Reinforcement Learning for LLM Alignment (Draft)

Contents

1 Important Concepts

1.1 Probability model and basic notation

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space on which all random variables are defined. We write $x \sim p$ to mean that x is sampled from a distribution p.

- $\mathcal{S}$: state space.
- $\mathcal{A}$: action space.
- $\Delta(\mathcal{X})$: set of probability measures on a measurable space $\mathcal{X}$.

Throughout this book, the reward is a deterministic function

$$r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}. \tag{1}$$

We will later define an MDP formally as $(\mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma)$, where

- $\rho_0 \in \Delta(\mathcal{S})$ is the initial-state distribution,
- $P(\cdot \mid s, a) \in \Delta(\mathcal{S})$ is the transition kernel,
- $\gamma \in [0, 1)$ is the discount factor,
- $\pi(\cdot \mid s) \in \Delta(\mathcal{A})$ is a policy.

Example 1.1 (Discrete expectation as a weighted average). *Let $\mathcal{X} = \{1, 2, 3\}$ and $p(1) = 0.2$, $p(2) = 0.5$, $p(3) = 0.3$. If $f(x) = x^2$, then*

$$\mathbb{E}_{x\sim p}[f(x)] = 0.2 \cdot 1^2 + 0.5 \cdot 2^2 + 0.3 \cdot 3^2 = 0.2 + 2.0 + 2.7 = 4.9.$$

This "sum of values weighted by probabilities" view will reappear in RL as weighted averages over states/actions.

Example 1.2 (Continuous expectation). *If $x \sim \mathcal{N}(0, 1)$ and $f(x) = x^2$, then $\mathbb{E}[f(x)] = \mathrm{Var}(x) + (\mathbb{E}[x])^2 = 1$. Here the expectation is an integral $\int x^2 \, \phi(x) \, dx$, but the meaning is the same: average of $f(x)$ under $p(x)$.*

1.2 Monte-Carlo estimation

Let $x \sim p$ and let f be p-integrable. Define

$$\mu := \mathbb{E}_{x\sim p}[f(x)]. \tag{2}$$

Given i.i.d. samples $x^{(1)}, \dots, x^{(N)} \overset{i.i.d.}{\sim} p$, the Monte-Carlo estimator is

$$\widehat{\mu}_{\mathrm{MC}} := \frac{1}{N} \sum_{i=1}^{N} f\left(x^{(i)}\right), \qquad x^{(i)} \sim p. \tag{3}$$

Remark 1.1 (Why Monte-Carlo works). *Under mild conditions, $\widehat{\mu}_{\mathrm{MC}} \to \mu$ as $N \to \infty$ (law of large numbers). Moreover, the typical estimation error scales like $O(1/\sqrt{N})$ (central limit theorem intuition).*

Example 1.3 (Estimating a probability). *Let $x \sim \mathrm{Bernoulli}(p)$, so $x \in \{0, 1\}$ and $\mathbb{E}[x] = p$. With samples $x^{(1:N)}$, the estimator*

$$\widehat{p} = \frac{1}{N} \sum_{i=1}^{N} x^{(i)}$$

is exactly Monte-Carlo with $f(x) = x$.

Example 1.4 (Monte-Carlo in RL: estimating return). *Suppose a policy rollout produces rewards (r_0, r_1, r_2) and then terminates, with $\gamma = 0.9$. The (sample) return is*

$$\widehat{G}_0 = r_0 + 0.9 \, r_1 + 0.9^2 r_2.$$

If we repeat rollouts $\tau^{(1)}, \dots, \tau^{(N)}$ and compute $\widehat{G}_0^{(i)}$ for each, then

$$\widehat{J}_{\mathrm{MC}}(\pi) := \frac{1}{N} \sum_{i=1}^{N} \widehat{G}_0^{(i)}$$

is a Monte-Carlo estimator of $J(\pi) = \mathbb{E}_{\tau \sim p_\pi}[G_0(\tau)]$.

1.3 Importance Sampling (IS)

Let p be a target distribution and q a proposal distribution such that $p \ll q$ (absolute continuity). Then for integrable f,

$$\mathbb{E}_{x\sim p}[f(x)] = \mathbb{E}_{x\sim q}\left[\frac{p(x)}{q(x)} f(x)\right]. \tag{4}$$

Define the importance weight $w(x) := \frac{p(x)}{q(x)}$. Given i.i.d. samples $x^{(1)}, \dots, x^{(N)} \overset{i.i.d.}{\sim} q$,

$$\widehat{\mu}_{\text{IS}} := \frac{1}{N}\sum_{i=1}^{N} w\left(x^{(i)}\right) f\left(x^{(i)}\right), \qquad x^{(i)} \sim q. \tag{5}$$

Remark 1.2 (Support mismatch is fatal)**.** *If there exists x with $p(x) > 0$ but $q(x) = 0$, then $w(x)$ is undefined/infinite and IS cannot correct the mismatch. In RL language: you cannot evaluate a new policy on actions that the behavior policy never takes.*

Example 1.5 (A tiny discrete IS calculation)**.** *Let $\mathcal{X} = \{a, b\}$. Target $p(a) = 0.8$, $p(b) = 0.2$; proposal $q(a) = 0.5$, $q(b) = 0.5$. Let $f(a) = 1$, $f(b) = 3$. Then $\mu = \mathbb{E}_p[f] = 0.8 \cdot 1 + 0.2 \cdot 3 = 1.4$. If we sample from q, IS uses weights*

$$w(a) = \frac{0.8}{0.5} = 1.6, \qquad w(b) = \frac{0.2}{0.5} = 0.4.$$

So $w(x)f(x)$ equals 1.6 when $x = a$ and 1.2 when $x = b$. Averaging these over samples from q recovers 1.4 in expectation.

Example 1.6 (Off-policy evaluation in a contextual bandit (single-step RL))**.** *A context $s \sim d(s)$ is drawn, then one action a is chosen, reward $r(s, a)$ is observed, and the episode ends. Let the* target *policy be $\pi'(a \mid s)$ and the* behavior *policy be $\pi(a \mid s)$. The target value is*

$$J(\pi') = \mathbb{E}_{s\sim d}\mathbb{E}_{a\sim\pi'(\cdot|s)}[r(s, a)].$$

If data is collected under π, then action-wise IS gives

$$J(\pi') = \mathbb{E}_{s\sim d}\mathbb{E}_{a\sim\pi(\cdot|s)}\left[\frac{\pi'(a \mid s)}{\pi(a \mid s)} r(s, a)\right].$$

With samples $(s^{(i)}, a^{(i)}, r^{(i)})$ from π, an unbiased estimator is

$$\widehat{J}_{\text{IS}}(\pi') = \frac{1}{N}\sum_{i=1}^{N} \frac{\pi'(a^{(i)} \mid s^{(i)})}{\pi(a^{(i)} \mid s^{(i)})} r^{(i)}.$$

This is the basic "off-policy correction" idea used throughout modern policy optimization.

Remark 1.3 (Practical trick: self-normalized IS)**.** *When $w(x)$ has high variance, a common stabilization is the self-normalized estimator*

$$\widehat{\mu}_{\text{SNIS}} = \frac{\sum_{i=1}^{N} w(x^{(i)}) f(x^{(i)})}{\sum_{i=1}^{N} w(x^{(i)})}.$$

It is biased in finite samples but often lower-variance in practice. Clipping or capping weights is another common variance-control trick (also biased).

1.4 Log-derivative trick (score-function identity)

This is a general mathematical identity independent of reinforcement learning.

Let $p_\theta(x)$ be a probability density (or mass function) parameterized by $\theta \in \mathbb{R}^d$. Let f be integrable under p_θ. Define

$$F(\theta) := \mathbb{E}_{x\sim p_\theta}[f(x)]. \tag{6}$$

Assume we may interchange differentiation and integration. Then

$$\nabla_\theta F(\theta) = \mathbb{E}_{x\sim p_\theta}\big[f(x)\, \nabla_\theta \log p_\theta(x)\big]. \tag{7}$$

Example 1.7 (Bernoulli example: see the gradient explicitly)**.** *Let $x \in \{0,1\}$ with $p_\theta(x) = \theta^x(1-\theta)^{1-x}$ for $\theta \in (0,1)$. Then*

$$\log p_\theta(x) = x\log\theta + (1-x)\log(1-\theta), \qquad \frac{\partial}{\partial\theta}\log p_\theta(x) = \frac{x}{\theta} - \frac{1-x}{1-\theta}.$$

For any $f(x)$,

$$\frac{d}{d\theta}\mathbb{E}[f(x)] = \mathbb{E}\left[f(x)\left(\frac{x}{\theta} - \frac{1-x}{1-\theta}\right)\right].$$

This is the simplest "REINFORCE-style" gradient form.

Example 1.8 (Softmax policy: the score is simple)**.** *Let a discrete policy be*

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top\varphi(s,a))}{\sum_{a'}\exp(\theta^\top\varphi(s,a'))},$$

where $\varphi(s,a) \in \mathbb{R}^d$ are features. Then

$$\nabla_\theta \log \pi_\theta(a \mid s) = \varphi(s,a) - \mathbb{E}_{a'\sim\pi_\theta(\cdot|s)}[\varphi(s,a')].$$

So policy-gradient methods often reduce to "(chosen features) minus (expected features)" weighted by an advantage signal.

Remark 1.4 (One more trick: baselines are free)**.** *If b does not depend on the sampled variable (e.g. $b = b(s)$ in RL), then*

$$\mathbb{E}_{x\sim p_\theta}\big[b\,\nabla_\theta \log p_\theta(x)\big] = b\,\nabla_\theta \int p_\theta(x)\,dx = 0.$$

Therefore you can replace $f(x)$ by $f(x) - b$ inside the expectation without changing the gradient, often reducing variance. (We will use this constantly in policy gradients.)

1.5 Temporal-Difference (TD) idea

Let $V : \mathcal{S} \to \mathbb{R}$ be a value-function approximator. Consider one transition generated by some policy π and kernel P:

$$s_t \sim d_t^\pi, \qquad a_t \sim \pi(\cdot \mid s_t), \qquad s_{t+1} \sim P(\cdot \mid s_t, a_t). \tag{8}$$

Define the TD(0) target and TD error:

$$y_t := r(s_t,a_t) + \gamma V(s_{t+1}), \qquad \delta_t := y_t - V(s_t) = r(s_t,a_t) + \gamma V(s_{t+1}) - V(s_t). \tag{9}$$

Example 1.9 (Two-state TD update by hand). *Let $\mathcal{S} = \{A, B\}$ and suppose the observed transition is $s_t = A \to s_{t+1} = B$ with reward $r = 1$ and $\gamma = 0.9$. Assume current estimates $V(A) = 0.2$ and $V(B) = 0.7$. Then*

$$y_t = 1 + 0.9 \cdot 0.7 = 1.63, \qquad \delta_t = 1.63 - 0.2 = 1.43.$$

A TD(0) update with stepsize η is

$$V(A) \leftarrow V(A) + \eta\, \delta_t.$$

For example, if $\eta = 0.1$, then $V(A) \leftarrow 0.2 + 0.1 \cdot 1.43 = 0.343$.

Remark 1.5 (Why TD is powerful). *Monte-Carlo targets use full returns, which can have high variance. TD uses* bootstrapping *($V(s_{t+1})$) to reduce variance and to learn from partial trajectories. The trade-off is bias: if $V(s_{t+1})$ is wrong, the TD target is biased. Many RL methods (e.g. actor–critic, GAE) interpolate between MC and TD to control this bias–variance trade-off.*

2 Reinforcement Learning Fundamentals

2.1 MDP problem setup

An infinite-horizon discounted Markov Decision Process (MDP) is a tuple

$$\mathcal{M} := (\mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma), \tag{10}$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\rho_0 \in \Delta(\mathcal{S})$ is the initial-state distribution, $P(\cdot \mid s, a) \in \Delta(\mathcal{S})$ is the transition kernel, $\gamma \in [0, 1)$ is the discount factor, and

$$r : \mathcal{S} \times \mathcal{A} \to \mathbb{R} \tag{11}$$

is the reward function (book-wide convention).

