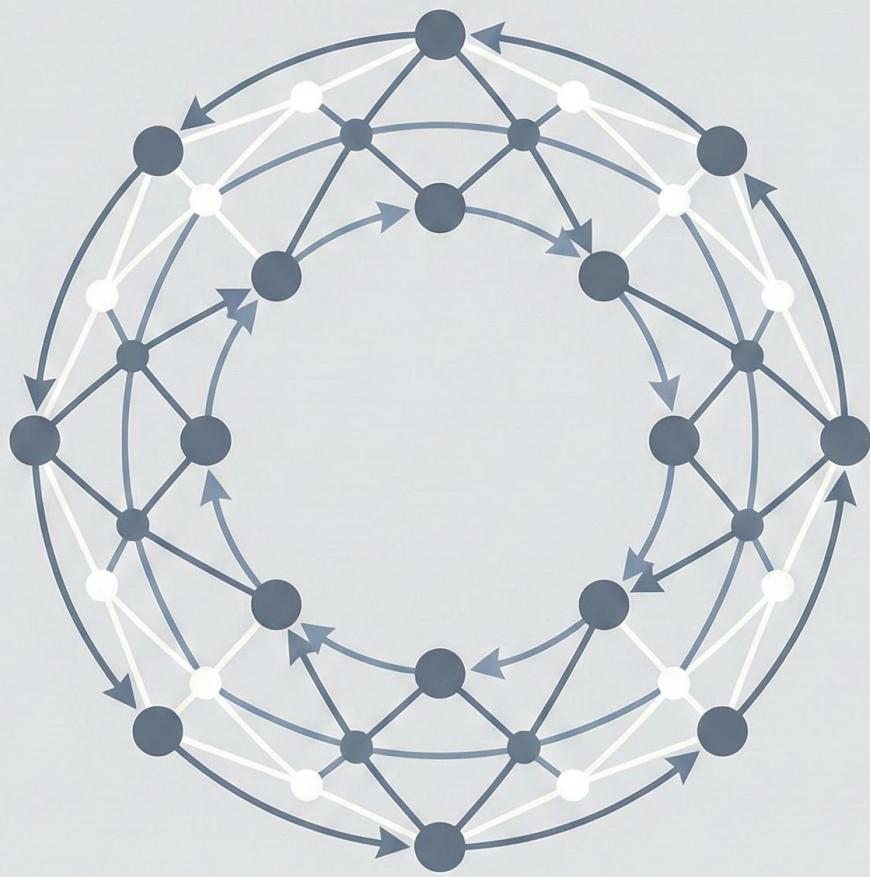


REINFORCEMENT LEARNING NOTES



Joshua Lin

1 Value Methods

Stationary Policies

A *Markov* policy is one which is only dependent on the current state. A *stationary* policy is a Markov policy whose decisions are time-independent.

Value Functions

Fix an MDP $M = (\mathcal{S}, \mathcal{A}, \mathbb{P}, \mu, r, \gamma)$. The *state-value function* is $V^\pi : \mathcal{S} \rightarrow \mathbb{R}$, while the *action-value function* is $Q^\pi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. The functions are defined by (τ is a trajectory)

$$V^\pi(s) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^T \gamma^t r(s_t, a_t) \mid s_0 = s \right]$$

$$Q^\pi(s, a) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^T \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a \right]$$

For simplicity of analysis, we often assume $T = \infty$. Note $\|V^\pi\|_\infty, \|Q^\pi\|_\infty \leq \frac{1}{1-\gamma}$, the latter of which is the *effective horizon*. They are connected by the *Bellman Consistency Equations*,

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

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

Remark. There is an operator-theoretic representation: view $V^\pi \in \mathbb{R}^S$ and $Q^\pi \in \mathbb{R}^{SA}$, and define $\mathbb{P} \in \mathbb{R}^{SA \times S}$ to be the transition dynamics $\mathbb{P}_{(s,a),s'} = \mathbb{P}(s' | s, a)$. Then TODO ■

Value Iteration

The goal of value iteration is to find the policy π^* whose values satisfy $V^*(s) = \max_\pi V^\pi(s)$ and $Q^*(s) = \max_\pi Q^\pi(s)$. It turns out there is a *deterministic* policy which is *simultaneously* optimal for all starting states s .

