Basics of Reinforcement Learning

Sina Baghal

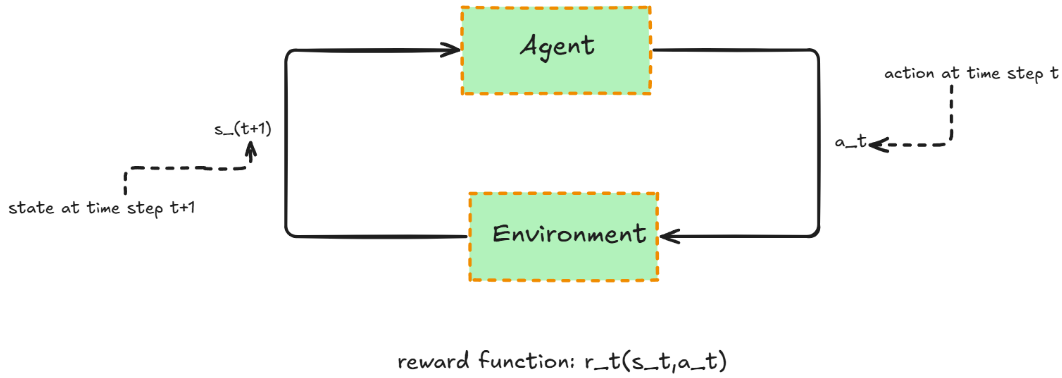
Abstract

This tutorial provides an introduction to the fundamentals of reinforcement learning. The main reference is the video lecture series by Sergey Levine.

Contents

1 What is RL?

RL In reinforcement learning, there is an *agent* and an *environment*. At time step t, the state is denoted by s_t. Given state s_t, the agent takes an action a_t resulting in a reward value $r_t := r(s_t, a_t)$.



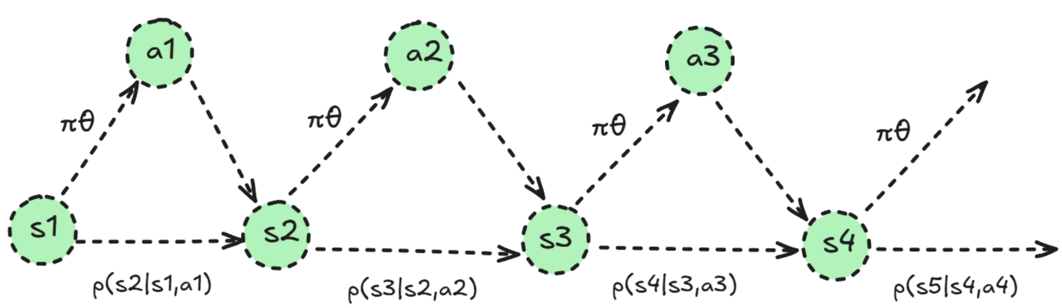
Policy The agent's *policy* is parameterized by π_θ, where $\pi_\theta(\cdot \mid s_t)$ defines a probability distribution over possible actions at time t, given the state s_t.

RL Goal The goal of an RL algorithm is to maximize the *expected cumulative reward*:

$$\mathrm{argmax}_\theta \, \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)\right],$$

where $0 \leq \gamma < 1$ and T are the discount factor and horizon resp. Notice that:

- More weight is placed on earlier steps.
- $\mathbb{E}_{\pi_\theta}$ is a smooth function of θ where r itself may not be (e.g., $r \in \{\pm 1\}$).
- s_t is independent of s_{t-1} (*Markov Property*).



MDP A *Markov Decision Process* (MDP) consists of a state space $\mathcal{S}$ and an action space $\mathcal{A}$, along with a transition operator $\mathcal{T}$ and a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_+$. An MDP allows us to write a probability distribution over trajectories:

$$p_\theta(\tau) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t), \quad \text{where } \tau = (s_1, a_1, \ldots, s_T, a_T).$$

2 Imitation Learning

The analogous concept in reinforcement learning, compared to supervised learning, is called *imitation learning*, where the agent learns by mimicking expert actions. However, imitation learning often does not work well in practice due to the *distributional shift problem*. This arises because, in supervised learning, samples are assumed to be i.i.d., while in reinforcement learning the agent's past actions affect future states.