Policy. A (stationary Markov) policy is a conditional distribution over actions given states:

$$\pi(\cdot \mid s) \in \Delta(\mathcal{A}) \quad \text{for each } s \in \mathcal{S}. \tag{12}$$

We use lowercase letters to denote sampled random variables, e.g.

$$a \sim \pi(\cdot \mid s). \tag{13}$$

Trajectory and trajectory distribution. Given (ρ_0, P, π), define the stochastic process $(s_t, a_t)_{t \geq 0}$ by

$$s_0 \sim \rho_0, \qquad a_t \sim \pi(\cdot \mid s_t), \qquad s_{t+1} \sim P(\cdot \mid s_t, a_t). \tag{14}$$

A trajectory is

$$\tau := (s_0, a_0, s_1, a_1, s_2, a_2, \dots) \in (\mathcal{S} \times \mathcal{A})^{\mathbb{N}}. \tag{15}$$

We denote by p_π the probability measure on trajectories induced by the generative process above. Expectations over rollouts always specify this distribution, e.g. $\mathbb{E}_{\tau \sim p_\pi}[\cdot]$.

Return and optimization objective. Define the discounted reward-to-go from time t:

$$G_t(\tau) := \sum_{k=0}^{\infty} \gamma^k \, r(s_{t+k}, a_{t+k}). \tag{16}$$

Define the performance objective:

$$J(\pi) := \mathbb{E}_{\tau\sim p_\pi}\big[G_0(\tau)\big] = \mathbb{E}_{\tau\sim p_\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]. \tag{17}$$

The reinforcement learning problem is to find an optimal policy

$$\pi^\star \in \arg\max_{\pi} J(\pi). \tag{18}$$

2.2 Important constructions

Discounted occupancy measures. Define the discounted state-occupancy distribution

$$d_\pi(s) := (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \, \mathbb{P}(s_t = s \mid \pi), \tag{19}$$

and the discounted state-action occupancy distribution

$$d_\pi(s,a) := (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \, \mathbb{P}(s_t = s, a_t = a \mid \pi) = d_\pi(s)\, \pi(a \mid s). \tag{20}$$

When we write $(s,a) \sim d_\pi$, we mean the pair is sampled from this discounted occupancy distribution.

2.3 Value functions

2.3.1 Definitions

The state-value function is

$$V^\pi(s) := \mathbb{E}[G_t \mid s_t = s], \tag{21}$$

and the action-value function is

$$Q^\pi(s,a) := \mathbb{E}[G_t \mid s_t = s,\ a_t = a]. \tag{22}$$

The advantage function is

$$A^\pi(s,a) := Q^\pi(s,a) - V^\pi(s). \tag{23}$$

All conditional expectations above are taken over future randomness induced by the same transition kernel P and policy π.

2.3.2 Connection between V^π and Q^π

(1) V^π is the policy expectation of Q^π. For every $s \in \mathcal{S}$,

$$V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\big[Q^\pi(s,a)\big]. \tag{24}$$

(2) Q^π is one-step reward plus discounted next-state value. For every $(s,a) \in \mathcal{S} \times \mathcal{A}$,

$$Q^\pi(s,a) = r(s,a) + \gamma\, \mathbb{E}_{s' \sim P(\cdot|s,a)}\big[V^\pi(s')\big]. \tag{25}$$

Bellman expectation equation for V^π. Combining the two identities yields:

$$V^\pi(s) = \mathbb{E}_{\substack{a \sim \pi(\cdot|s) \\ s' \sim P(\cdot|s,a)}}\big[r(s,a) + \gamma V^\pi(s')\big]. \tag{26}$$

Bellman expectation equation for Q^π. Substituting $V^\pi(s') = \mathbb{E}_{a' \sim \pi(\cdot|s')}[Q^\pi(s',a')]$ into the Q^π recursion gives:

$$Q^\pi(s,a) = r(s,a) + \gamma\, \mathbb{E}_{\substack{s' \sim P(\cdot|s,a) \\ a' \sim \pi(\cdot|s')}}\big[Q^\pi(s',a')\big]. \tag{27}$$

2.3.3 Advantage function properties

(A) Advantage is centered under the policy. For every $s \in \mathcal{S}$,

$$\mathbb{E}_{a \sim \pi(\cdot|s)}\big[A^\pi(s,a)\big] = 0. \tag{28}$$

Reason: $\mathbb{E}_{a\sim\pi(\cdot|s)}[A^\pi(s,a)] = \mathbb{E}_{a\sim\pi(\cdot|s)}[Q^\pi(s,a)] - V^\pi(s) = V^\pi(s) - V^\pi(s) = 0$.

(B) Advantage has zero mean under discounted occupancy. Sampling $(s,a) \sim d_\pi(s,a)$,

$$\mathbb{E}_{(s,a)\sim d_\pi}\big[A^\pi(s,a)\big] = 0. \tag{29}$$

Reason: expand as $\mathbb{E}_{s\sim d_\pi}\mathbb{E}_{a\sim\pi(\cdot|s)}[A^\pi(s,a)]$ and apply (A).

(C) One-step form of the advantage. Using $Q^\pi(s,a) = r(s,a) + \gamma\, \mathbb{E}_{s'\sim P(\cdot|s,a)}[V^\pi(s')]$,

$$A^\pi(s,a) = r(s,a) + \gamma\, \mathbb{E}_{s'\sim P(\cdot|s,a)}[V^\pi(s')] - V^\pi(s). \tag{30}$$

This identity is frequently used to build practical estimators by replacing V^π with an approximator and the expectation over s' with samples.

3 Policy Optimization

We consider a parameterized policy $\pi_\theta(a \mid s)$ with parameters $\theta \in \mathbb{R}^d$.

3.1 Objective

Define

$$J(\theta) := J(\pi_\theta) = \mathbb{E}_{\tau\sim p_{\pi_\theta}}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t,a_t)\right]. \tag{31}$$

The goal of policy optimization is to maximize $J(\theta)$ over θ.

3.2 Policy gradient theorem

Apply the log-derivative trick (Section 1.4) with $x = \tau$ and $p_\theta(\cdot) = p_{\pi_\theta}(\cdot)$:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_{\pi_\theta}}[G_0(\tau)\, \nabla_\theta \log p_{\pi_\theta}(\tau)]\,. \tag{32}$$

Only the policy terms depend on θ, hence

$$\nabla_\theta \log p_{\pi_\theta}(\tau) = \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t). \tag{33}$$

Therefore,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_{\pi_\theta}}\left[G_0(\tau) \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]. \tag{34}$$

A standard variance-reduced equivalent form uses reward-to-go:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_{\pi_\theta}}\left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, G_t(\tau)\right]. \tag{35}$$

Using the discounted occupancy distribution, one may write

$$\nabla_\theta J(\theta) = \frac{1}{1-\gamma}\, \mathbb{E}_{(s,a) \sim d_{\pi_\theta}}\big[\nabla_\theta \log \pi_\theta(a \mid s)\, Q^{\pi_\theta}(s,a)\big]. \tag{36}$$

3.3 Basic policy optimization algorithm (Vanilla policy gradient)

At iteration k with parameters θ_k:

1. **Sampling step:** Collect N trajectories

$$\tau^{(1)}, \ldots, \tau^{(N)} \overset{i.i.d.}{\sim} p_{\pi_{\theta_k}}. \tag{37}$$

2. **Return computation:** For each trajectory i and time t, define (finite-horizon) reward-to-go

$$\widehat{G}_t^{(i)} := \sum_{j=t}^{T_i - 1} \gamma^{j-t}\, r\left(s_j^{(i)}, a_j^{(i)}\right). \tag{38}$$

3. **Gradient estimator (distribution: $\tau^{(i)} \sim p_{\pi_{\theta_k}}$):**

$$\widehat{\nabla_\theta J}(\theta_k) := \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T_i - 1} \nabla_\theta \log \pi_{\theta_k}\left(a_t^{(i)} \mid s_t^{(i)}\right)\, \widehat{G}_t^{(i)}. \tag{39}$$

4. **Update:**

$$\theta_{k+1} := \theta_k + \alpha\, \widehat{\nabla_\theta J}(\theta_k). \tag{40}$$

3.4 Improvements for gradient stability

3.4.1 Baseline (control variate)

Let $b : \mathcal{S} \to \mathbb{R}$ be any function. For each fixed $s \in \mathcal{S}$,

$$\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\big[\nabla_\theta \log \pi_\theta(a \mid s)\, b(s)\big] = 0. \tag{41}$$

Hence, subtracting $b(s_t)$ inside the policy-gradient estimator does not change the expected gradient under $\tau \sim p_{\pi_\theta}$:

$$\mathbb{E}_{\tau\sim p_{\pi_\theta}}\left[\sum_{t\ge 0} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, G_t(\tau)\right] = \mathbb{E}_{\tau\sim p_{\pi_\theta}}\left[\sum_{t\ge 0} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, (G_t(\tau) - b(s_t))\right]. \tag{42}$$

A canonical choice is $b(s) = V^{\pi_\theta}(s)$, which yields an advantage-style estimator.

3.4.2 A2C (Advantage Actor-Critic)

A2C maintains:

- an actor $\pi_\theta(a \mid s)$,
- a critic $V_\phi(s) \approx V^{\pi_\theta}(s)$.

Critic update (TD). From on-policy samples

$$s_t \sim d_t^{\pi_\theta}, \qquad a_t \sim \pi_\theta(\cdot \mid s_t), \qquad s_{t+1} \sim P(\cdot \mid s_t, a_t), \tag{43}$$

define TD error

$$\delta_t(\phi) := r(s_t, a_t) + \gamma V_\phi(s_{t+1}) - V_\phi(s_t). \tag{44}$$

The critic parameters ϕ are typically updated to reduce (an empirical approximation of)

$$\mathbb{E}\big[\delta_t(\phi)^2\big], \tag{45}$$

where the expectation is under the on-policy transition distribution above.

Actor update (advantage estimate). Define an advantage estimator, for example

$$\widehat{A}_t := \widehat{G}_t - V_\phi(s_t), \tag{46}$$

where $\widehat{G}_t$ is computed from samples generated under $\tau \sim p_{\pi_\theta}$. Then a common actor gradient estimate is

$$\widehat{g} := \widehat{\mathbb{E}}\Big[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\, \widehat{A}_t\Big], \tag{47}$$

where $\widehat{\mathbb{E}}$ denotes an empirical average over collected on-policy samples.

3.4.3 A3C (Asynchronous Advantage Actor-Critic)

A3C uses the same mathematical form as actor-critic with an advantage baseline, but changes the data-collection and update protocol:

- multiple workers interact with (copies of) the environment in parallel,
- each worker produces on-policy samples and computes gradients,
- shared parameters are updated asynchronously.

Each worker's empirical estimates still correspond to expectations under its own on-policy sampling distribution; asynchrony affects the optimization dynamics, not the underlying estimator definitions.

4 Bounded Policy Optimization

Motivation. Policy-gradient methods optimize $J(\pi)$ using samples generated by the current policy. A large policy update can sharply change the trajectory/state distribution, making previously collected data, value/advantage estimates, and importance weights unreliable. Bounded policy optimization makes this issue explicit:

- start from an *exact* expression for $J(\pi') - J(\pi)$,
- introduce a *local surrogate* objective that is estimable from data generated by π,
- expose the resulting *distribution shift penalty* and control it by conservative updates.