SARSA (TODO)

Deep Q Networks (TODO)

Rainbow (TODO)

2 Policy Methods

Policy Optimization

Model the policy using a neural network $\pi_\theta : \mathcal{S} \rightarrow \mathcal{A}$. Then the problem becomes

$$\underset{\theta}{\text{maximize}} \quad J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)] = \mathbb{E}_{s \sim \mu}[V^{\pi_\theta}(s)]$$

where $\tau = (s_0, a_0, r_0, \dots, s_T, a_T, r_T)$ is a trajectory, $R_t(\tau) = \sum_{t' \geq t} \gamma^{t'-t} r(s_{t'}, a_{t'})$ is a one-step estimate of $Q^{\pi_\theta}(s_t, a_t)$, and μ is the initial distribution. By convention, we set $R(\tau) = R_0(\tau)$. The optimizer is trained using gradient ascent,

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\pi_\theta)$$

for some appropriate learning rate $\alpha > 0$.

Remark. Advantages compared to DQN include a more efficient approach to learning continuous action spaces and encouraging more exploration (not just epsilon-greedy). Many policy gradient-methods are on-policy, allowing the model to directly optimize for its evaluated behavior.

Note that the last equality in the objective follows by conditioning on the initial state,

$$\mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)] = \mathbb{E}_{s \sim \mu}[\mathbb{E}_{\tau \sim \pi_\theta|s_0=s}[R(\tau)]] = \mathbb{E}_{s \sim \mu}[V^{\pi_\theta}(s)]$$

We typically use the latter characterization when analyzing infinite time horizons since V^{π_θ} can be manipulated using the Bellman equations. ■

Policy Gradients

Evaluating $\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)]$ is generally a difficult problem because the transition dynamics $\mathbb{P}(s' | s, a)$ are unknown. However, it turns out that

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

where the last equality follows from the identity $Q^{\pi_\theta}(s_t, a_t) = \mathbb{E}_{\tau \sim \pi_\theta}[R_t(\tau) | s_t, a_t]$. Notably, $\nabla_\theta \log \pi_\theta(a_t | s_t)$ can be found using modern auto-differentiation tools. So, we can estimate the policy gradient by averaging this object over several trajectories.

Remark. A straightforward proof is as follows via the **log-derivative trick**. For any function $f(x)$ and likelihood $p(x; \theta)$,

$$\begin{aligned} \nabla_\theta \mathbb{E}_{x \sim p(x; \theta)}[f(x)] &= \nabla_\theta \int f(x) p(x; \theta) dx = \int f(x) \nabla_\theta p(x; \theta) dx \\ &= \int f(x) p(x; \theta) \frac{\nabla_\theta p(x; \theta)}{p(x; \theta)} dx = \int f(x) \nabla_\theta \log p(x; \theta) p(x; \theta) dx \\ &= \mathbb{E}_{x \sim p(x; \theta)}[f(x) \nabla_\theta \log p(x; \theta)] \end{aligned}$$

and thus $\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau) \nabla_\theta \log p(\tau; \theta)]$. So we just need to differentiate the log-likelihood. Observe that if μ is the initial state distribution,

$$\begin{aligned} p(\tau; \theta) &= \mu(s_0) \prod_{t=0}^T \mathbb{P}(s_{t+1} | s_t, a_t) \pi_\theta(a_t | s_t) \\ \implies \log p(\tau; \theta) &= \log \mu(s_0) + \sum_{t=0}^T (\log \mathbb{P}(s_{t+1} | s_t, a_t) + \log \pi_\theta(a_t | s_t)) \end{aligned}$$

so $\nabla_\theta \log p(\tau; \theta) = \sum_{t=0}^T \nabla_\theta \log \pi_\theta(a_t | s_t)$, where $\nabla_\theta \log \pi_\theta(a | s)$ is the **score function**. Thus,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[R(\tau) \sum_{t=0}^T \nabla_\theta \log \pi_\theta(a_t | s_t) \right] = \sum_{t=0}^T \sum_{t'=0}^T \gamma^{t'} \mathbb{E}_{\tau \sim \pi_\theta}[r_{t'} \nabla_\theta \log \pi_\theta(a_t | s_t)]$$

