# Reinforcement Learning

Jungyoon Kim

January $18^{th}$, 2026

Seoul National University

# Overview

# Review

# Markov Decision Processes

**Definition**

A *Markov decision process* is a tuple $(S, A, P, R, \gamma)$ where $S$ is a set of states, $P : S \times A \times S \to [0, 1]$ is a state transition probability, $R : S \times A \to R$ is a reward function, and $\gamma \in (0, 1]$ is a discount factor.

We define a policy $\pi : S \times A \to [0, 1]$ (stochastic or deterministic) that assigns a probability of taking some action with respect to the current state. If the policy is deterministic, then we can treat $\pi : S \to A$.

# State and Action Value Functions

**Definition**

Given an MDP and policy $\pi$, its state-value function $v^\pi : S \to \mathbb{R}$ is defined as

$$v^\pi(s) := \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R^\pi(N_k^\pi(s))\right]$$

where $N_k(s)$ is the random variable describing its state after $k$ steps from $s$.

Moreover, its action-value function $q^\pi : S \times A \to \mathbb{R}$ is defined as

$$q^\pi(s,a) := R(s,a) + \gamma \sum_{s' \in S} \left(P(s,a,s')v^\pi(s')\right)$$

# Optimality

**Definition**

The optimal state/action-value function is defined as

$$v^\star(s) := \sup_{\pi \in \Pi} v^\pi(s), \quad \text{for all } s \in S$$

$$q^\star(s,a) := \sup_{\pi \in \Pi} q^\pi(s,a), \quad \text{for each } s \in S, a \in A.$$

where $\Pi$ is the space of all possible policies.

The optimal policy $\pi^\star$ is the maximizer of the state-value function.

$$v^{\pi_\star}(s) = v^\star(s), \quad \forall s \in S.$$

For discounted MDPs with bounded rewards and standard measurability assumptions, such optimal policy exists.

# Review: Bellman Optimality

**Theorem**

*Assume $\gamma \in (0, 1)$ and bounded rewards. Then the following statements hold.*

- *There exists an optimal policy $\pi^\star$.*
- *Every optimal policy achieves the optimal action-value function.*
- *The optimal state-value function is the maximizer (supremum) of the optimal action-value function.*

*Moreover, if its state and action space are finite, then the optimal policy $\pi^\star$ is deterministic.*

# Review: Bellman Optimality

**Theorem (Bellman Optimality Conditions)**

*For finite $S$, $A$ and $\gamma \in (0, 1)$,*

$$v^\star(s) = \max_{a \in A} \left( R(s, a) + \gamma \sum_{s' \in S} P(s, a, s') v^\star(s) \right)$$

$$q^\star(s, a) = R(s, a) + \gamma \sum_{s' \in S} P(s, a, s') \left( \max_{a' \in A} q^\star(s', a') \right).$$

*Moreover and greedy policy is optimal.*

$$\pi^\star(s) = \arg\max_{a \in A} \ q^\star(s, a)$$

# Preliminaries

# Policy Iteration

Policy iteration alternates

- **Policy evaluation:** estimate $v^{\pi_k}$ (approximately in practice).
- **Policy improvement:** make $\pi_{k+1}$ greedy w.r.t. $q^{\pi_k}$:

$$\pi_{k+1}(s) \in \arg\max_{a \in A} q^{\pi_k}(s, a).$$

For finite MDPs, PI provably converges. In reality this is impossible to implement, but the *idea* of policy evaluation and policy improvement are used to this day.

# Policy Gradient Methods

Instead of solving Bellman equations for each step, we optimizer the policy directly. We restrict to a parameterized family $\{\pi_\theta\}_{\theta \in \mathbb{R}^d}$ and define the performance objective

$$J(\theta) = \mathbb{E}_{s_0 \sim p_0}[v^{\pi_\theta}(s_0)]\,.$$

Our goal is to find the best possible policy $\pi$. Hence, we try to pick

$$\theta^\star = \arg\max_{\theta \in \mathbb{R}^d} J(\theta) = \arg\max_{\theta \in \mathbb{R}^d} \mathbb{E}_{s_0 \sim p_0}\left[v^{\pi_\theta}(s_0)\right]$$

*A simple idea*: perform gradient ascent

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t)$$

# REINFORCE

The REINFORCE algorithm uses the following unbiased estimator

$$\nabla J(\theta) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \left[ \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t \right]$$

where $G_t$ is the discounted reward achieved starting at time $t$, i.e.,

$$G_t := \sum_{k=t}^{T-1} \gamma^{k-t} r_k$$

*Intuition:* If an action leads to a high return $G_t$, then we increase its log-probability of taking that action.