Assume that π^* is the expert policy and the learned policy π_θ makes an error with probability at most ϵ under the training distribution:

$$\Pr_{s_t \sim p_{\text{train}}} \left[\pi_\theta(s_t) \neq \pi^*(s_t) \right] \leq \epsilon.$$

Then,

$$p_\theta(s_t) = (1-\epsilon)^t p_{\text{train}}(s_t) + (1 - (1-\epsilon)^t) p_{\text{mistake}}(s_t).$$

Denote $c_t(s_t, a_t) = 1_{\{a_t \neq \pi^*(s_t)\}} \in \{0, 1\}$. Then the total number of times the policy π_θ deviates from the optimal policy grows quadratically with T:

$$\begin{aligned}
\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} c(s_t, a_t)\right] &= \sum_{t=0}^{T} \int p_\theta(s_t) c(s_t, a_t) ds_t \\
&= \sum_{t=0}^{T} (1-\epsilon)^t \int p_{\text{train}}(s_t) c(s_t, a_t) ds_t + \sum_{t=0}^{T} (1 - (1-\epsilon)^t) \int p_{\text{mistake}}(s_t) c(s_t, a_t) ds_t \\
&\leq \sum_{t=0}^{T} (1-\epsilon)^t \epsilon + \sum_{t=0}^{T} 1 - (1-\epsilon)^t \\
&\leq \sum_{t=0}^{T} (1-\epsilon)^t \epsilon + 2\epsilon \sum_{t=0}^{T} t \\
&= \epsilon \cdot \mathcal{O}(T^2)
\end{aligned}$$

This bound is achieved in the *tightrope walking* problem Figure 1, where the agent must learn to go straight; otherwise, it will enter unknown territory. Imitation learning can still be useful with some modifications, such as including bad actions along with corrective steps.

Figure 1: A tightrope walker.

3 REINFORCE

An MDP allows us to rewrite the goal of RL as the following optimization problem:

$$\text{argmax}_\theta \; J(\theta) := \mathbb{E}_{\tau \sim p_\theta}[r(\tau)] = \int p_\theta(\tau) r(\tau) d\tau,$$

enabling a direct policy differentiation:

$$\begin{aligned}
\nabla_\theta J(\theta) &= \int \nabla_\theta p_\theta(\tau) r(\tau) d\tau \\
&= \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) r(\tau) d\tau \\
&= \mathbb{E}_{\tau \sim p_\theta} \nabla_\theta \log p_\theta(\tau) r(\tau) \\
&= \mathbb{E}_{\tau \sim p_\theta} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \right) \cdot \left(\sum_{t=1}^{T} r(s_t, a_t) \right) \qquad \nabla_\theta p(s_{t+1}|s_t, a_t) = 0
\end{aligned}$$

We are now ready to state the first policy gradient method:

REINFORCE

1. Run the current policy N times to generate sample τ_i for $i = 1, \ldots, N$.
2. Compute the Monte Carlo estimate:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \cdot \left(\sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right)$$

3. Apply Gradient Ascent: $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$.

4 Variance Reduction

One of the main issues with REINFORCE is the high variance in the reward term $\sum_{t=1}^{T} r(s_{i,t}, a_{i,t})$. In this section, we introduce some techniques to reduce this variance.

Causality As a first step toward variance reduction, we apply the *causality trick*:

Policy at time t' cannot impact reward at time $t < t'$.

Using which, the policy gradient is estimated as below:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \left(\sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'}) \right)$$

The term $\sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'})$ is referred to as the *reward-to-go*.