Note that for $t' < t$, conditioning on the history until immediately before a_t is selected allows us to remove $r_{t'}$ from the conditional expectation:

$$\mathbb{E}_{\tau \sim \pi_\theta}[r_{t'} \nabla_\theta \log \pi_\theta(a_t | s_t)] = \mathbb{E}_{\tau \sim \pi_\theta}[r_{t'} \mathbb{E}_{a_t \sim \pi_\theta(\cdot | s_t)}[\nabla_\theta \log \pi_\theta(a_t | s_t)]]$$

But it turns out that the **expectation of the score function is zero**:

$$\mathbb{E}_{a_t \sim \pi_\theta(\cdot | s_t)}[\nabla_\theta \log \pi_\theta(a_t | s_t)] = \int_{\mathcal{A}} \pi_\theta(a_t | s_t) \nabla_\theta \log \pi_\theta(a_t | s_t) da_t = \nabla_\theta \underbrace{\int_{\mathcal{A}} \pi_\theta(a_t | s_t) da_t}_{=1} = 0$$

So only terms for which $t' \geq t$ survive in the policy gradient; hence

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T \sum_{t'=t}^T (\gamma^{t'} r_{t'}) \nabla_\theta \log \pi_\theta(a_t | s_t) \right] = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T \gamma^t R_t(\tau) \nabla_\theta \log \pi_\theta(a_t | s_t) \right]$$

which is precisely as desired. ■

Discounted State Occupancy Measure (DSOM)

It is a probability mass $d^{\pi_\theta} : \mathcal{S} \rightarrow [0, 1]$ representing the total discounted proportion of time spent in state s following policy π_θ ,

$$d^{\pi_\theta}(s) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}^{\pi_\theta}[s_t = s]$$

Importantly, for any function $f(s_t, a_t)$, we have the **occupancy measure transform**,

$$\mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T \gamma^t f(s_t, a_t) \right] = \underbrace{(1 - \gamma)^{-1}}_{\text{effective horizon}} \underbrace{\mathbb{E}_{s \sim d^{\pi_\theta}, a \sim \pi_\theta(\cdot | s)}[f(s, a)]}_{\text{expected } f \text{ under DSOM}}$$

An important special case arises from our existing identities for $J(\theta)$ and $\nabla_\theta J(\theta)$:

$$\begin{aligned} J(\theta) &= (1 - \gamma) \mathbb{E}_{s \sim d^{\pi_\theta}, a \sim \pi_\theta(\cdot | s)}[r(s, a)] \\ \nabla_\theta J(\theta) &= (1 - \gamma) \mathbb{E}_{s \sim d^{\pi_\theta}, a \sim \pi_\theta(\cdot | s)}[A^{\pi_\theta}(s, a) \nabla_\theta \log \pi_\theta(a | s)] \end{aligned}$$

The DSOM can be extended to continuous state spaces as well. Notably, the expressions for $J(\theta)$ and $\nabla_\theta J(\theta)$ remain identical.