4.1 Why naive policy optimization can fail: two toy pathologies

This section introduces bounded ("conservative") policy optimization. Before we build the formal machinery, we give two fully explicit examples showing why unconstrained policy updates can be unreliable.

Throughout, fix an infinite-horizon discounted MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma)$ with $\gamma \in [0, 1)$ and deterministic reward $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. For any stationary policy π, define the performance

$$J(\pi) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right],$$

where $s_0 \sim \rho_0$, $a_t \sim \pi(\cdot \mid s_t)$, $s_{t+1} \sim P(\cdot \mid s_t, a_t)$. Define the value functions

$$V^\pi(s) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \,\middle|\, s_0 = s\right], \qquad Q^\pi(s, a) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \,\middle|\, s_0 = s,\ a_0 = a\right],$$

and the advantage $A^\pi(s, a) := Q^\pi(s, a) - V^\pi(s)$. Define also the normalized discounted state-occupancy measure

$$d_\pi(s) := (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \ \Pr(s_t = s \mid \pi), \qquad \sum_s d_\pi(s) = 1.$$

4.1.1 Example 1: A local surrogate predicts improvement, but true performance collapses (distribution shift)

MDP definition. Let $\mathcal{S} = \{s_0, s_1, s_\perp\}$ where $s_\perp$ is absorbing. Let $\rho_0(s_0) = 1$. Let the action sets be $\mathcal{A}(s_0) = \{a_0, a_1\}$, $\mathcal{A}(s_1) = \{g, b\}$, and $\mathcal{A}(s_\perp) = \{\perp\}$. Transitions are deterministic:

$$P(s_\perp \mid s_0, a_0) = 1, \qquad P(s_1 \mid s_0, a_1) = 1, \qquad P(s_\perp \mid s_1, g) = P(s_\perp \mid s_1, b) = 1, \qquad P(s_\perp \mid s_\perp, \perp) = 1.$$

Rewards are

$$r(s_0, a_0) = r(s_0, a_1) = 0, \qquad r(s_1, g) = 1, \qquad r(s_1, b) = -M, \qquad r(s_\perp, \perp) = 0,$$

for some constant $M > 0$.

Two policies. Fix parameters $\eta \in (0, 1)$ and $\eta_2 \in (0, 1)$. Define an "old" policy π by

$$\pi(a_1 \mid s_0) = \eta,\ \pi(a_0 \mid s_0) = 1 - \eta, \qquad \pi(b \mid s_1) = \eta_2,\ \pi(g \mid s_1) = 1 - \eta_2.$$

Define a "candidate new" policy π' by

$$\pi'(a_1 \mid s_0) = 1,\ \pi'(a_0 \mid s_0) = 0, \qquad \pi'(b \mid s_1) = 1,\ \pi'(g \mid s_1) = 0.$$

Note that π' is absolutely continuous w.r.t. π action-wise (needed for ratios): $\pi(a_1 \mid s_0) = \eta > 0$ and $\pi(b \mid s_1) = \eta_2 > 0$.

Compute $J(\pi)$ and $J(\pi')$ exactly. First compute the value at s_1 under π:

$$V^\pi(s_1) = \mathbb{E}_{a\sim\pi(\cdot|s_1)}[r(s_1, a)] = (1 - \eta_2) \cdot 1 + \eta_2 \cdot (-M) = 1 - \eta_2(1 + M).$$

From s_0, choosing a_0 terminates immediately with reward 0, while choosing a_1 moves to s_1 and then terminates. Thus

$$J(\pi) = V^\pi(s_0) = \eta \cdot \gamma V^\pi(s_1) = \eta\,\gamma\big(1 - \eta_2(1 + M)\big).$$

Under π', we deterministically take a_1 then b, so

$$J(\pi') = \gamma \cdot (-M) = -\gamma M.$$

Compute the advantages A^π needed for bounded optimization. At s_1,

$$A^\pi(s_1, b) = Q^\pi(s_1, b) - V^\pi(s_1) = (-M) - \big(1 - \eta_2(1 + M)\big) = -(1 + M)(1 - \eta_2).$$

At s_0,

$$Q^\pi(s_0, a_1) = \gamma V^\pi(s_1), \qquad V^\pi(s_0) = \eta\gamma V^\pi(s_1),$$

so

$$A^\pi(s_0, a_1) = Q^\pi(s_0, a_1) - V^\pi(s_0) = (1 - \eta)\gamma V^\pi(s_1) = (1 - \eta)\gamma\big(1 - \eta_2(1 + M)\big).$$

Compute the discounted occupancies under π and π'. Because s_0 is visited only at $t = 0$, we have $d_\pi(s_0) = d_{\pi'}(s_0) = (1 - \gamma)$. State s_1 is visited at $t = 1$ with probability η under π and with probability 1 under π'. Hence

$$d_\pi(s_1) = (1 - \gamma)\gamma\eta, \qquad d_{\pi'}(s_1) = (1 - \gamma)\gamma.$$

A standard local surrogate that *ignores* state-distribution shift. Define the (action-wise importance-weighted) local surrogate around π:

$$L_\pi(\pi') := J(\pi) + \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\pi,\ a\sim\pi(\cdot|s)}\left[\frac{\pi'(a \mid s)}{\pi(a \mid s)}A^\pi(s,a)\right].$$

In this example, the ratio collapses the action expectation at each visited state:

$$\mathbb{E}_{a\sim\pi(\cdot|s_0)}\left[\frac{\pi'(a \mid s_0)}{\pi(a \mid s_0)}A^\pi(s_0,a)\right] = A^\pi(s_0,a_1), \qquad \mathbb{E}_{a\sim\pi(\cdot|s_1)}\left[\frac{\pi'(a \mid s_1)}{\pi(a \mid s_1)}A^\pi(s_1,a)\right] = A^\pi(s_1,b),$$

so

$$\begin{aligned} L_\pi(\pi') - J(\pi) &= \frac{1}{1-\gamma}\Big(d_\pi(s_0)A^\pi(s_0,a_1) + d_\pi(s_1)A^\pi(s_1,b)\Big) \\ &= \gamma\Big((1-\eta)\big(1-\eta_2(1+M)\big) - \eta(1+M)(1-\eta_2)\Big). \end{aligned}$$

Concrete numerical instantiation. Take $\gamma = 0.9$, $M = 100$, $\eta_2 = 10^{-3}$, $\eta = 10^{-4}$. Then $V^\pi(s_1) = 1 - \eta_2(1+M) = 1 - 0.001 \cdot 101 = 0.899$, and therefore

$$J(\pi) = \eta\gamma V^\pi(s_1) = 10^{-4}\cdot 0.9 \cdot 0.899 = 8.091\times 10^{-5}, \qquad J(\pi') = -\gamma M = -90.$$

The surrogate "improvement" equals

$$L_\pi(\pi') - J(\pi) = \gamma\Big((1-\eta)V^\pi(s_1) - \eta(1+M)(1-\eta_2)\Big) \approx 0.9\cdot(0.8989101 - 0.0100899) \approx 0.79994 > 0.$$

Thus the local surrogate predicts a *large positive* improvement, while the true performance drops from 8.091×10^{-5} to -90.

What went wrong (formal diagnosis). The exact performance difference identity weights $A^\pi(s,a)$ by the *new* occupancy $d_{\pi'}$:

$$J(\pi') - J(\pi) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{\pi'},\ a\sim\pi'(\cdot|s)}[A^\pi(s,a)].$$

In our MDP, the harmful term $A^\pi(s_1,b) = -(1+M)(1-\eta_2)$ is multiplied by $d_{\pi'}(s_1) = (1-\gamma)\gamma$, whereas the surrogate uses only $d_\pi(s_1) = (1-\gamma)\gamma\eta$. Since $\eta \ll 1$, the surrogate almost *ignores* the bad behavior in s_1 even though π' visits s_1 with high probability. This is precisely the **state-distribution shift** problem that bounded policy optimization controls.

4.1.2 Example 2: Trajectory importance sampling has variance that grows exponentially with horizon

MDP definition (single-state, H-step episode embedded in an infinite-horizon MDP). Let $\mathcal{S} = \{s, s_\perp\}$ with $\rho_0(s) = 1$. For $t = 0, 1, \dots, H-1$, the state is s; after H actions the process deterministically moves to the absorbing $s_\perp$. Let $\mathcal{A}(s) = \{0,1\}$ and $\mathcal{A}(s_\perp) = \{\perp\}$. Let the only nonzero reward be granted at the final transition: $r(s_\perp,\perp) = 1$ and all other rewards are 0. Hence, for any policy, the return is the *deterministic constant* $G_0 = \gamma^H \cdot 1$.

Off-policy evaluation by trajectory importance sampling. Suppose we have data $\tau^{(1)}, \dots, \tau^{(N)}$ generated i.i.d. by a behavior policy π. For a target policy π', define the trajectory likelihood ratio

$$w(\tau) := \prod_{t=0}^{H-1} \frac{\pi'(a_t \mid s)}{\pi(a_t \mid s)}.$$

The standard IS estimator of $J(\pi')$ is

$$\widehat{J}_{\text{IS}}(\pi') := \frac{1}{N} \sum_{i=1}^{N} w(\tau^{(i)})\, G_0.$$

It is unbiased because $\mathbb{E}_{\tau \sim p_\pi}[w(\tau)] = 1$ (assuming action-wise absolute continuity).

Closed-form variance and exponential blow-up. Because $G_0 = \gamma^H$ is constant here,

$$\widehat{\text{Var}}\left(\widehat{J}_{\text{IS}}(\pi')\right) = \frac{\gamma^{2H}}{N}\, \widehat{\text{Var}}_\pi(w(\tau)).$$

Now take Bernoulli policies $\pi(1 \mid s) = p$ and $\pi'(1 \mid s) = q$ with $p, q \in (0, 1)$. At each step the ratio is

$$\frac{\pi'(a \mid s)}{\pi(a \mid s)} = \begin{cases} \frac{q}{p}, & a = 1, \\ \frac{1-q}{1-p}, & a = 0, \end{cases}$$

and steps are i.i.d. under π, so

$$\mathbb{E}_\pi[w(\tau)^2] = \left(\mathbb{E}_{a \sim \pi(\cdot|s)} \left[\left(\frac{\pi'(a \mid s)}{\pi(a \mid s)} \right)^2 \right] \right)^H = \left(\frac{q^2}{p} + \frac{(1-q)^2}{1-p} \right)^H.$$

Therefore,

$$\widehat{\text{Var}}_\pi(w(\tau)) = \mathbb{E}_\pi[w(\tau)^2] - 1 = \left(\frac{q^2}{p} + \frac{(1-q)^2}{1-p} \right)^H - 1,$$

which grows exponentially in H whenever $p \neq q$ (since then the base exceeds 1).

Numerical illustration. Let $p = 0.5$, $q = 0.9$. Then

$$\frac{q^2}{p} + \frac{(1-q)^2}{1-p} = \frac{0.81}{0.5} + \frac{0.01}{0.5} = 1.62 + 0.02 = 1.64,$$

so $\widehat{\text{Var}}_\pi(w(\tau)) = 1.64^H - 1$. Even for moderate H, the IS weights become extremely high-variance, making reliable improvement from reweighted old data difficult.

Connection to bounded updates. Example 4.1.1 shows that ignoring the state-distribution shift can make a "local" objective misleading. Example 4.1.2 shows that trying to *correct* for distribution shift by exact trajectory importance sampling introduces variance that can explode with horizon. Bounded policy optimization methods (CPI/TRPO/PPO-style) can be understood as principled compromises: they seek policy improvements using surrogates that are estimable from on-policy (or nearly on-policy) data, while explicitly controlling how far the new policy can move from the old one.