Value Functions The next idea is to replace the reward-to-go with a function estimator. To understand why this matters, see Figure 2. Notice two things: the ideal target for the reward-to-go function is the quantity $Q(s_{i,t}, a_{i,t}) = \sum_{t'=t}^{T} \mathbb{E}_{\pi_\theta}[r(s_{t'}, a_{t'})|s_{i,t}, a_{i,t}]$ rather than the single-sample estimate $\sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'})$. This represents the *value* of state $s_{i,t}$ under the current policy where action $a_{i,t}$ is taken at state $s_{i,t}$. Another advantage is that, as shown in Figure 2, if the state $s'_{i,t}$ is quite close to $s_{i,t}$ and $p(s_{t+1}|s'_{i,t}, a'_{i,t}) \approx p(s_{t+1}|s_{i,t}, a_{i,t})$, we expect their reward-to-go values to be similar. However, when working with a single-sample estimate, this relationship may easily be violated.

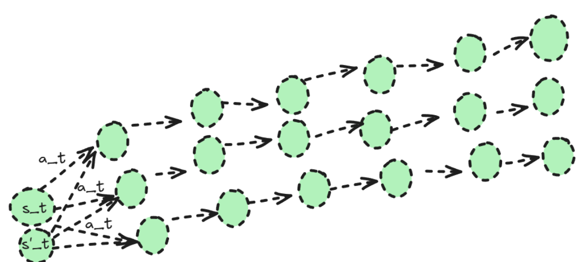


Figure 2: Value function fitting for variance reduction

Baselines Translation of the reward $r \mapsto r - b$ can help reduce the variance. Assuming this translation,

$$\begin{aligned}\mathrm{Var}[\nabla_\theta J(\theta)] &= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left(\nabla_\theta \log p_\theta(\tau)(r(\tau) - b)\right)^2 - \left(\mathbb{E}_{\tau \sim p_\theta(\tau)} \nabla_\theta \log p_\theta(\tau)(r(\tau) - b)\right)^2 \\ &= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left(\nabla_\theta \log p_\theta(\tau)(r(\tau) - b)\right)^2 - \left(\mathbb{E}_{\tau \sim p_\theta(\tau)} \nabla_\theta \log p_\theta(\tau) r(\tau)\right)^2\end{aligned}$$

Table 1: Value Functions

Q-function (reward-to-go)	$Q^{\pi_\theta}(s_t, a_t)$	$\sum_{t'=t}^{T} \mathbb{E}_\theta[r(s_{t'}, a_{t'})\|s_t, a_t]$
Value function	$V^{\pi_\theta}(s_t)$	$\mathbb{E}_{a_t \sim \pi_\theta(a_t\|s_t)}\left[Q^{\pi_\theta}(s_t, a_t)\right]$
Advantage function	$A^{\pi_\theta}(s_t, a_t)$	$Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t)$

Appropriate choice of b can therefore reduce the variance. A proper choice is the expected value of Q function. Table 1 summarizes value functions used throughout. Note

$$\begin{aligned} Q(s_t, a_t) &= r(s_t, a_t) + \mathbb{E}_{s_{t+1} \sim p(.|s_t, a_t)} V^{\pi_\theta}(s_{t+1}) \\ &\approx r(s_t, a_t) + V^{\pi_\theta}(s_{t+1}) \end{aligned}$$

The following policy gradient therefore favors a lower variance.

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot [r(s_t, a_t) + V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)]$$

Discounts The discount factor also helps reduce variance, as terms further in the horizon are weighted less. We then arrive at the following policy gradient:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot \left[r(s_t, a_t) + \gamma \hat{V}_\phi^{\pi_\theta}(s_{t+1}) - \hat{V}_\phi^{\pi_\theta}(s_t)\right]$$

Here $\hat{V}_\phi$ estimates V.

