Crash Coruse in Reinforcement Learning – Cheat Sheet

Basic notation:

- $o \in \mathbb{O}$ – observation (vector of real numbers),
- $s \in \mathbb{S}$ – state, $o_t = s_t$ for all t,
- $a \in \mathbb{A} = \{a^{(1)}, \ldots, a^{(A)}\}$ – action,
- $r \in \mathbb{R}$ – reward,
- $\tau = (s_1, a_1, r_1, s_2, \ldots, s_T, a_T, r_T, s_{\text{END}})$ – trajectory,
- $R_t = r_t + r_{t+1} + r_{t+2} + \cdots + r_T = \sum_{j=t}^{T} r_j$ – return,
- $R_t^\gamma = \gamma^0 r_t + \gamma^1 r_{t+1} + \gamma^2 r_{t+2} + \ldots + \gamma^{T-t} r_T = \sum_{j=t}^{T} \gamma^{j-t} r_j$ – discounted return,
  - $0 < \gamma < 1$, usually $0.9 < \gamma < 1$,
  - $\gamma = 1 - \frac{1}{h}$, where h is the horizon,
- $R(\tau) = R(s_1, a_1, r_1, s_2, \ldots, s_T, a_T, r_T, s_{\text{END}}) = \sum_{j=1}^{T} r_j$.

Probabilities:

- $p(s_1)$ – initial state (environment),
- $\pi_\theta(a_t \mid s_t)$ – action (agent),
- $p(r_t, s_{t+1} \mid s_t, a_t)$ – state transition (environment),
- $p(r_t, s_{\text{END}} \mid s_t, a_t)$ – end of episode (environment),
- $\pi_\theta(\tau) = p(s_1)\ \pi_\theta(a_1 \mid s_1)\ p(r_1, s_2 \mid s_1, a_1) \ldots p(r_T, s_{\text{END}} \mid s_T, a_T)$ – trajectory (agent+environment).

Policy:

$$\pi : \mathbb{O} \ni o \mapsto \Big(\pi(a^{(1)} \mid o), \ldots, \pi(a^{(A)} \mid o)\Big) \in \mathbb{R}^A$$
$$0 \le \pi(a^{(1)} \mid o), \ldots, \pi(a^{(A)} \mid o) \le 1$$
$$\pi(a^{(1)} \mid o) + \ldots + \pi(a^{(A)} \mid o) = 1$$

1 Policy Gradient

Objective:

$$\mathop{\mathbb{E}}_{\tau\sim\pi_\theta} R(\tau) \approx \frac{1}{N}\sum_{j=1}^{N} R(\tau_j)$$

$$\begin{aligned}
\nabla_\theta \mathop{\mathbb{E}}_{\tau\sim\pi_\theta} R(\tau) &= \mathop{\mathbb{E}}_{\tau\sim\pi_\theta} \frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} R(\tau) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta} \sum_{t=1}^{T} \frac{\nabla_\theta \pi_\theta(a_t \mid s_t)}{\pi_\theta(a_t \mid s_t)} R(\tau) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta} \sum_{t=1}^{T} \nabla_\theta \ln \pi_\theta(a_t \mid s_t) \cdot R(\tau) \\
&= \mathop{\mathbb{E}}_{\tau\sim\pi_\theta} \sum_{t=1}^{T} \nabla_\theta \ln \pi_\theta(a_t \mid s_t) \cdot R_t \approx \mathop{\mathbb{E}}_{\tau\sim\pi_\theta} \sum_{t=1}^{T} \nabla_\theta \ln \pi_\theta\big(a_t \mid s_t\big) \cdot R_t^\gamma \approx \frac{1}{N}\sum_{j=1}^{N}\sum_{t=1}^{T} \nabla_\theta \ln \pi_\theta\big(a_t(\tau_j) \mid s_t(\tau_j)\big) \cdot R_t^\gamma(\tau_j) \\
&= \nabla_\theta \frac{1}{N}\sum_{j=1}^{N}\sum_{t=1}^{T} \ln \pi_\theta\big(a_t(\tau_j) \mid s_t(\tau_j)\big) \cdot R_t^\gamma(\tau_j) =: \nabla_\theta J(\theta)
\end{aligned}$$

Training loop:

1. Sample N trajectories $\tau_1, \ldots, \tau_N$.
2. Calculate the function
$$-J(\theta) = -\frac{1}{N} \sum_{j=1}^{N} \sum_{t=1}^{T} \ln \pi_\theta\big(a_t(\tau_j) \mid s_t(\tau_j)\big) \cdot R_t^\gamma(\tau_j).$$
3. Ask PyTorch to minimize this function by performing gradient descent step.