4.2 Performance Difference Lemma

Let π and π' be two stationary policies. Recall the (normalized) discounted state occupancy

$$d_\pi(s) := (1-\gamma)\sum_{t=0}^{\infty}\gamma^t\,\mathbb{P}(s_t = s \mid \pi), \qquad \sum_s d_\pi(s) = 1.$$

Lemma (Performance Difference). For any π, π',

$$J(\pi') - J(\pi) = \mathbb{E}_{\tau\sim p_{\pi'}}\left[\sum_{t=0}^{\infty}\gamma^t A^\pi(s_t, a_t)\right] = \frac{1}{1-\gamma}\,\mathbb{E}_{s\sim d_{\pi'},\ a\sim\pi'(\cdot|s)}\big[A^\pi(s,a)\big]. \tag{48}$$

Comment. Equation (48) is exact, but it depends on the unknown new-policy occupancy $d_{\pi'}$, which is the core difficulty we will address next.

4.3 Local surrogate objective and distribution shift penalty

Start from (48) and expand the expectation over states:

$$J(\pi') = J(\pi) + \frac{1}{1-\gamma}\sum_s d_{\pi'}(s)\,\mathbb{E}_{a\sim\pi'(\cdot|s)}\big[A^\pi(s,a)\big]. \tag{49}$$

Now add and subtract the same term with $d_\pi(s)$, and then apply action-wise importance sampling (to rewrite expectations under $\pi'(\cdot \mid s)$ using actions sampled from $\pi(\cdot \mid s)$):

$$\begin{aligned}
J(\pi') &= J(\pi) + \frac{1}{1-\gamma}\sum_s d_\pi(s)\,\mathbb{E}_{a\sim\pi'(\cdot|s)}\big[A^\pi(s,a)\big] \\
&\quad + \frac{1}{1-\gamma}\sum_s \big(d_{\pi'}(s) - d_\pi(s)\big)\,\mathbb{E}_{a\sim\pi'(\cdot|s)}\big[A^\pi(s,a)\big] \\
&= J(\pi) + \frac{1}{1-\gamma}\sum_s d_\pi(s)\,\mathbb{E}_{a\sim\pi(\cdot|s)}\Big[\frac{\pi'(a\mid s)}{\pi(a\mid s)}A^\pi(s,a)\Big] \\
&\quad + \frac{1}{1-\gamma}\sum_s \big(d_{\pi'}(s) - d_\pi(s)\big)\,\mathbb{E}_{a\sim\pi(\cdot|s)}\Big[\frac{\pi'(a\mid s)}{\pi(a\mid s)}A^\pi(s,a)\Big].
\end{aligned} \tag{50}$$

We name the two parts in (50).

Local surrogate objective.

$$L_\pi(\pi') := J(\pi) + \frac{1}{1-\gamma}\,\mathbb{E}_{s\sim d_\pi,\ a\sim\pi(\cdot|s)}\Big[\frac{\pi'(a\mid s)}{\pi(a\mid s)}A^\pi(s,a)\Big]. \tag{51}$$

Distribution shift penalty.

$$\mathrm{DSP}_\pi(\pi') := \frac{1}{1-\gamma}\sum_s \big(d_{\pi'}(s) - d_\pi(s)\big)\,\mathbb{E}_{a\sim\pi(\cdot|s)}\Big[\frac{\pi'(a\mid s)}{\pi(a\mid s)}A^\pi(s,a)\Big]. \tag{52}$$

Thus, the exact identity is

$$J(\pi') = L_\pi(\pi') + \mathrm{DSP}_\pi(\pi'). \tag{53}$$

Why $\mathrm{DSP}_\pi(\pi') \to 0$ when policies are close. The penalty depends on the occupancy difference $d_{\pi'} - d_\pi$. Intuitively, if π' is close to π at every state, then the induced Markov chain changes only slightly, so $d_{\pi'}$ is close to d_π. Formally, one can bound

$$|\mathrm{DSP}_\pi(\pi')| \le \frac{1}{1-\gamma}\, \|d_{\pi'} - d_\pi\|_1 \cdot \max_s \left|\mathbb{E}_{a\sim\pi'(\cdot|s)}[A^\pi(s,a)]\right|,$$

and $\|d_{\pi'} - d_\pi\|_1 \to 0$ as $\pi' \to \pi$ (pointwise in s), hence $\mathrm{DSP}_\pi(\pi') \to 0$.

Importance sampling viewpoint (action-wise). The ratio $\pi'(a \mid s)/\pi(a \mid s)$ is exactly the importance weight that changes the action distribution from $\pi(\cdot \mid s)$ to $\pi'(\cdot \mid s)$ while keeping the state distribution d_π fixed. This is what makes $L_\pi(\pi')$ estimable from trajectories generated by π.

4.4 Conservative Policy Update

Assume we have a current policy π and an "improved" policy

$$\pi' \in \arg\max_{\tilde{\pi}} L_\pi(\tilde{\pi}). \tag{54}$$

Instead of jumping directly to π', we update conservatively using a mixture policy:

$$\pi_\alpha(\cdot \mid s) := (1-\alpha)\,\pi(\cdot \mid s) + \alpha\,\pi'(\cdot \mid s), \qquad \alpha \in [0,1]. \tag{55}$$

Properties of the mixture policy π_α.

- **Validity.** For each s, $\pi_\alpha(\cdot \mid s)$ is a convex combination of distributions, hence a distribution.
- **Controlled change.** For each s,

$$D_{\mathrm{TV}}\big(\pi(\cdot \mid s), \pi_\alpha(\cdot \mid s)\big) = \alpha\, D_{\mathrm{TV}}\big(\pi(\cdot \mid s), \pi'(\cdot \mid s)\big) \le \alpha.$$

- **Importance-weight form vs. π.** Whenever $\pi(a \mid s) > 0$,

$$\frac{\pi_\alpha(a \mid s)}{\pi(a \mid s)} = (1-\alpha) + \alpha\,\frac{\pi'(a \mid s)}{\pi(a \mid s)}. \tag{56}$$

Surrogate objective for π_α is linear in α. Using $\mathbb{E}_{a\sim\pi(\cdot|s)}[A^\pi(s,a)] = 0$,

$$\begin{aligned} L_\pi(\pi_\alpha) &= J(\pi) + \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\pi,\ a\sim\pi(\cdot|s)}\Big[\frac{\pi_\alpha(a \mid s)}{\pi(a \mid s)}A^\pi(s,a)\Big] \\ &= J(\pi) + \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\pi}\mathbb{E}_{a\sim\pi(\cdot|s)}\Big[\Big((1-\alpha) + \alpha\frac{\pi'(a \mid s)}{\pi(a \mid s)}\Big)A^\pi(s,a)\Big] \\ &= J(\pi) + \alpha \cdot \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\pi,\ a\sim\pi'(\cdot|s)}\big[A^\pi(s,a)\big]. \end{aligned} \tag{57}$$

Define the (surrogate) improvement term and a uniform statewise advantage magnitude:

$$g := \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\pi,\ a\sim\pi'(\cdot|s)}\big[A^\pi(s,a)\big] = L_\pi(\pi') - J(\pi), \qquad \epsilon := \max_s \left|\mathbb{E}_{a\sim\pi'(\cdot|s)}\big[A^\pi(s,a)\big]\right|. \tag{58}$$

Then (57) becomes $L_\pi(\pi_\alpha) = J(\pi) + \alpha g$.

A conservative lower bound for $J(\pi_\alpha)$ (CPI bound). For mixture updates (55), one can bound the distribution shift penalty (using that both the occupancy change and the expected advantage under π_α scale with α), obtaining the quadratic lower bound

$$J(\pi_\alpha) \;\geq\; L_\pi(\pi_\alpha) \;-\; \frac{2\gamma}{(1-\gamma)^2}\,\epsilon\,\alpha^2. \tag{59}$$

Combining with $L_\pi(\pi_\alpha) = J(\pi) + \alpha g$ yields

$$J(\pi_\alpha) - J(\pi) \;\geq\; \alpha\, g \;-\; \frac{2\gamma}{(1-\gamma)^2}\,\epsilon\,\alpha^2. \tag{60}$$

Choosing α. The right-hand side of (60) is a concave quadratic in α. Maximizing it over $\alpha \in [0,1]$ gives

$$\alpha^\star = \min\left\{1,\; \frac{(1-\gamma)^2}{4\gamma} \cdot \frac{g}{\epsilon}\right\}. \tag{61}$$

This choice guarantees a nontrivial lower bound on improvement whenever $g > 0$.

4.5 Conservative Policy Iteration

We now turn the previous derivations into an algorithmic template.

Conservative Policy Iteration (CPI) (template). Initialize π_0. For $k = 0, 1, 2, \ldots$ repeat:

1. **Collect on-policy data.** Sample trajectories $\tau \sim p_{\pi_k}$ and build an advantage estimator $\widehat{A}^{\pi_k}(s,a)$ (e.g. via a critic and GAE / reward-to-go baseline).

2. **Surrogate improvement step (importance-sampled).** Since data is generated by π_k, optimize an importance-weighted objective:

$$\pi'_k \approx \arg\max_{\tilde{\pi}} \mathbb{E}_{s\sim d_{\pi_k},\; a\sim\pi_k(\cdot|s)}\Big[\frac{\tilde{\pi}(a \mid s)}{\pi_k(a \mid s)}\, \widehat{A}^{\pi_k}(s,a)\Big]. \tag{62}$$

   (The additive constant $J(\pi_k)$ and factor $1/(1-\gamma)$ are omitted since they do not affect the maximizer.)

3. **Estimate bound parameters.** Estimate

$$\widehat{g}_k := \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{\pi_k},\; a\sim\pi'_k(\cdot|s)}\big[\widehat{A}^{\pi_k}(s,a)\big], \qquad \widehat{\epsilon}_k := \max_s \Big|\mathbb{E}_{a\sim\pi'_k(\cdot|s)}\big[\widehat{A}^{\pi_k}(s,a)\big]\Big|,$$

   using empirical averages (and, for $\max_s$, typically a conservative approximation from batch data).

4. **Choose conservative step size.**

$$\alpha_k := \min\left\{1,\; \frac{(1-\gamma)^2}{4\gamma} \cdot \frac{\widehat{g}_k}{\widehat{\epsilon}_k}\right\}. \tag{63}$$

5. **Conservative mixture update.**

$$\pi_{k+1}(\cdot \mid s) := (1-\alpha_k)\,\pi_k(\cdot \mid s) + \alpha_k\,\pi'_k(\cdot \mid s). \tag{64}$$

Interpretation. Step 2 tries to improve the estimable surrogate $L_{\pi_k}(\cdot)$ using data from π_k. Step 5 keeps the update conservative so that the distribution shift penalty stays small, and the quadratic bound (60) ensures the update is safe for sufficiently small α_k.

4.6 Trust Region Policy Optimization (TRPO)

Transition from CPI to trust regions. In the previous subsections we obtained an exact decomposition

$$J(\pi') = L_\pi(\pi') + \mathrm{DSP}_\pi(\pi'),$$

where $L_\pi(\pi')$ is estimable from data generated by π (via action-wise importance sampling), and $\mathrm{DSP}_\pi(\pi')$ captures the error caused by the state-distribution shift $d_{\pi'} - d_\pi$. CPI controls this shift by making conservative mixture updates $\pi_\alpha = (1-\alpha)\pi + \alpha\pi'$.

TRPO follows the same core idea—*maximize the local surrogate while keeping the policy update small*—but instead of mixing policies with a scalar α, it directly constrains the update size using a **trust region** measured by KL divergence between the old and new policies.