In practice, we add a baseline (often a value function) to reduce variance without bias.

# Policy Improvement

# Problems with REINFORCE

- The unbiased estimate of REINFORCE has large variance, which leads to noisy gradients and instability.
- A single gradient step can radically change the distribution, which leads to collapse.
- Thus, in reality a very small stepsize $\alpha$ needs to be used to ensure stability. This hurts the sample efficiency.

# Trust Region Policy Optimization (ICML 2015)

*Motivation:* We want to take the *best possible step* in increasing $J(\theta)$, while simultaneously preventing collapse.

First define the *advantage*

$$A^{\pi}(s, a) := q^{\pi}(s, a) - v^{\pi}(s)$$

which represents the advantage of selecting a certain action, relative to the current value of the state.

# Trust Region Policy Optimization (ICML 2015)

We defined the 'score' of a policy as

$$J(\theta) = \mathop{\mathbb{E}}_{s_0 \sim p_0} [v^{\pi_\theta}(s_0)]$$

Then we can show, using simple reordering of terms,

$$J(\theta) - J(\theta_0) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \left[ \sum_{t=0}^{T-1} \gamma^t A^{\pi_{\theta_0}}(s_t, a_t) \right].$$

Then we use the following result. Given a state $s$,

$$\mathop{\mathbb{E}}_{a \sim \pi_\theta} [A^{\pi_{\theta_0}}(s, a)] = \mathop{\mathbb{E}}_{a \sim \pi_\theta, a' \sim \pi_{\theta_0}} \left[ \frac{\pi_\theta(a'|s)}{\pi_{\theta_0}(a|s)} A^{\pi_{\theta_0}}(s, a') \right].$$

where we denoted $\pi_\theta = \pi_\theta(\cdot|s)$, $\pi_{\theta_0} = \pi_{\theta_0}(\cdot|s)$ for convenience.

# Trust Region Policy Optimization (ICML 2015)

Hence,

$$J(\theta) - J(\theta_0) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta, a'_t \sim \pi_{\theta_0}(\cdot|s_t)} \left[ \sum_{t=0}^{T-1} \gamma^t \frac{\pi_\theta(a'_t|s_t)}{\pi_{\theta_0}(a'_t|s_t)} A^{\pi_{\theta_0}}(s_t, a'_t) \right]$$

We use a surrogate objective (which works when $\pi_\theta \approx \pi_{\theta_0}$)

$$\mathcal{L}(\theta, \theta_0) = \mathop{\mathbb{E}}_{\tau \sim \pi_{\theta_0}, a'_t \sim \pi_{\theta_0}(\cdot|s_t)} \left[ \sum_{t=0}^{T-1} \gamma^t \frac{\pi_\theta(a'_t|s_t)}{\pi_{\theta_0}(a'_t|s_t)} A^{\pi_{\theta_0}}(s_t, a'_t) \right] + C$$

and try to maximize this quantity instead. The two quantities have the same derivatives but not Hessians (which is concerning) - but $\mathcal{L}$ works relatively well empirically.

# Trust Region Policy Optimization (ICML 2015)

Empirically, we sample trajectories $\tau \sim \pi_{\theta_0}$ and take the average. However, this Monte-Carlo estimation only holds well when $\pi_\theta$ and $\pi_{\theta_0}$ are close.

Thus, from (1) the surrogate approximation and (2) importance sampling approximation, we need to keep $\pi_\theta \approx \pi_{\theta_0}$. This also fits into our goal - stepping too far from the initial distribution can cause collapse.