2 Value Function

Policy gradient with baseline ($b : \mathbb{S} \to \mathbb{R}$):

$$\mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \sum_{t=1}^{T} \nabla_\theta \ln \pi_\theta(a_t \mid s_t) \cdot R_t = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \sum_{t=1}^{T} \nabla_\theta \ln \pi_\theta(a_t \mid s_t) \cdot \big(R_t - b(s_t)\big).$$

The value function ($V_{\pi_\theta} : \mathbb{S} \to \mathbb{R}$) approximation with value network (V_ψ):

$$V_\psi \approx V_{\pi_\theta}(s) := \mathop{\mathbb{E}}_{\tau \sim \pi_\theta, s_1(\tau)=s} R(\tau) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta, s_t(\tau)=s} R_t(\tau)$$

Advantage learning (value network as a baseline):

$$\mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \sum_{t=1}^{T} \nabla_\theta \ln \pi_\theta(a_t \mid s_t) \cdot R_t = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \sum_{t=1}^{T} \nabla_\theta \ln \pi_\theta(a_t \mid s_t) \cdot (R_t - V_\psi(s_t)) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \sum_{t=1}^{T} \nabla_\theta \ln \pi_\theta(a_t \mid s_t) \cdot A_t$$

Training loop:

1. Sample N trajectories $\tau_1, \ldots, \tau_N$.
2. Improve the parameters of our value network by minimizing the loss:
$$\frac{1}{M} \sum_{j=1}^{N} \sum_{t=1}^{T} \big(R_t(\tau_j) - V_\psi(s_t(\tau_j))\big)^2,$$
where M is the total number of samples in all trajectories.
3. Calculate the function
$$J(\theta) = \frac{1}{N} \sum_{j=1}^{N} \sum_{t=1}^{T} \ln \pi_\theta\big(a_t(\tau_j) \mid s_t(\tau_j)\big) \cdot [R_t(\tau_j) - V_\psi(s_t(\tau_j))],$$
4. Ask PyTorch to minimize the function $-J(\theta)$ by performing gradient descent step.

3 PPO

Let's define ρ_t as:
$$\rho_t := \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta\mathrm{OLD}}(a_t \mid s_t)}, \;\; \nabla_\theta \rho_t = \frac{\nabla_\theta \pi_\theta(a_t \mid s_t)}{\pi_{\theta\mathrm{OLD}}(a_t \mid s_t)}.$$
As long as for a given ϵ (e.g. $\epsilon = 0.2$):
$$1 - \epsilon < \rho_t < 1 + \epsilon,$$
we assume:

$$\pi_\theta(s_t) \approx \pi_{\theta\mathrm{OLD}}(s_t)$$
$$\mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \frac{\nabla_\theta \pi_\theta(a_t \mid s_t)}{\pi_\theta(a_t \mid s_t)} \approx \mathop{\mathbb{E}}_{\tau \sim \pi_{\theta\mathrm{OLD}}} \frac{\nabla_\theta \pi_\theta(a_t \mid s_t)}{\pi_{\theta\mathrm{OLD}}(a_t \mid s_t)}$$
$$\mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \sum_{t=1}^{T} \nabla_\theta \ln \pi_\theta(a_t \mid s_t) \cdot A_t \approx \mathop{\mathbb{E}}_{\tau \sim \pi_{\theta\mathrm{OLD}}} \sum_{t=1}^{T} \nabla_\theta \rho_t \cdot A_t$$

Surrogate objective:

$$\mathcal{J} := \mathop{\mathbb{E}}_{\tau \sim \pi_{\theta \text{OLD}}} \sum_{t=1}^{T} \rho_t A_t$$

$$\nabla_\theta \mathcal{J} = \nabla_\theta \mathop{\mathbb{E}}_{\tau \sim \pi_{\theta \text{OLD}}} \sum_{t=1}^{T} \rho_t \cdot A_t = \mathop{\mathbb{E}}_{\tau \sim \pi_{\theta \text{OLD}}} \sum_{t=1}^{T} \nabla_\theta \rho_t \cdot A_t$$

Clipped surrogate objective:

$$\tilde{\mathcal{J}} := \mathop{\mathbb{E}}_{\tau \sim \pi_{\theta \text{OLD}}} \sum_{t=1}^{T} \Big[\min \big(\rho_t \cdot A_t, clip(\rho_t, 1-\epsilon, 1+\epsilon) \cdot A_t \big) \Big]$$

$$\nabla_\theta \tilde{\mathcal{J}} = \mathop{\mathbb{E}}_{\tau \sim \pi_{\theta \text{OLD}}} \sum_{t=1}^{T} \nabla_\theta \Big[\min \big(\rho_t \cdot A_t, clip(\rho_t, 1-\epsilon, 1+\epsilon) \cdot A_t \big) \Big]$$

Training loop:

1. Sample N trajectories $\tau_1, \ldots, \tau_N$.
2. Improve the parameters of our value network by minimizing the loss:

$$\frac{1}{M} \sum_{j=1}^{T} \sum_{t=1}^{N} \Big[R_t(\tau_j) - V_\psi(s_t(\tau_j)) \Big]^2,$$

   where M is the total number of samples in all trajectories.
3. Calculate the function

$$\tilde{J}(\theta) = \frac{1}{M} \sum_{j=1}^{N} \sum_{t=1}^{T} \Big[\min \big(\rho_t \cdot A_t, clip(\rho_t, 1-\epsilon, 1+\epsilon) \cdot A_t \big) \Big],$$

   where M is the batch size.
4. Ask PyTorch to minimize the function $-\tilde{J}(\theta)$ by performing gradient descent step.