4.6.1 Parameterized policies and the trust-region problem

Let $\pi_\theta(a \mid s)$ be a differentiable, parameterized policy and let θ_{old} denote the parameters of the behavior policy that generated the current batch of trajectories. Following our earlier surrogate definition (51), we define the (parameterized) surrogate objective

$$L_{\theta_{\text{old}}}(\theta) := J(\pi_{\theta_{\text{old}}}) + \frac{1}{1-\gamma}\, \mathbb{E}_{s\sim d_{\pi_{\theta_{\text{old}}}},\, a\sim\pi_{\theta_{\text{old}}}(\cdot|s)}\left[\frac{\pi_\theta(a \mid s)}{\pi_{\theta_{\text{old}}}(a \mid s)}\, A^{\pi_{\theta_{\text{old}}}}(s,a)\right]. \tag{65}$$

As usual, the additive constant $J(\pi_{\theta_{\text{old}}})$ does not affect the maximizer, but we keep it to match the book-wide definition of $L_\pi(\cdot)$.

To keep the distribution shift penalty small, TRPO constrains how far the new policy moves away from the old one, using a KL trust region. A theoretically motivated (but impractical) version constrains the *maximum* per-state KL:

$$\max_s\ \mathrm{KL}\Big(\pi_{\theta_{\text{old}}}(\cdot \mid s)\ \|\ \pi_\theta(\cdot \mid s)\Big) \le \delta. \tag{66}$$

Since (66) is difficult to enforce directly (it is effectively a large collection of constraints), TRPO uses a practical approximation based on the *average* KL under the old-policy state distribution:

$$\mathbb{E}_{s\sim d_{\pi_{\theta_{\text{old}}}}}\Big[\mathrm{KL}\big(\pi_{\theta_{\text{old}}}(\cdot \mid s)\ \|\ \pi_\theta(\cdot \mid s)\big)\Big] \le \delta. \tag{67}$$

Thus TRPO updates θ by approximately solving:

$$\begin{aligned} \max_\theta \quad & L_{\theta_{\text{old}}}(\theta) \\ \text{s.t.} \quad & \mathbb{E}_{s\sim d_{\pi_{\theta_{\text{old}}}}}\Big[\mathrm{KL}\big(\pi_{\theta_{\text{old}}}(\cdot \mid s)\ \|\ \pi_\theta(\cdot \mid s)\big)\Big] \le \delta. \end{aligned} \tag{68}$$

This is the defining optimization problem of TRPO.

4.6.2 Sample-based objective and constraint

In practice, we have a batch of samples (s_t, a_t) generated by $\pi_{\theta_{\text{old}}}$ and an advantage estimator $\widehat{A}_t \approx A^{\pi_{\theta_{\text{old}}}}(s_t, a_t)$. Using empirical averages over timesteps in the batch, we optimize the empirical surrogate (dropping constants):

$$\widehat{L}(\theta) := \widehat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \, \widehat{A}_t \right], \tag{69}$$

subject to the empirical average-KL constraint

$$\widehat{D}_{\text{KL}}(\theta_{\text{old}}, \theta) := \widehat{\mathbb{E}}_t \left[\text{KL}\big(\pi_{\theta_{\text{old}}}(\cdot \mid s_t) \,\|\, \pi_\theta(\cdot \mid s_t)\big) \right] \le \delta. \tag{70}$$

This keeps the update within a trust region around $\pi_{\theta_{\text{old}}}$ and is designed to prevent large distribution shifts.

4.6.3 Efficient approximate solution (natural-gradient style step)

The constrained problem (68) is solved approximately via: (i) a linear approximation to the surrogate and (ii) a quadratic approximation to the KL constraint. Let

$$g := \nabla_\theta \widehat{L}(\theta)\big|_{\theta=\theta_{\text{old}}}. \tag{71}$$

A second-order expansion of (70) around θ_{old} gives

$$\widehat{D}_{\text{KL}}(\theta_{\text{old}}, \theta) \approx \frac{1}{2}(\theta - \theta_{\text{old}})^\top A (\theta - \theta_{\text{old}}), \qquad A := \nabla^2_\theta \widehat{D}_{\text{KL}}(\theta_{\text{old}}, \theta)\big|_{\theta=\theta_{\text{old}}}. \tag{72}$$

The matrix A is the (batch-averaged) Fisher information matrix associated with the policy class (as induced by KL).

With these approximations, the update direction s is obtained by (approximately) solving the linear system

$$As = g, \tag{73}$$

which is done efficiently using conjugate gradient and Fisher-vector products (so we never form A explicitly).

Given a direction s, choose the largest step length that satisfies the quadratic constraint:

$$\beta := \sqrt{\frac{2\delta}{s^\top A s}}, \qquad \theta_{\text{cand}} := \theta_{\text{old}} + \beta s. \tag{74}$$

Finally, because both the surrogate and the KL are nonlinear in θ, TRPO performs a backtracking line search along s to ensure (a) improvement in $\widehat{L}$ and (b) satisfaction of the KL constraint.

4.6.4 Algorithm (TRPO)

TRPO (one iteration). Given current parameters θ_{old}:

1. **Collect on-policy data.** Roll out $\pi_{\theta_{\text{old}}}$ to collect a batch of (s_t, a_t, r_t) and compute an advantage estimator $\widehat{A}_t$ for each timestep (e.g. via a critic and GAE / reward-to-go baseline).

2. **Form empirical surrogate and constraint.** Use (69) and (70).

3. **Compute search direction.** Compute $g = \nabla_\theta \widehat{L}(\theta)\big|_{\theta=\theta_{\text{old}}}$ and use conjugate gradient to approximately solve $As = g$, where A is the Hessian of $\widehat{D}_{\text{KL}}(\theta_{\text{old}}, \theta)$ at $\theta = \theta_{\text{old}}$ (implemented via Fisher-vector products).

4. **Scale step to match trust region.** Set $\beta = \sqrt{2\delta/(s^\top As)}$ and propose $\theta_{\text{cand}} = \theta_{\text{old}} + \beta s$.

5. **Backtracking line search.** Starting from θ_{cand}, repeatedly shrink the step (e.g. $\theta = \theta_{\text{old}} + \xi\beta s$ with $\xi \in (0,1)$) until both conditions hold:

$$\widehat{L}(\theta) \geq \widehat{L}(\theta_{\text{old}}), \qquad \widehat{D}_{\text{KL}}(\theta_{\text{old}}, \theta) \leq \delta.$$

   Accept the first θ that satisfies them.

6. **Update.** Set $\theta \leftarrow \theta_{\text{new}}$ (the accepted parameters), and repeat.

Summary. TRPO directly instantiates the bounded-policy-optimization principle developed earlier: it maximizes a local surrogate objective that is estimable from $\pi_{\theta_{\text{old}}}$ data, while enforcing a KL trust region so that the update remains conservative and the distribution shift penalty stays small.

4.7 Proximal Policy Optimization (PPO)

Transition from trust regions to a first-order "proximal" objective. From bounded policy optimization we obtained the local surrogate

$$L_\pi(\pi') = J(\pi) + \frac{1}{1-\gamma}\, \mathbb{E}_{s\sim d_\pi,\ a\sim\pi(\cdot|s)}\Big[\frac{\pi'(a \mid s)}{\pi(a \mid s)} A^\pi(s,a)\Big],$$

together with the fact that large updates make the distribution shift penalty non-negligible. TRPO enforces "small updates" by a KL trust-region constraint. PPO keeps the same surrogate structure but replaces the explicit constrained solve with a *first-order* objective that discourages ratios far from 1, enabling multiple epochs of minibatch optimization on the same on-policy batch.

4.7.1 The bounded-policy-optimization objective in parameter space

Let $\pi_\theta(a \mid s)$ be a differentiable policy class and let θ_{old} be the parameters that generated the current batch of data (the behavior policy). Dropping the additive constant $J(\pi_{\theta_{\text{old}}})$, the optimization target suggested by the bounded policy optimization surrogate is:

$$\theta_\star \in \arg\max_\theta\ \mathbb{E}_{s\sim d_{\pi_{\theta_{\text{old}}}},\ a\sim\pi_{\theta_{\text{old}}}(\cdot|s)}\left[\frac{\pi_\theta(a \mid s)}{\pi_{\theta_{\text{old}}}(a \mid s)}\ A^{\pi_{\theta_{\text{old}}}}(s,a)\right]. \tag{75}$$

Finite-batch estimator. Assume we collected a finite batch of m timesteps

$$\mathcal{D} = \{(s_i, a_i, r_i, s_i')\}_{i=1}^m \quad \text{by rolling out} \quad a_i \sim \pi_{\theta_{\text{old}}}(\cdot \mid s_i),$$

and computed advantage estimates $\widehat{A}_i \approx A^{\pi_{\theta_{\text{old}}}}(s_i, a_i)$. Define the probability ratio

$$r_i(\theta) := \frac{\pi_\theta(a_i \mid s_i)}{\pi_{\theta_{\text{old}}}(a_i \mid s_i)}. \tag{76}$$

Then the empirical surrogate corresponding to (75) is

$$\widehat{L}_{\mathrm{BPO}}(\theta) := \frac{1}{m}\sum_{i=1}^{m} r_i(\theta)\,\widehat{A}_i. \tag{77}$$

Maximizing (77) with many gradient steps can push ratios $r_i(\theta)$ far from 1, which empirically leads to destructively large updates; PPO modifies the objective to prevent that.

4.7.2 Clipped PPO objective (policy loss)

Fix $\varepsilon_{\mathrm{clip}} \in (0,1)$ and define the clipping operator

$$\mathrm{clip}(x, 1-\varepsilon_{\mathrm{clip}}, 1+\varepsilon_{\mathrm{clip}}) := \min\{1+\varepsilon_{\mathrm{clip}},\ \max\{x,\ 1-\varepsilon_{\mathrm{clip}}\}\}.$$

Define the clipped per-sample surrogate term

$$\ell_i^{\mathrm{CLIP}}(\theta) := \min\Big(r_i(\theta)\,\widehat{A}_i,\ \mathrm{clip}(r_i(\theta), 1-\varepsilon_{\mathrm{clip}}, 1+\varepsilon_{\mathrm{clip}})\,\widehat{A}_i\Big), \tag{78}$$

and the empirical clipped policy objective

$$\widehat{L}_{\mathrm{CLIP}}(\theta) := \frac{1}{m}\sum_{i=1}^{m} \ell_i^{\mathrm{CLIP}}(\theta). \tag{79}$$

The "min" construction makes $\widehat{L}_{\mathrm{CLIP}}(\theta)$ a pessimistic (lower-bound style) modification of the basic surrogate (77): it removes incentive to improve the objective by pushing $r_i(\theta)$ outside $[1-\varepsilon_{\mathrm{clip}}, 1+\varepsilon_{\mathrm{clip}}]$.

4.7.3 Full PPO optimization objective (actor–critic form)

In PPO practice, advantages are computed using a learned value function $V_\phi(s)$, and learning is done by optimizing a combined objective for policy and value parameters.

Advantage and value targets (one standard choice). Define TD residuals (on the collected batch)

$$\delta_i(\phi) := r_i + \gamma V_\phi(s_i') - V_\phi(s_i).$$

For a trajectory segment, define a truncated generalized advantage estimator (GAE-style):

$$\widehat{A}_i := \sum_{j=0}^{L_i-1} (\gamma\lambda)^j\,\delta_{i+j}(\phi_{\mathrm{old}}), \qquad \lambda \in [0,1], \tag{80}$$

where L_i is the remaining length of the segment starting at index i (so the sum stays within the segment). A common compatible value target is

$$V_i^{\mathrm{targ}} := \widehat{A}_i + V_{\phi_{\mathrm{old}}}(s_i). \tag{81}$$

Value-function loss and entropy bonus. Define the squared-error value loss per sample

$$\ell_i^{\mathrm{VF}}(\phi) := \big(V_\phi(s_i) - V_i^{\mathrm{targ}}\big)^2, \tag{82}$$

and define an entropy bonus (Shannon entropy for discrete actions; differential entropy for continuous actions)

$$\mathcal{H}\big(\pi_\theta(\cdot \mid s)\big) := -\sum_{a\in\mathcal{A}} \pi_\theta(a \mid s)\log \pi_\theta(a \mid s) \quad (\text{discrete } \mathcal{A}). \tag{83}$$

Combined PPO objective (maximization form). Let $c_1, c_2 \geq 0$ be coefficients. PPO (actor–critic style) performs approximate maximization of

$$(\theta_{\text{new}}, \phi_{\text{new}}) \in \arg\max_{\theta,\phi} \ \widehat{L}_{\text{CLIP+VF+S}}(\theta, \phi), \tag{84}$$

where

$$\widehat{L}_{\text{CLIP+VF+S}}(\theta, \phi) := \frac{1}{m} \sum_{i=1}^{m} \Big(\ell_i^{\text{CLIP}}(\theta) - c_1 \, \ell_i^{\text{VF}}(\phi) + c_2 \, \mathcal{H}\big(\pi_\theta(\cdot \mid s_i)\big) \Big). \tag{85}$$

(Equivalently: one may *minimize* the negative of (85).)