Remark. It's straightforward to prove the transformation in the discrete case:

$$\begin{aligned}\mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t f(s_t, a_t) \right] &= \sum_{t=0}^{\infty} \gamma^t \sum_{s \in \mathcal{S}} \mathbb{P}^{\pi_\theta}(s_t = s) \sum_{a \in \mathcal{A}} \pi(a | s) f(s, a) \\ &= \sum_{s \in \mathcal{S}} \left(\sum_{t=0}^{\infty} \gamma^t \mathbb{P}^{\pi_\theta}(s_t = s) \right) \sum_{a \in \mathcal{A}} \pi(a | s) f(s, a) \\ &= (1 - \gamma) \sum_{s \in \mathcal{S}} d^{\pi_\theta}(s) \sum_{a \in \mathcal{A}} \pi(a | s) f(s, a) \\ &= (1 - \gamma) \mathbb{E}_{s \sim d^{\pi_\theta}, a \sim \pi(\cdot | s)} [f(s, a)]\end{aligned}$$

In the gradient expression, we can replace $Q^{\pi_\theta}(s, a) = A^{\pi_\theta}(s, a) + V^{\pi_\theta}(s)$ with $A^{\pi_\theta}(s, a)$ since

$$\mathbb{E}_{s \sim d^{\pi_\theta}, a \sim \pi_\theta(\cdot | s)} [V^{\pi_\theta}(s) \nabla_\theta \log \pi_\theta(a | s)] = \mathbb{E}_{s \sim d^{\pi_\theta}} [V^{\pi_\theta}(s) \mathbb{E}_{a \sim \pi_\theta(\cdot | s)} [\nabla_\theta \log \pi_\theta(a | s)]] = 0$$

and in fact, **adding any state function $b(s)$ does not change the bias.** ■

Importance Sampling

For likelihoods p_1, p_2 and function f satisfying $p_1 f \ll p_2$ (absolute continuity),

$$\mathbb{E}_{x \sim p_1} [f(x)] = \int_{\mathcal{X}} f(x) p_1(x) dx = \int_{\mathcal{X}} \frac{p_1(x)}{p_2(x)} f(x) p_2(x) dx = \mathbb{E}_{x \sim p_2} \left[\frac{p_1(x)}{p_2(x)} f(x) \right]$$

The likelihood ratio $p_1(x)/p_2(x)$ is called the *importance weight*. Informally, $x \sim p_2$ "cancels" with $p_2(x)$ and is replaced by $x \sim p_1$.

2.1 REINFORCE

REINFORCE (Williams, 1992)

The key idea is to estimate the policy gradient using one-sample Monte-Carlo:

REINFORCE Algorithm

- 1: Initialize learning rate α
- 2: Initialize weights θ of a policy network π_θ
- 3: **for** $episode = 0, \dots, \text{MAX_EPISODE}$ **do**
- 4: Sample a trajectory $\tau = (s_0, a_0, r_0, \dots, s_T, a_T, r_T) \sim \pi_\theta$
- 5: Set $\nabla_\theta J(\theta) \leftarrow 0$
- 6: **for** $t = 0, \dots, T$ **do**
- 7: $R_t(\tau) \leftarrow \sum_{t'=t}^T \gamma^{t'-t} r_{t'}$
- 8: $\nabla_\theta J(\theta) \leftarrow \nabla_\theta J(\theta) + R_t(\tau) \nabla_\theta \log \pi_\theta(a_t | s_t)$
- 9: **end for**
- 10: $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
- 11: **end for**

Remark. **Notes.** Each trajectory's experiences are discarded after each use; REINFORCE is an on-policy algorithm. Some ways to reduce the variance of the algorithm include subtracting a state-dependent baseline,

$$\nabla_{\theta} J(\theta) = \sum_{t=0}^T (R_t(\tau) - b(s_t)) \nabla_{\theta} \log \pi_{\theta}(a_t | s_t)$$

One option for b is the value function V^{π} , which motivates the Actor-Critic algorithm. Another is the average (state-independent) reward-to-date in the trajectory, $b = \frac{1}{T} \sum_{t=0}^T R_t(\tau)$, which encourages half of the better actions and discouraging half of the worse actions. This can be helpful if, for instance, all of the actions give negative rewards, but some are more negative than others.

