{Q}$-value function. But then how to get a good estimate of Q^*-value function? Recall the online version of mean estimation

$$\overline{X}_t = \frac{\sum_{i=1}^t X_i}{t} = \overline{X}_{t-1} + \frac{1}{t}\cdot(X_t - \overline{X}_{t-1}),$$

where X_t is an unbiased estimation of X. Similarly, we want to estimate $\widetilde{Q}_t(x,a)$ using $\widetilde{Q}_{t-1}(x,a)$. Hence the new update rule becomes

$$\widetilde{Q}_{t+1}(x_t,a_t) = \widetilde{Q}_t(x_t,a_t) + \alpha\cdot(\overline{X}_{Q^*(x_t,a_t)} - \widetilde{Q}_t(x_t,a_t)),$$

where α is the so-called *learning rate* and $\overline{X}_{Q^*(x_t,a_t)}$ denotes an unbiased estimation of $Q^*(x_t, a_t)$. However, it is hard to get $\overline{X}_{Q^*(x_t,a_t)}$. In Q-learning, it uses $R(x_t, a_t) + \gamma \cdot \max_{a \in \mathcal{A}} \widetilde{Q}_t(x_{t+1}, a)$ to approximate $\overline{X}_{Q^*(x_t,a_t)}$. The details are shown in the following Algorithm 1.

Algorithm 1: Q-learning

```
1 initialization: $\widetilde{Q}_t(x, a) = 0$ for every $(x, a) \in \mathcal{S} \times \mathcal{A}$
2 for time $t = 1$ to $+\infty$ do
3     if $t > 1$ then
4         call Algorithm 2 to compute $\widetilde{Q}_t(\cdot, \cdot)$ and $\widetilde{V}_t(\cdot)$
5     observe state $x_t$
6     take action $a_t = \mathrm{argmax}_{a \in \mathcal{A}} \widetilde{Q}_t(x_t, a)$
```

Algorithm 2: Computation of $\widetilde{Q}_t(\cdot, \cdot)$ and $\widetilde{V}_t(\cdot)$

```
1 for every state-action pair $(x, a)$ do
2     if $(x, a) = (x_{t-1}, a_{t-1})$ then
3         $\widetilde{Q}_t(x, a) = \widetilde{Q}_{t-1}(x, a) + \alpha_t(x, a)(R(x, a) + \gamma \widetilde{V}_{t-1}(x_t) - \widetilde{Q}_{t-1}(x, a))$
4     else
5         $\widetilde{Q}_t(x, a) = \widetilde{Q}_{t-1}(x, a)$
6 for every state $x \in \mathcal{S}$ do
7     $\widetilde{V}_t(x) = \max_{a \in \mathcal{A}} \widetilde{Q}_t(x, a)$
```

Here $\alpha_t(x, a)$ is the learning rate. Note that when $(x, a) \neq (x_{t-1}, a_{t-1})$ at time t, $\alpha_t(x, a) = 0$.

Theorem 5. *When $\sum_{t=1}^{+\infty} \alpha_t(x, a) = +\infty$ and $\sum_{t=1}^{+\infty} \alpha_t(x, a)^2 < +\infty$ for all $(x, a) \in \mathcal{S} \times \mathcal{A}$, it can be guaranteed that in Algorithm 1, $\widetilde{Q}_t$ converges to Q^* in probability.*

Proof. See [5]. □

Remark 6. *To apply Theorem 5 i.e., to prove that Q-learning algorithm converges, we must set the learning rate such that each state-action pair (x, a) is visited infinitely often.*

References

[1] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[2] Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In *ICML*, pages 263–272, 2017.

[3] Eugene A. Feinberg and Jefferson Huang. On the reduction of total-cost and average-cost mdps to discounted mdps. *Naval Research Logistics (NRL)*, 66(1):38–56, 2019.

[4] Nan Jiang. Cs 598 statistical reinforcement learning, 2019.

[5] Francisco S Melo. Convergence of q-learning: A simple proof. *Institute Of Systems and Robotics, Tech. Rep*, pages 1–4, 2001.

[6] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., USA, 1st edition, 1994.

[7] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[8] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.

[9] Oriol Vinyals, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojciech M Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, et al. Alphastar: Mastering the real-time strategy game starcraft ii. *DeepMind blog*, page 2, 2019.