# Trust Region Policy Optimization (ICML 2015)

We keep the iterates close to each other in **distribution space**, i.e.,

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}(\theta_k, \theta)$$
$$\text{subject to } \bar{D}_{KL}(\theta \| \theta_k) \leq \delta$$

where the *surrogate objective* is

$$\mathcal{L}(\theta_k, \theta) = \mathop{\mathbb{E}}_{\tau \sim \pi_{\theta_k}} \left[ \sum_{s,a \sim \tau} \frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a) \right]$$

and the contraint is a bound on the expectation of the KL divergence (some references use the supremum, either works).

$$\bar{D}_{KL}(\theta \| \theta_k) := \mathop{\mathbb{E}}_{s \sim \pi_{\theta_k}} \left[ D_{KL}\left(\pi_\theta(\cdot|s) \| \pi_{\theta_k}(\cdot|s)\right) \right]$$

# Trust Region Policy Optimization (ICML 2015)

Implementation-wise, we make the Taylor approximations

$$\mathcal{L}(\theta_k, \theta) \approx g^\top (\theta - \theta_k)$$

$$\bar{D}_{KL}(\theta \| \theta_k) \approx \frac{1}{2}(\theta - \theta_k)^\top H (\theta - \theta_k)$$

which yields a quadratic-contrained linear problem. This can be solved using standard Lagrangian duality in convex optimization.

$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{g^\top H^{-1} g}} H^{-1} g$$

# c.f. Lagrangian Duality

Consider the standard optimization problem

$$\begin{aligned} \text{minimize} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0, \quad i = 1, \ldots, n \\ & g_j(x) = 0, \quad j = 1, \cdots m. \end{aligned}$$

Then its *Lagrangian* is defined as

$$\mathcal{L}(x, \mu, \nu) = f_0(x) + \sum_i \lambda_i f_i(x) + \sum_j \nu_j g_j(x)$$

and its dual objective is defined as

$$g(\lambda, \nu) = \inf_x \mathcal{L}(x, \mu, \nu).$$

# c.f. Langrangian Duality

The dual optimization problem becomes

$$\begin{aligned} \text{maximize} \quad & g(\lambda, \nu) \\ \text{subject to} \quad & \lambda \geq 0. \end{aligned}$$

Consider the primal optimal point $p^\star$ and the dual optimal point $d^\star$. For any optimization problem, weak duality holds, i.e.,

$$p^\star \geq d^\star$$

Strong duality ($p^\star = d^\star$) holds when the problem is convex and *Slater's condition* (a strictly feasible point exists) holds. In such case, we can solve the primal problem by considering its dual.

# Trust Region Policy Optimization (ICML 2015)

However, this iteration does not guarantee improvement nor guarantee a bound on the KL divergence. Hence we use a backtracking line search method

$$\theta_{k+1} = \theta_k + \alpha \sqrt{\frac{2\delta}{g^\top H^{-1} g}} H^{-1} g$$

for a stepsize $\alpha \in (0, 1]$ found using backtracking line search.

# Problems of TRPO

- Implementation is difficult due to trust-region formulation.
- Second-order methods are computationally expensive.
- Line search failures occur frequently and results are sensitive to hyperparameters such as $\delta$.

*Suggestion:* use a soft penalty loss, i.e., instead of the hard contraint, maximize

$$\mathop{\mathbb{E}}_{\tau\sim\pi_{\theta_k}}\left[\sum_{t=0}^{T-1}\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}A^{\pi_{\theta_k}}(s_t,a_t)-\beta\bar{D}_{KL}(\theta\|\theta_k)\right]$$

Empirically, choosing a good $\beta$ is difficult and often times yield suboptimal results.

# Proximal Policy Optimization (2017)

Let $r(\theta) := \pi_\theta(a|s)/\pi_{\theta_k}(a|s)$. The objective function of TRPO is

$$\mathcal{L}(\theta_k, \theta) = \underset{s,a\sim\pi_{\theta_k}}{\mathbb{E}} \left[ r(\theta) A^{\pi_{\theta_k}}(s,a) \right].$$

PPO replaces this objective with

$$\mathcal{L}^{\text{CLIP}}(\theta_k, \theta) := \underset{s,a\sim\pi_{\theta_k}}{\mathbb{E}} \left[ \min \left( r(\theta) A^{\pi_{\theta_k}}(s,a), g\left(\epsilon, A^{\pi_{\theta_k}}(s,a)\right) \right) \right]$$

where

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A, & \text{if } A \geq 0 \\ (1-\epsilon)A, & \text{if } A < 0 \end{cases}$$

*Intuition*: clips the ratio $r_t(\theta)$ if it is too far from 1 - we want to keep the ratio in $[1-\epsilon, 1+\epsilon]$ to prevent going too far.

# Proximal Policy Optimization (2017)

*Why clipped surrogate?* Let's look at a single $(s, a)$ pair.