Issues. A bad policy (e.g. if exploration is low) leads to bad data, which leads to worse gradient estimates, creating a cycle that leads to performance collapse. REINFORCE is very unstable! ■

2.2 Trust Region Algorithms

Fisher Information Matrix

It is the covariance matrix of the score function, or equivalently the negative expected Hessian of the log-likelihood,

$$\begin{aligned} \mathcal{I}(\pi_{\theta}) &= \mathbb{E}_{s \sim d^{\pi_{\theta}}, a \sim \pi_{\theta}(\cdot | s)} \left[\nabla_{\theta} \log \pi_{\theta}(a | s) \nabla_{\theta} \log \pi_{\theta}(a | s)^{\top} \right] \\ &= \mathbb{E}_{s \sim d^{\pi_{\theta}}, a \sim \pi_{\theta}(\cdot | s)} \left[-\nabla_{\theta}^2 \log \pi_{\theta}(a | s) \right] \end{aligned}$$

where the second equality is known as the **Fisher information identity**. $\mathcal{I}(\pi_{\theta})$ is also the Hessian of the expected conditional KL divergence $\mathcal{I}(\pi_{\theta}) = \nabla_{\theta'}^2 \tilde{\mathbb{D}}_{\text{KL}}[\pi_{\theta} \parallel \pi_{\theta'}]|_{\theta'=\theta}$, where

$$\begin{aligned} \tilde{\mathbb{D}}_{\text{KL}}[\pi_{\theta} \parallel \pi_{\theta'}] &= \mathbb{E}_{s \sim d^{\pi_{\theta}}} [\mathbb{D}_{\text{KL}}[\pi_{\theta}(\cdot | s) \parallel \pi_{\theta'}(\cdot | s)]] \\ &= \mathbb{E}_{s \sim d^{\pi_{\theta}}, a \sim \pi_{\theta}(\cdot | s)} [\log \pi_{\theta}(a | s) - \log \pi_{\theta'}(a | s)] \end{aligned}$$

Since the score has zero expectation, it locally defines the geometry of the *statistical manifold*:

$$\tilde{\mathbb{D}}_{\text{KL}}[\pi_{\theta} \parallel \pi_{\theta+\Delta\theta}] \approx \underbrace{\tilde{\mathbb{D}}_{\text{KL}}[\pi_{\theta} \parallel \pi_{\theta}]}_{=0} + \frac{1}{2} \Delta\theta^{\top} \mathcal{I}(\pi_{\theta}) \Delta\theta = \frac{1}{2} \Delta\theta^{\top} \mathcal{I}(\pi_{\theta}) \Delta\theta$$

(A statistical manifold is a family of distributions $\mathcal{M} = \{p(x; \theta) : \theta \in \Theta\}$ for *parameter space* $\Theta \subseteq \mathbb{R}^n$, with local coordinates θ^i and *Fisher-Rao metric* $g_{ij}(\theta) = \mathcal{I}(\pi_{\theta})_{ij}$).

Remark. In this context, we care about the Fisher information identity primarily since it allows us to connect $\mathcal{I}(\pi_{\theta})$ to the KL divergence. Indeed,

$$\begin{aligned} \nabla_{\theta'}^2 \tilde{\mathbb{D}}_{\text{KL}}[\pi_{\theta} \parallel \pi'_{\theta}]|_{\theta'=\theta} &= \nabla_{\theta'}^2 \mathbb{E}_{s \sim d^{\pi_{\theta}}, a \sim \pi_{\theta}(\cdot | s)} [\log \pi_{\theta}(a | s) - \log \pi'_{\theta}(a | s)] \\ &= \mathbb{E}_{s \sim d^{\pi_{\theta}}, a \sim \pi_{\theta}(\cdot | s)} [-\nabla_{\theta'}^2 \log \pi'_{\theta}(a | s)] = \mathcal{I}(\pi_{\theta}) \end{aligned}$$

So, why is the identity true? Intuitively, for some parametrized distribution $p(x; \theta)$, the Hessian $-\nabla_{\theta}^2 \log p(x; \theta)$ measures how sensitive the log-likelihood is to θ : if it is large (e.g. in operator norm), then the model will sharply degrade under perturbation of θ . On the other hand, $\nabla_{\theta} \log p(x; \theta) \nabla_{\theta} p(x; \theta)^{\top}$ measures the variance of the gradient of log-likelihood.