4.7.4 Algorithm (PPO, parameterized by θ)

PPO (one iteration, mathematically explicit). Fix hyperparameters: clipping $\varepsilon_{\text{clip}}$, discount γ, GAE λ, coefficients c_1, c_2, number of epochs K, minibatch size M ($M \leq m$), and an optimizer/stepsize.

Let $(\theta_{\text{old}}, \phi_{\text{old}})$ be current parameters.

1. **Sampling.** Roll out $\pi_{\theta_{\text{old}}}$ to obtain a batch $\mathcal{D} = \{(s_i, a_i, r_i, s_i')\}_{i=1}^{m}$.
2. **Compute targets from ϕ_{old}.** Compute $\widehat{A}_i$ using (80) (with ϕ_{old} fixed inside it) and compute V_i^{targ} via (81).
3. **Optimization (approximate argmax by SGD).** Initialize $(\theta, \phi) \leftarrow (\theta_{\text{old}}, \phi_{\text{old}})$. For epoch $e = 1, \dots, K$:
   - Partition (or sample) $\mathcal{D}$ into minibatches $B \subset \{1, \dots, m\}$ of size M.
   - For each minibatch B, compute the minibatch objective

     $$\widehat{L}_B(\theta, \phi) := \frac{1}{|B|} \sum_{i \in B} \Big(\ell_i^{\text{CLIP}}(\theta) - c_1 \, \ell_i^{\text{VF}}(\phi) + c_2 \, \mathcal{H}(\pi_\theta(\cdot \mid s_i)) \Big),$$

     where ℓ_i^{CLIP} uses ratios $r_i(\theta)$ from (76).
   - Update (θ, ϕ) by one step of stochastic gradient ascent on $\widehat{L}_B(\theta, \phi)$.
4. **Commit the update.** Set $(\theta_{\text{old}}, \phi_{\text{old}}) \leftarrow (\theta, \phi)$ and proceed to the next iteration.

Optional: KL-penalty proximal objective. Instead of clipping, one may optimize the KL-penalized surrogate

$$\widehat{L}_{\text{KLPEN}}(\theta) := \frac{1}{m} \sum_{i=1}^{m} \Big(r_i(\theta) \widehat{A}_i - \beta \ \text{KL}(\pi_{\theta_{\text{old}}}(\cdot \mid s_i) \parallel \pi_\theta(\cdot \mid s_i)) \Big),$$

possibly with adaptive updates of β to keep the realized average KL near a target.

5 Effective Policy Optimization methods

Previously we introduced on-policy policy-gradient methods (e.g. PPO-style bounded updates) that work well in practice. However, in large language model (LLM) alignment, a direct actor–critic implementation is often expensive: the critic V_ϕ is typically a neural network of comparable scale to the policy π_θ, which (for LLMs) means training *two* large sequence models simultaneously.

This chapter focuses on *critic-free* (or critic-light) policy-optimization methods that are widely used in modern LLM alignment pipelines. The main idea is to replace learned value baselines with *group-based Monte-Carlo baselines* and to stabilize updates via *bounded* or *softly gated* importance weighting.

5.1 RL task setup for LLM alignment

Tokens as actions, prefixes as states. Fix a vocabulary $\mathcal{V}$ and a prompt (query) space $\mathcal{X}$. A parameterized autoregressive LLM policy π_θ defines a distribution over token sequences conditioned on a prompt:

$$\pi_\theta(y \mid x) = \prod_{t=1}^{T(y)} \pi_\theta\big(y_t \mid x, y_{<t}\big), \qquad y = (y_1, \dots, y_{T(y)}) \in \mathcal{V}^{T(y)}. \tag{86}$$

We view generation as an episodic MDP:

- **State space:** $\mathcal{S} := \bigcup_{t \geq 0} \big(\mathcal{X} \times \mathcal{V}^t\big)$. A state is the current prompt-prefix pair $s_t := (x, y_{<t})$.
- **Action space:** $\mathcal{A} := \mathcal{V} \cup \{\texttt{EOS}\}$. The action is the next token $a_t := y_t \sim \pi_\theta(\cdot \mid s_t)$.
- **Transition:** deterministic concatenation $s_{t+1} = (x, y_{\leq t})$, until EOS or a max length.

Reward model and KL-regularized objective. Let ρ be a distribution over prompts $x \sim \rho$ (e.g. a dataset). Let $R(x, y) \in \mathbb{R}$ be a scalar sequence-level score (often produced by a reward model or preference model). A standard alignment objective penalizes deviation from a fixed *reference* policy π_{ref} (usually the SFT model) via a KL term:

$$J_\beta(\theta) := \mathbb{E}_{x \sim \rho} \mathbb{E}_{y \sim \pi_\theta(\cdot \mid x)} \left[R(x, y) - \beta \sum_{t=1}^{T(y)} \log \frac{\pi_\theta(y_t \mid x, y_{<t})}{\pi_{\text{ref}}(y_t \mid x, y_{<t})} \right], \qquad \beta > 0. \tag{87}$$

The log-ratio term in (87) is a Monte-Carlo estimator of the per-state KL divergence (under actions sampled from π_θ), so maximizing J_β encourages high reward while keeping π_θ close to π_{ref}.

On-policy iteration with a frozen behavior policy. As in PPO-style methods, each update iteration uses a frozen *behavior* policy $\pi_{\theta_{\text{old}}}$ to sample data, and then updates θ using importance ratios that correct from $\pi_{\theta_{\text{old}}}$ to π_θ.

5.2 Group-based Monte-Carlo baselines (critic-free advantages)

A key practical trick in LLM RL is that rewards are often computed at the *sequence* level, and learning a high-quality token-level critic can be costly and unstable. Group-based methods replace the critic baseline with statistics computed from multiple samples per prompt.

Fix a group size $G \in \mathbb{N}$. For a prompt x, sample a *group* of responses from the behavior policy:

$$y^{(1)}, \dots, y^{(G)} \overset{i.i.d.}{\sim} \pi_{\theta_{\text{old}}}(\cdot \mid x), \qquad y^{(i)} = (y_1^{(i)}, \dots, y_{T_i}^{(i)}), \quad T_i := T(y^{(i)}). \tag{88}$$

Compute scalar scores (possibly already KL-shaped)

$$u^{(i)} := u(x, y^{(i)}) \in \mathbb{R} \qquad \text{(e.g. } u(x,y) = R(x,y) \text{ or } R(x,y) - \beta\,\mathrm{KL} \text{ penalty).} \tag{89}$$

Define the group mean and standard deviation

$$\bar{u} := \frac{1}{G}\sum_{j=1}^{G} u^{(j)}, \qquad s_u := \sqrt{\frac{1}{G}\sum_{j=1}^{G}\big(u^{(j)} - \bar{u}\big)^2 + \varepsilon}, \quad \varepsilon > 0, \tag{90}$$

and the *group-normalized advantage* (shared across all tokens within the same sampled response)

$$\widehat{A}^{(i)} := \frac{u^{(i)} - \bar{u}}{s_u}. \tag{91}$$

By construction, $\sum_{i=1}^{G} \widehat{A}^{(i)} = 0$ and the baseline is computed with *no* learned value network.

Token and sequence importance ratios. For each sampled response $y^{(i)}$ and each token position t define the token-level ratio

$$r_t^{(i)}(\theta) := \frac{\pi_\theta\big(y_t^{(i)} \mid x, y_{<t}^{(i)}\big)}{\pi_{\theta_{\text{old}}}\big(y_t^{(i)} \mid x, y_{<t}^{(i)}\big)}. \tag{92}$$

Token ratios can be high-variance for long generations. A common stabilization is a *length-normalized sequence ratio* (geometric mean of token ratios):

$$r_{\text{seq}}^{(i)}(\theta) := \left(\frac{\pi_\theta(y^{(i)} \mid x)}{\pi_{\theta_{\text{old}}}(y^{(i)} \mid x)}\right)^{1/T_i} = \exp\!\left(\frac{1}{T_i}\sum_{t=1}^{T_i} \log r_t^{(i)}(\theta)\right). \tag{93}$$

5.3 Group Relative Policy Optimization (GRPO)

Motivation. In LLM alignment, training a separate value network V_ϕ (critic) is often as expensive as training the policy itself. GRPO is a *critic-free* bounded policy optimization method: it replaces the learned baseline with a *group-relative* baseline computed from multiple rollouts of the same prompt, and it keeps updates conservative via a PPO-style clipped surrogate. This combination is widely used in reasoning-style RL for LLMs.

5.3.1 Optimized surrogate objective

Fix a group size $G \in \mathbb{N}$ and a clipping parameter $\epsilon \in (0,1)$. For a prompt $x \sim \rho$, sample

$$y^{(1)}, \dots, y^{(G)} \overset{i.i.d.}{\sim} \pi_{\theta_{\text{old}}}(\cdot \mid x), \qquad y^{(i)} = (y_1^{(i)}, \dots, y_{T_i}^{(i)}), \quad T_i := |y^{(i)}|. \tag{94}$$

Compute scalar rewards $R^{(i)} := R(x, y^{(i)})$ and the group-relative advantage $\widehat{A}^{(i)}$ (e.g. mean/std normalization within the group).

Token-level importance ratio. For each token position t in completion $y^{(i)}$, define the token-level importance ratio

$$r_t^{(i)}(\theta) := \frac{\pi_\theta\big(y_t^{(i)} \mid x, y_{<t}^{(i)}\big)}{\pi_{\theta_{\text{old}}}\big(y_t^{(i)} \mid x, y_{<t}^{(i)}\big)}. \tag{95}$$

Clipped per-token surrogate. Define the clipped per-token term

$$\ell_{i,t}^{\mathrm{GRPO}}(\theta) := \min\Big(r_t^{(i)}(\theta)\,\widehat{A}^{(i)},\ \mathrm{clip}\big(r_t^{(i)}(\theta),\,1-\epsilon,\,1+\epsilon\big)\,\widehat{A}^{(i)}\Big). \tag{96}$$

This is the standard PPO-style clipping applied at token-level.

GRPO Objective. The GRPO surrogate objective is the group-average of token losses, with a KL regularizer to a reference policy (as in the RLHF-style objective introduced earlier).

$$\begin{aligned} L^{\mathrm{GRPO}}(\theta) := \mathbb{E}_{x\sim\rho}\,\mathbb{E}_{\{y^{(i)}\}_{i=1}^{G}\sim(\pi_{\theta_{\mathrm{old}}})^G(\cdot|x)}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T_i}\sum_{t=1}^{T_i}\ell_{i,t}^{\mathrm{GRPO}}(\theta)\right] \\ -\,\beta\,\mathbb{E}_{x\sim\rho}\big[\mathrm{KL}(\pi_\theta(\cdot\mid x)\,\|\,\pi_{\mathrm{ref}}(\cdot\mid x))\big], \end{aligned} \tag{97}$$

where the KL term is implemented in practice via sampled token-level log-ratio penalties (or an explicit per-state KL), exactly as in our earlier KL-regularized LLM objective.