- Let $A(s, a) > 0$. Then its contribution to the objective is

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

  Hence, if the action becomes more likely, then the objective increases - but only up to a ratio of $1 + \epsilon$.
- Let $A(s, a) < 0$. Then its contribution to the objective is

$$\mathcal{L}(s, a, \theta_k, \theta) = \max\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, (1-\epsilon)\right) A^{\pi_{\theta_k}}(s, a).$$

  Again, the objective increases when the action is less likely, but only up to a ratio of $1 - \epsilon$.
- Hence, the new policy does not benefit by going far from the old policy.

# Policy Evaluation

# Why Policy Evaluation?

In policy gradient methods, the update direction depends on the **advantage**

$$A^{\pi}(s_t, a_t) \;=\; q^{\pi}(s_t, a_t) - v^{\pi}(s_t).$$

Evaluating this function directly is impossible, hence we make an approximation using neural networks, parametrized by $\phi$.

Along a rollout $(s_t, a_t, r_t, s_{t+1})$, define the (learned) baseline $v_\phi(s)$ and build

$$\hat{A}_t = \hat{q}_t - v_\phi(s_t),$$

where $\hat{q}_t$ is a data-based approximation of $q^{\pi}(s_t, a_t)$.

# Monte-Carlo (MC) Methods

Monte-Carlo uses the full return as a sample of $Q^\pi(s_t, a_t)$:

$$G_t := \sum_{l=0}^{T-1-t} \gamma^l r_{t+l} \approx q^\pi(s_t, a_t).$$

The **MC advantage estimator** is

$$\hat{A}_t^{\text{MC}} := G_t - v_\phi(s_t) = \sum_{l=0}^{T-1-t} \gamma^l r_{t+l} - v_\phi(s_t).$$

# Monte-Carlo (MC) Methods

For actual actor-critic implementation, we learn the value function using gradient descent, i.e.,

$$\phi^\star = \arg\min_{\phi \in \mathbb{R}^d} \mathop{\mathbb{E}}_{s \sim p_0} \left[ \frac{1}{2} \left( v_\phi(s) - v^\pi(s) \right) \right] =: \arg\min_{\phi \in \mathbb{R}^d} L(\phi)$$

with some calculations, we can show that

$$\nabla_\phi L(\phi) = \mathop{\mathbb{E}}_{s_0 \sim p_0} \left[ \left( v_\phi(s_0) - \sum_{t=0}^{T-1} \gamma^t r_t \right) \nabla_\phi v_\phi(s_0) \right]$$

Hence we have an *unbiased* gradient.

# Temporal-Difference (TD) Methods

TD(0) replaces the long return with a one-step bootstrap:

$$q^{\pi}(s_t, a_t) \approx r_t + \gamma v^{\pi}(s_{t+1}) \approx r_t + \gamma v_{\phi}(s_{t+1}).$$

This yields the **TD advantage estimator**

$$\hat{A}_t^{\mathrm{TD}} := r_t + \gamma v_{\phi}(s_{t+1}) - v_{\phi}(s_t) =: \delta_t,$$

where $\delta_t$ is the **TD residual**.

# Temporal-Difference (TD) Methods

Similar to MC, we perform gradient descent, i.e., minimize $L(\phi)$. With some calculations we can show that

$$\nabla_\phi L(\phi) = \mathop{\mathbb{E}}_{s\sim p_0} \left[ \left( v_\phi(s) - r_0 - \gamma v^\pi(s_1) \right) \nabla_\phi v_\phi(s) \right]$$

However, this estimate is impossible to implement and hence we use a *biased* estimate

$$g := \left( v_\phi(s_0) - r_0 - \gamma v_\phi(s_1) \right) \nabla_\phi v_\phi(s_0)$$

# $k$-step TD

Interpolate between MC and TD by bootstrapping after $k$ steps:

$$\hat{q}_t^{(k)} := \sum_{l=0}^{k-1} \gamma^l r_{t+l} + \gamma^k v_\phi(s_{t+k}) \approx q^\pi(s_t, a_t).$$

The corresponding **$k$-step advantage estimator** is

$$\hat{A}_t^{(k)} := \hat{q}_t^{(k)} - v_\phi(s_t) = \sum_{l=0}^{k-1} \gamma^l r_{t+l} + \gamma^k v_\phi(s_{t+k}) - v_\phi(s_t).$$

Using telescoping, we acquire an equivalent expression

$$\hat{A}_t^{(k)} = \sum_{l=0}^{k-1} \gamma^l \delta_{t+l}, \quad \delta_t := r_t + \gamma v_\phi(s_{t+1}) - v_\phi(s_t).$$

$k = 1$ recovers TD; $k \to \infty$ approaches MC.