Therefore, the identity says that on average, if the log-likelihood is very sensitive to θ , then its gradient must fluctuate a lot. Formally,

$$\int_{\mathcal{X}} p(x; \theta) dx = 1 \implies \int_{\mathcal{X}} \nabla_{\theta} p(x; \theta) dx = \int_{\mathcal{X}} (\nabla_{\theta} \log p(x; \theta)) p(x; \theta) dx = 0$$

Differentiating with respect to θ ,

$$\int_{\mathcal{X}} [(\nabla_{\theta}^2 \log p(x; \theta)) p(x; \theta) + \nabla_{\theta} p(x; \theta) \nabla_{\theta} \log p(x; \theta)^{\top}] dx = 0$$

Applying the log-derivative trick again yields the identity. ■

Natural Policy Gradient (Kakade, 2001)

The natural policy gradient defines the optimal search direction subject to constraints in the KL divergence; equivalently, it is the direction of steepest ascent on the statistical manifold defined by the Fisher-Rao metric. It is the (approximate) solution to the following convex optimization, which itself is a first-order expansion of the objective change $\Delta J(\theta)$:

$$\begin{aligned} & \underset{\Delta \theta}{\text{maximize}} \quad \nabla_{\theta} J(\theta)^{\top} \Delta \theta \\ & \text{subject to} \quad \tilde{\mathbb{D}}_{\text{KL}}[\pi_{\theta} \parallel \pi_{\theta+\Delta \theta}] \leq \delta \end{aligned}$$

The optimal step can be found by a Taylor approximation of the KL divergence (using the Fisher information matrix) and enforcing the KKT conditions; its direction is

$$\tilde{\nabla}_{\theta} J(\theta) = \mathcal{I}(\pi_{\theta})^{-1} \nabla_{\theta} J(\theta)$$

with optimal step size

$$\alpha = \sqrt{\frac{2\delta}{\nabla_{\theta} J(\theta)^{\top} \mathcal{I}(\pi_{\theta})^{-1} \nabla_{\theta} J(\theta)}}$$

The natural policy gradient update is $\theta \leftarrow \theta + \alpha \tilde{\nabla}_{\theta} J(\theta)$.

Remark. The natural policy gradient is motivated by these issues with vanilla policy gradients:

- A small step in parameter space can lead to massive distributional shifts (large KL divergence) when the policy is in a sensitive region, leading to *catastrophic performance collapse*.
- Conversely, in flat regions of parameter space, it may take several updates to improve a policy, leading to slow training.

This phenomenon arises because the vanilla policy gradient is *parametrization dependent*. For instance, a linear reparametrization of θ would change the step size and direction: with new coordinates $\phi = A\theta$, the gradient becomes $\nabla_{\phi} J(\pi_{\phi}) = A^{-\top} \nabla_{\theta} J(\theta)$. So, the effective updates are

$$\begin{aligned} \phi_{t+1} &= \phi_t + \alpha A^{-\top} \nabla_{\theta} J(\theta) \\ \theta_{t+1}^{\text{eff}} &= A^{-1} \phi_{t+1} = \theta_t + \alpha (A^{\top} A)^{-1} \nabla_{\theta} J(\theta) \end{aligned}$$

so the step is dependent on parametrization unless A is orthogonal. Informally, the issue is that $\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$ is dimensionally inconsistent since $\nabla_{\theta} J(\theta) = (\partial J / \partial \theta_i)_i$ has dimensions θ_i^{-1} .