5.3.2 Optimization algorithm (GRPO)

At iteration k with parameters θ_k:

1. **Freeze behavior policy:** set $\theta_{\mathrm{old}} \leftarrow \theta_k$.
2. **Sample prompts:** draw $x^{(1)},\dots,x^{(B)} \sim \rho$.
3. **Group rollouts:** for each $x^{(b)}$, sample G completions $y^{(b,1)},\dots,y^{(b,G)} \sim \pi_{\theta_{\mathrm{old}}}(\cdot \mid x^{(b)})$.
4. **Compute rewards and advantages:** compute $R^{(b,i)} = R(x^{(b)}, y^{(b,i)})$ and $\widehat{A}^{(b,i)}$ (group-relative baseline; typically mean/std normalization within group).
5. **Optimize:** perform E epochs of stochastic gradient ascent on $L^{\mathrm{GRPO}}(\theta)$ (using ratios (95) and clipped term (96)), updating $\theta \leftarrow \theta + \alpha\,\widehat{\nabla_\theta L^{\mathrm{GRPO}}}(\theta)$.

5.4 Modifications of GRPO

5.4.1 Dr. GRPO (Group Relative Policy Optimization Done Right)

Motivation: removing aggregation bias. Follow-up work observed that the original GRPO aggregation $\frac{1}{T_i}\sum_{t=1}^{T_i}\cdot$ introduces a *length-dependent* scaling of token gradients, and that dividing by the within-group reward standard deviation can introduce a *question-difficulty bias* (since groups with smaller reward variance get amplified). Dr. GRPO proposes to remove these normalizations, yielding a cleaner, empirically more stable update.

(1) Advantage without std scaling. Dr. GRPO keeps the group-relative mean baseline but removes the standard-deviation scaling:

$$\widehat{A}_{\mathrm{Dr}}^{(i)} := R^{(i)} - \bar{R}, \qquad \bar{R} := \frac{1}{G}\sum_{j=1}^{G} R^{(j)}. \tag{98}$$

This addresses the difficulty-dependent rescaling effect attributed to $\mathrm{std}(\{R^{(j)}\})$.

(2) Token aggregation with a global normalizer. Instead of per-response normalization by T_i, Dr. GRPO aggregates token losses and normalizes by a *global constant* (often chosen as the maximum completion length $T_{\max}$ used in training). Define the Dr. GRPO per-token clipped term by reusing (96) but replacing $\widehat{A}^{(i)}$ with $\widehat{A}^{(i)}_{\mathrm{Dr}}$:

$$\ell^{\mathrm{DrGRPO}}_{i,t}(\theta) := \min\Big(r^{(i)}_t(\theta)\,\widehat{A}^{(i)}_{\mathrm{Dr}},\ \mathrm{clip}\big(r^{(i)}_t(\theta),\,1-\epsilon,\,1+\epsilon\big)\,\widehat{A}^{(i)}_{\mathrm{Dr}}\Big). \tag{99}$$

Then the Dr. GRPO surrogate is

$$L^{\mathrm{DrGRPO}}(\theta) := \mathbb{E}_{x\sim\rho}\,\mathbb{E}_{\{y^{(i)}\}_{i=1}^{G}\sim(\pi_{\theta_{\mathrm{old}}})^{G}(\cdot|x)}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T_{\max}}\sum_{t=1}^{T_i}\ell^{\mathrm{DrGRPO}}_{i,t}(\theta)\right] - \beta\,\mathbb{E}_{x\sim\rho}\big[\mathrm{KL}(\pi_\theta(\cdot\mid x)\,\|\,\pi_{\mathrm{ref}}(\cdot\mid x))\big]. \tag{100}$$

Because the normalization no longer depends on T_i, this objective avoids incentivizing length changes purely through the loss aggregation rule.

Algorithmic changes. Dr. GRPO is implemented by the same loop as GRPO, but with: (i) advantages (98) (no std scaling), (ii) aggregation (100) (global constant normalizer).

5.4.2 DAPO: Decoupled Clip and Dynamic sAmpling Policy Optimization

Motivation: entropy/gradient collapse and length bias. DAPO is a GRPO-based recipe designed for stable large-scale reasoning RL, addressing several failure modes observed in practice: (i) *entropy collapse* from overly restrictive symmetric clipping, (ii) *vanishing gradients* when a whole group has identical rewards (hence zero group-relative signal), (iii) *length-related bias* in how token losses are aggregated, and (iv) reward noise induced by truncation/overlong generations.

(1) Decoupled (asymmetric) clipping. Replace symmetric clipping with separate lower/upper bounds:

$$\mathrm{clip}_{\mathrm{d}}(r) := \mathrm{clip}\big(r,\ 1-\epsilon_{\mathrm{low}},\ 1+\epsilon_{\mathrm{high}}\big), \qquad \epsilon_{\mathrm{low}},\epsilon_{\mathrm{high}} > 0. \tag{101}$$

This permits larger probability *increases* (when beneficial) while keeping probability decreases conservative, which empirically helps maintain exploration.

(2) Dynamic sampling to avoid zero-signal groups. If all sampled completions for a prompt receive the same reward, then $\widehat{A}^{(i)} \equiv 0$ (or undefined when dividing by std), producing a zero policy gradient. DAPO therefore resamples prompts/groups until they contain a non-degenerate reward signal (e.g. nonzero within-group reward variance, or a mix of outcomes) so that each batch carries learning signal.

(3) Token-level aggregation that removes length bias. Rather than averaging token losses inside each response via $1/T_i$, DAPO aggregates token losses over the batch and normalizes by the number of *active tokens* (or an equivalent global normalizer), which reduces systematic preference for certain lengths induced by per-response scaling. Formally, if a batch contains total active token count $N_{\mathrm{tok}} := \sum_{b=1}^{B}\sum_{i=1}^{G} T_{b,i}$, then a clean mathematical surrogate is:

$$\begin{aligned} L^{\mathrm{DAPO}}(\theta) := \mathbb{E}\left[\frac{1}{N_{\mathrm{tok}}}\sum_{b=1}^{B}\sum_{i=1}^{G}\sum_{t=1}^{T_{b,i}}\min\Big(r^{(b,i)}_t(\theta)\,\widehat{A}^{(b,i)},\ \mathrm{clip}_{\mathrm{d}}\big(r^{(b,i)}_t(\theta)\big)\,\widehat{A}^{(b,i)}\Big)\right] \\ - \beta\,\mathbb{E}_{x\sim\rho}\big[\mathrm{KL}(\pi_\theta(\cdot\mid x)\,\|\,\pi_{\mathrm{ref}}(\cdot\mid x))\big] + \text{(optional overlong reward shaping / filtering terms)}. \end{aligned} \tag{102}$$

DAPO additionally recommends explicit handling of overlong/truncated completions (filtering or reward shaping) to stabilize training in long-CoT regimes.

Algorithmic changes (relative to GRPO). DAPO keeps the GRPO outer loop but modifies:

- *clipping:* use (101) instead of symmetric clipping,
- *sampling:* apply dynamic sampling to avoid degenerate (zero-signal) groups,
- *aggregation:* normalize token losses globally (e.g. by active tokens) instead of $1/T_i$,
- *reward handling:* optionally filter/shape overlong completions.

These components are presented as the core practical recipe in the DAPO paper and reference implementations.

5.5 Group Sequence Policy Optimization (GSPO)

Setup: off-policy optimization and importance sampling. Fix a prompt x. Let $\pi_{\theta_{\text{old}}}$ be the behavior policy used to generate rollouts, and let π_θ be the policy being optimized. We observe responses $y \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$ and wish to optimize an objective that (implicitly) concerns expectations under $\pi_\theta(\cdot \mid x)$.

5.5.1 Token-level vs. sequence-level importance ratios as change of measure

Sequence distribution and exact IS weight. For an autoregressive policy,

$$\pi_\theta(y \mid x) = \prod_{t=1}^{|y|} \pi_\theta(y_t \mid x, y_{<t}). \tag{103}$$

Define the *token ratio*

$$r_t(\theta) := \frac{\pi_\theta(y_t \mid x, y_{<t})}{\pi_{\theta_{\text{old}}}(y_t \mid x, y_{<t})}, \tag{104}$$

and the *sequence (trajectory) ratio*, i.e. the Radon–Nikodym derivative of $\pi_\theta(\cdot \mid x)$ w.r.t. $\pi_{\theta_{\text{old}}}(\cdot \mid x)$:

$$w(\theta; y, x) := \frac{\pi_\theta(y \mid x)}{\pi_{\theta_{\text{old}}}(y \mid x)} = \prod_{t=1}^{|y|} r_t(\theta). \tag{105}$$

Then, for any integrable functional $F(x, y)$,

$$\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[F(x,y)] = \mathbb{E}_{y\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\big[w(\theta; y, x)\, F(x,y)\big]. \tag{106}$$

Equation (106) is the basic principle of importance sampling.

Why token ratios are not "the" IS correction for sequence-level reward. In group-based LLM RL, the reward and advantage are typically *sequence-level* (one scalar per response). GRPO applies *token-level* ratios $r_t(\theta)$ inside a token average, which is not the same as the exact sequence-level IS correction $w(\theta; y, x) = \prod_t r_t(\theta)$ for a sequence-level quantity. GSPO argues this mismatch is a fundamental source of instability and variance accumulation, especially for long responses.

Concretely, suppose $F(x, y) = \widehat{A}(x, y)$ is a scalar (sequence) advantage. Exact IS gives

$$\mathbb{E}_{\pi_\theta}[\widehat{A}] = \mathbb{E}_{\pi_{\theta_{\text{old}}}}[w(\theta; y, x)\widehat{A}],$$

whereas the GRPO token-weighted surrogate resembles $\mathbb{E}_{\pi_{\theta_{\text{old}}}}\big[\frac{1}{|y|}\sum_t r_t(\theta)\widehat{A}\big]$, which is generally *not* equal to the exact IS estimator unless strong special conditions hold.

Variance and scaling. Even the exact ratio $w(\theta; y, x) = \prod_t r_t(\theta)$ can be extremely high-variance for long sequences (product of many random ratios). GSPO therefore introduces a length-normalized (geometric mean) ratio:

$$s(\theta; y, x) := \left(\frac{\pi_\theta(y \mid x)}{\pi_{\theta_{\text{old}}}(y \mid x)} \right)^{1/|y|} = \exp\left(\frac{1}{|y|} \sum_{t=1}^{|y|} \log r_t(\theta) \right). \tag{107}$$

This is exactly the ratio used by GSPO. It places responses of different lengths on a comparable numerical scale and reduces the sensitivity to a few extreme token ratios (since it averages log-ratios).

5.5.2 GSPO objective (sequence-level clipping, reward alignment)

Group advantage (sequence-level). For a group of G responses $\{y^{(i)}\}_{i=1}^G$ sampled i.i.d. from $\pi_{\theta_{\text{old}}}(\cdot \mid x)$, compute rewards $R^{(i)} = R(x, y^{(i)})$ and define the group-normalized advantage

$$\widehat{A}^{(i)} := \frac{R^{(i)} - \text{mean}\{R^{(j)}\}_{j=1}^G}{\text{std}\{R^{(j)}\}_{j=1}^G}. \tag{108}$$

GSPO uses this sequence-level advantage, matching the fact that the reward is granted to the entire response.