5 Bias Reduction

The policy gradient derived in the previous section, while enjoying low variance, is prone to higher bias. We tune this bias-variance trade-off as follows: n-step return estimator is:

$$\hat{A}_n^{\pi_\theta}(s_t, a_t) = \sum_{t'=t}^{t+n} \gamma^{t'-t} r(s_{t'}, a_{t'}) + \gamma^n \hat{V}_\phi(S_{t+n}) - \hat{V}_\phi(S_t)$$

For $n = 1$, we recover the previously mentioned policy gradient. As $n \to +\infty$, the bias is reduced while the variance increases. To manage this trade-off, we define

$$\begin{aligned} \hat{A}_{GAE}^{\pi_\theta} &= \sum_{n=1}^{+\infty} \lambda^{n-1} \hat{A}_n^{\pi_\theta} \\ &= \sum_{t'=t}^{+\infty} (\gamma\lambda)^{t'-1} \delta_{t'} \qquad \delta_{t'} = r(s_{t'}, a_{t'}) + \gamma \hat{V}_\phi(s_{t'+1}) - \hat{V}_\phi(s_{t'}) \end{aligned}$$

We therefore arrive at the following policy gradient.

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot \hat{A}^{\pi_\theta}_{GAE}(s_t, a_t)$$

6 PPO

Next, we explain Proximal Policy Optimization (PPO) algorithm [Ope25]. See Algorithm 1.

Algorithm 1 PPO-Clip [Sch+17]

Input: Initial policy parameters θ_0, initial value function parameters ϕ_0

for $k = 0, 1, 2, \dots$ **do**

  Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment

  Compute rewards-to-go $\hat{R}_t$ and advantage estimates $\hat{A}_t$

  # Option A (var ↑): $\hat{R}_t = r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t} r_T$ and $\hat{A}_t = \hat{R}_t - V_{\phi_k}(s_t)$

  # Option B (var ↓): $\hat{A}_t = \sum_{\ell=0}^{T-1} (\gamma\lambda)^\ell \delta_{\ell+t}$ w/ $\delta_t = r_t + \gamma V_{\phi_k}(s_{t+1}) - V_{\phi_k}(s_t)$. $\hat{R}_t = \hat{A}_t + V_{\phi_k}(s_t)$

  Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_\theta \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} L(s_t, a_t, \theta_k, \theta)$$

  where

$$L(s, a, \theta_k, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a),\ \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon \right) A^{\pi_{\theta_k}}(s, a) \right)$$

  Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_\phi \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2$$

end for

6.1 Why Clip?

The loss function is designed so that

The new policy does not benefit by going far away from the old policy.

Table 2 summarizes the components of the PPO loss function where $r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}$.

$A^{\pi_{\theta_k}}(s,a)$	$r(\theta)$	Clip when?	Clipped	Unclipped
$+$	$\uparrow$	$r(\theta) \geq 1+\epsilon$	$(1+\epsilon) \cdot A^{\pi_{\theta_k}}(s,a)$	$r(\theta) \cdot A^{\pi_{\theta_k}}(s,a)$
$-$	$\downarrow$	$r(\theta) \leq 1-\epsilon$	$(1-\epsilon) \cdot A^{\pi_{\theta_k}}(s,a)$	$r(\theta) \cdot A^{\pi_{\theta_k}}(s,a)$

Table 2: PPO Clipped Loss

Utilize the following fact to drive column Clipped

$$1-\epsilon \leq \mathrm{clip}(x, 1-\epsilon, 1+\epsilon) \leq 1+\epsilon$$

e.g., $x \leq 1-\epsilon \Rightarrow x \leq \mathrm{clip}(x, 1-\epsilon, 1+\epsilon)$ etc.

References

[Ope25] OpenAI. *Proximal Policy Optimization (PPO)*. 2025. URL: `https://spinningup.openai.com/en/latest/algorithms/ppo.html` (visited on 10/21/2025).

[Sch+17] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).