# GAE (Generalized Advantage Estimation)

The methods above are insufficient for PPO for the following reasons.

- **MC:** high variance due to long horizon
- **TD:** high bias due to biased gradient estimate

When performing PPO, $k$-step TD often yields suboptimal results. This is because tuning $k$ is difficult. Generalized Advantage Estimate (GAE) uses an exponentially weighted average of all $k$-step TD estimators.

# GAE (Generalized Advanatage Estimation)

Define

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} := \sum_{k=1}^{\infty} (1-\lambda)\lambda^{k-1}\, \hat{A}_t^{(k)}, \quad \lambda \in [0,1].$$

Using $\hat{A}_t^{(k)} = \sum_{l=0}^{k-1} \gamma^l \delta_{t+l}$, this simplifies to the standard form:

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l\, \delta_{t+l}.$$

# Implementations of GAE

In practice we compute GAE on a finite rollout segment $t = 0, \ldots, T-1$. Let

$$\delta_t := r_t + \gamma v_\phi(s_{t+1}) - v_\phi(s_t)$$

Then compute advantages backward.

$$\hat{A}_t = \delta_t + \gamma\lambda\hat{A}_{t+1}.$$

# Implementations of GAE

Similar to before, we want an estimate of the (stochastic) gradient to learn the value function. To do this, we choose

$$\phi^\star = \arg\min_{\phi \in \mathbb{R}^d} \mathbb{E}\left[\frac{1}{2}\left(v_\phi(s_t) - \hat{v}_t\right)^2\right]$$

where

$$\hat{v}_t = v_{\phi_{\text{old}}}(s_t) + \hat{A}_t^{\text{GAE}(\gamma,\lambda)}$$

Hence the stochastic gradient is

$$g = (v_\phi(s_t) - \hat{v}_t)\nabla_\phi v_\phi(s_t)$$

# Full PPO Algorithm

# 1. Rollout

Start with a policy $\pi_{\theta_k}(\cdot|s)$ and critic $v_{\phi_k}(s)$. Then rollout using this policy for time $T$ steps.

This gives us $(s_t, a_t, r_t, s_{t+1}, d_{t+1})$ for $t = 0, \ldots, T$. Then calculate a value estimate $v_{\phi_k}(s_t)$ for each step. $d_{t+1}$ indicates terminal states:

$$d_{t+1} = \begin{cases} 1 & \text{if } s_{t+1} \text{ is terminal} \\ 0 & \text{otherwise} \end{cases}$$

If $s_{t+1}$ is terminal, then $v_{t+1} = 0$.

# 2. GAE Calculation

Calculate the TD errors

$$\delta_t = r_t + \gamma(1 - d_{t+1})v_{\phi_k}(s_{t+1}) - v_{\phi_k}(s_t)$$

Now iterate backwards from $t = T - 1$ down to 0.

$$\hat{A}_t = \delta_t + \gamma\lambda(1 - d_{t+1})\hat{A}_{t+1}$$

This gives use the GAE estimate for the advantages of each step.

# 3. Updating the Parameters

Define the PPO probability ratio

$$r_t(\theta) := \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}.$$

Update the policy using the clipped objective:

$$\max_\theta \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right].$$

Define critic targets

$$\hat{v}_t := \hat{A}_t + v_{\phi_k}(s_t),$$

and update the critic by regression:

$$\min_\phi \mathbb{E}_t\left[\frac{1}{2}\left(v_\phi(s_t) - \hat{v}_t\right)^2\right].$$

via some gradient descent algorithm.

# References

# References


- Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press.
- Schulman, J., Levine, S., Moritz, P., Jordan, M. I., & Abbeel, P. (2015). Trust Region Policy Optimization. *ICML*.
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. arXiv:1707.06347.
- Achiam, J. (2018). *Spinning Up in Deep Reinforcement Learning*. OpenAI.
- Ryu, E. K. (2025). Lecture notes on reinforcement learning for large language models.