GSPO surrogate. Let $\epsilon > 0$ be the clipping range. With $s^{(i)}(\theta) := s(\theta; y^{(i)}, x)$ from (107), GSPO maximizes

$$J_{\text{GSPO}}(\theta) := \mathbb{E}_{x\sim\rho,\ \{y^{(i)}\}_{i=1}^G \sim (\pi_{\theta_{\text{old}}})^G(\cdot|x)} \left[\frac{1}{G} \sum_{i=1}^G \min\Big(s^{(i)}(\theta)\widehat{A}^{(i)},\ \text{clip}(s^{(i)}(\theta), 1-\epsilon, 1+\epsilon)\widehat{A}^{(i)} \Big) \right]. \tag{109}$$

This is the GSPO objective (sequence-level ratio + sequence-level clipping).

5.5.3 Gradient comparison: GSPO vs. GRPO (why GSPO is sequence-coherent)

GSPO gradient (no clipping, for clarity). Omitting clipping for the derivation (as done in the GSPO paper), we have

$$\nabla_\theta J_{\text{GSPO}}(\theta) = \mathbb{E}\left[\frac{1}{G} \sum_{i=1}^G s^{(i)}(\theta)\widehat{A}^{(i)}\ \nabla_\theta \log s^{(i)}(\theta) \right] \tag{110}$$

$$= \mathbb{E}\left[\frac{1}{G} \sum_{i=1}^G s^{(i)}(\theta)\widehat{A}^{(i)} \cdot \frac{1}{|y^{(i)}|} \sum_{t=1}^{|y^{(i)}|} \nabla_\theta \log \pi_\theta\big(y_t^{(i)} \mid x, y_{<t}^{(i)}\big) \right]. \tag{111}$$

This matches the GSPO gradient analysis: *all tokens in the same response share the same scalar weight* $s^{(i)}(\theta)\widehat{A}^{(i)}$ (up to the $1/|y^{(i)}|$ normalization).

GRPO gradient (no clipping, for comparison). Again omitting clipping, GRPO yields

$$\nabla_\theta J_{\text{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{G} \sum_{i=1}^G \widehat{A}^{(i)} \cdot \frac{1}{|y^{(i)}|} \sum_{t=1}^{|y^{(i)}|} r_t^{(i)}(\theta)\ \nabla_\theta \log \pi_\theta\big(y_t^{(i)} \mid x, y_{<t}^{(i)}\big) \right], \tag{112}$$

so token weights vary across positions through $r_t^{(i)}(\theta)$. GSPO emphasizes that these unequal token weights can accumulate unpredictably for long sequences and contribute to collapse, while GSPO avoids this by weighting tokens uniformly within a response.

5.5.4 Algorithm (GSPO)

At iteration k:

1. Freeze behavior policy: $\pi_{\theta_{\text{old}}} \leftarrow \pi_{\theta_k}$.
2. Sample prompts $x^{(1)}, \dots, x^{(B)} \sim \rho$.
3. For each prompt $x^{(b)}$, sample G responses $y^{(b,1)}, \dots, y^{(b,G)} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x^{(b)})$.
4. Compute rewards $R^{(b,i)}$, advantages $\widehat{A}^{(b,i)}$ via (108), and ratios $s^{(b,i)}(\theta)$ via (107).
5. Perform E epochs of minibatch stochastic gradient ascent on $J_{\text{GSPO}}(\theta)$ in (109).

(Practical note in GSPO: sequence-level likelihoods are less sensitive to token-level precision/routing issues, aiding stability in MoE training.)

5.6 Soft Adaptive Policy Optimization (SAPO)

Motivation. Hard clipping (as in GRPO/PPO-style objectives) induces a binary trust region: samples outside the clipping band contribute *zero* policy gradient, while samples inside contribute fully. This all-or-nothing behavior can be brittle: tight clipping discards too much learning signal, whereas loose clipping may admit unstable off-policy updates. SAPO replaces hard clipping by a *smooth, adaptive gate* that continuously attenuates gradient contributions as the importance ratio deviates from 1, while preserving the on-policy gradient at $r = 1$.

5.6.1 Setup: group sampling and critic-free advantages

Fix group size $G \in \mathbb{N}$. For a prompt $x \sim \rho$, sample a group of responses from the behavior policy $\pi_{\theta_{\text{old}}}$:

$$y^{(1)}, \dots, y^{(G)} \overset{i.i.d.}{\sim} \pi_{\theta_{\text{old}}}(\cdot \mid x), \qquad y^{(i)} = (y_1^{(i)}, \dots, y_{T_i}^{(i)}), \quad T_i := |y^{(i)}|. \tag{113}$$

Compute sequence-level scores $R^{(i)} := R(x, y^{(i)})$ and define the group-normalized advantage (shared across all tokens in the same response)

$$\widehat{A}^{(i)} := \frac{R^{(i)} - \bar{R}}{s_R}, \qquad \bar{R} := \frac{1}{G}\sum_{j=1}^{G} R^{(j)}, \qquad s_R := \sqrt{\frac{1}{G}\sum_{j=1}^{G} \big(R^{(j)} - \bar{R}\big)^2} + \varepsilon,\ \varepsilon > 0. \tag{114}$$

5.6.2 From GRPO hard clipping to a soft gate

Token-level ratio. For each token position t in response $y^{(i)}$, define the token-wise importance ratio

$$r_t^{(i)}(\theta) := \frac{\pi_\theta\big(y_t^{(i)} \mid x, y_{<t}^{(i)}\big)}{\pi_{\theta_{\text{old}}}\big(y_t^{(i)} \mid x, y_{<t}^{(i)}\big)}. \tag{115}$$

GRPO clipping as a hard gate (binary trust region). A GRPO/PPO-style clipped term at a single token has the form

$$\ell^{\mathrm{CLIP}}(r, \widehat{A}) = \min\Big(r\,\widehat{A},\ \mathrm{clip}(r, 1-\epsilon, 1+\epsilon)\,\widehat{A}\Big).$$

Its (sub)gradient w.r.t. r equals a *binary* gate:

$$\frac{\partial}{\partial r}\ell^{\mathrm{CLIP}}(r, \widehat{A}) = g_{\epsilon}^{\mathrm{hard}}(r, \widehat{A}) := \mathbf{1}\{\widehat{A} > 0,\ r \le 1+\epsilon\} + \mathbf{1}\{\widehat{A} \le 0,\ r \ge 1-\epsilon\}. \tag{116}$$

Thus clipping is equivalent to multiplying the policy-gradient direction by a hard (0/1) gate.

SAPO idea. Replace the hard gate (116) with a *smooth* gate that is close to 1 when $r \approx 1$ and decays continuously as r moves away from 1.

5.6.3 SAPO objective

Smooth shaping function. Define a smooth shaping function (applied to ratios)

$$f_{i,t}(r) := \frac{4}{\tau_{i,t}}\,\sigma\big(\tau_{i,t}(r-1)\big), \qquad \sigma(z) := \frac{1}{1+e^{-z}}, \qquad \tau_{i,t} > 0. \tag{117}$$

SAPO uses an advantage-sign dependent temperature:

$$\tau_{i,t} := \begin{cases} \tau_{\mathrm{pos}}, & \widehat{A}^{(i)} > 0, \\ \tau_{\mathrm{neg}}, & \widehat{A}^{(i)} \le 0. \end{cases} \tag{118}$$

Typically $\tau_{\mathrm{neg}} > \tau_{\mathrm{pos}}$, which attenuates negative-advantage updates more aggressively.

SAPO surrogate (policy part). SAPO maximizes the following critic-free surrogate (policy) objective:

$$J_{\mathrm{SAPO}}(\theta) := \mathbb{E}_{x\sim\rho}\,\mathbb{E}_{\{y^{(i)}\}_{i=1}^{G}\sim(\pi_{\theta_{\mathrm{old}}})^{G}(\cdot|x)}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T_i}\sum_{t=1}^{T_i} f_{i,t}\big(r_t^{(i)}(\theta)\big)\ \widehat{A}^{(i)}\right]. \tag{119}$$

Optional KL regularization to a reference policy. If a reference policy π_{ref} is used, add the standard penalty

$$-\beta\,\widehat{\mathbb{E}}\left[\sum_{t=1}^{T_i}\log\frac{\pi_\theta(y_t^{(i)} \mid x, y_{<t}^{(i)})}{\pi_{\mathrm{ref}}(y_t^{(i)} \mid x, y_{<t}^{(i)})}\right],$$

to (119). (We omit it in the derivations below for readability.)

5.6.4 SAPO gradient and the gate interpretation

SAPO gate (the additional gradient weight). Differentiate $f_{i,t}(r)$ w.r.t. r:

$$g_{i,t}(\theta) := \frac{d}{dr}f_{i,t}(r)\Big|_{r=r_t^{(i)}(\theta)} = 4\,\sigma\big(\tau_{i,t}(r_t^{(i)}(\theta)-1)\big)\Big(1-\sigma\big(\tau_{i,t}(r_t^{(i)}(\theta)-1)\big)\Big) =^{2}\Big(\frac{\tau_{i,t}}{2}\,(r_t^{(i)}(\theta)-1)\Big). \tag{120}$$

This satisfies $g_{i,t}(\theta) \in (0,1]$ and $g_{i,t}(\theta) = 1$ at the on-policy point $r_t^{(i)}(\theta) = 1$.

Token-level policy-gradient contribution. Using $\nabla_\theta r_t^{(i)}(\theta) = r_t^{(i)}(\theta) \nabla_\theta \log \pi_\theta(y_t^{(i)} \mid x, y_{<t}^{(i)})$, we obtain for each token (i, t):

$$\nabla_\theta \Big(f_{i,t}(r_t^{(i)}(\theta))\, \widehat{A}^{(i)} \Big) = \underbrace{g_{i,t}(\theta)}_{\textbf{SAPO gate}} \cdot \underbrace{r_t^{(i)}(\theta)\, \widehat{A}^{(i)}}_{\text{standard IS} \times \text{advantage}} \cdot \nabla_\theta \log \pi_\theta\big(y_t^{(i)} \mid x, y_{<t}^{(i)}\big). \tag{121}$$

Full gradient (policy part). Aggregating over group samples and token positions,

$$\nabla_\theta J_{\text{SAPO}}(\theta) = \mathbb{E} \left[\frac{1}{G} \sum_{i=1}^{G} \frac{1}{T_i} \sum_{t=1}^{T_i} g_{i,t}(\theta)\; r_t^{(i)}(\theta)\; \widehat{A}^{(i)}\; \nabla_\theta \log \pi_\theta\big(y_t^{(i)} \mid x, y_{<t}^{(i)}\big) \right]. \tag{122}$$

Relative to the unclipped token-ratio objective (which would weight by $r_t^{(i)}(\theta)\widehat{A}^{(i)}$), SAPO inserts the extra multiplicative gate $g_{i,t}(\theta)$.

5.6.5 Algorithm (SAPO)

Fix hyperparameters: group size G, temperatures $(\tau_{\text{pos}}, \tau_{\text{neg}})$, and (optionally) KL coefficient β.

At iteration k:

1. **Freeze behavior policy:** set $\theta_{\text{old}} \leftarrow \theta_k$.
2. **Sample prompts:** draw $x^{(1)}, \dots, x^{(B)} \sim \rho$.
3. **Group rollouts:** for each $x^{(b)}$, sample G responses $y^{(b,1)}, \dots, y^{(b,G)} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x^{(b)})$.
4. **Compute group advantages:** compute rewards $R^{(b,i)} = R(x^{(b)}, y^{(b,i)})$ and advantages $\widehat{A}^{(b,i)}$ via (114).
5. **Optimize:** for E epochs, perform minibatch stochastic gradient ascent on $J_{\text{SAPO}}(\theta)$: for each token compute $r_t^{(b,i)}(\theta)$ via (115), set $\tau_{b,i,t}$ via (118), compute the gate $g_{b,i,t}(\theta)$ via (120), and update θ using the gradient form (122) (plus the gradient of the KL regularizer if included).
6. **Commit:** set $\theta_{k+1} \leftarrow \theta$.