Natural policy gradients are limited in practice since inversion of the Fisher information matrix at each iteration is extremely costly at $\mathcal{O}(|\theta|^3)$, especially for neural networks with millions or billions of parameters; additionally, it may be that some steps blow up the KL divergence if the Taylor approximation is not good (e.g., if the Hessian is not positive-definite). However, the ideas form the basis of TRPO, which itself is the foundation of modern policy optimization. ■

The Performance Difference Identity

The *performance difference* between two policies π_θ and $\pi_{\theta'}$ is

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

So, the difference is the average π_θ -advantage sampled from $\pi_{\theta'}$. Note that it vanishes when the policies are equivalent, as expected.

Remark. The identity is a result of the Bellman consistency equations,

$$\begin{aligned} \mathbb{E}_{\tau \sim \pi_{\theta'}} \left[\sum_{t=0}^T \gamma^t A^{\pi_\theta}(s_t, a_t) \right] &= \mathbb{E}_{\tau \sim \pi_{\theta'}} \left[\sum_{t=0}^T \gamma^t (r(s_t, a_t) + \gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)) \right] \\ &= \mathbb{E}_{\tau \sim \pi_{\theta'}} [R(\tau) - V^{\pi_\theta}(s_0)] = J(\pi'_\theta) - J(\theta) \end{aligned}$$

The penultimate step follows since the last two series telescope. Note that the final step is only true under the MDP assumption that the initial state distribution μ is set by the environment. ■

Conservative Policy Iteration (Kakade & Langford, 2002)

CPI provides monotonic improvement guarantees for *mixture* policies as a way to address catastrophic performance collapse. It does this by introducing a *surrogate objective*,

$$L_{\pi_\theta}(\pi_{\theta'}) = J(\theta) + \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi_\theta}, a \sim \pi_\theta(\cdot|s)} \left[\frac{\pi_{\theta'}(a|s)}{\pi_\theta(a|s)} A^{\pi_\theta}(s, a) \right]$$

Often we drop the $J(\theta)$ and $(1-\gamma)^{-1}$ since they don't impact the optimization. Notably $\nabla_\theta J(\theta) = \nabla_{\theta'} L_{\pi_\theta}(\pi_{\theta'})|_{\theta'=\theta}$, so $L_{\pi_\theta}(\pi_{\theta'})$ and $J(\theta')$ agree to first-order. If the update returns a *mixture policy* $\pi_{\theta'} = (1-\alpha^*)\pi_\theta + \alpha^* \operatorname{argmax}_\pi L_{\pi_\theta}(\pi)$ for some optimal mixing parameter $\alpha^* \in [0, 1]$ defined by

$$\alpha^* = \operatorname{argmax}_\alpha \left(\alpha \mathbb{E}_{s \sim d^{\pi_\theta}, a \sim \pi_{\theta'}(\cdot|s)} \left[\frac{\pi_{\theta'}(a|s)}{\pi_\theta(a|s)} A^{\pi_\theta}(s, a) \right] - \frac{2\gamma\varepsilon\alpha^2}{(1-\gamma)^2} \right)$$

where $\varepsilon = \max_s |\mathbb{E}_{a \sim \pi_{\theta'}(\cdot|s)} [A^{\pi_\theta}(s, a)]|$, then the *CPI lower bound* is satisfied,

$$J(\theta') \geq L_{\pi_\theta}(\pi_{\theta'}) - \frac{2\gamma\varepsilon\alpha^2}{(1-\gamma)^2} = J(\theta) + \alpha \mathbb{E}_{s \sim d^{\pi_\theta}, a \sim \pi_\theta(\cdot|s)} \left[\frac{\pi_{\theta'}(a|s)}{\pi_\theta(a|s)} A^{\pi_\theta}(s, a) \right] - \frac{2\gamma\varepsilon\alpha^2}{(1-\gamma)^2}$$

Thus for small enough α , monotonic improvement is guaranteed, since the performance difference is bounded below by $c_1\alpha - c_2\alpha^2$.

Remark. **Notes.** Since maximizing the objective is equivalent to maximizing the performance difference, let's compare this quantity with the surrogate: by importance sampling,

$$J(\theta') - J(\theta) = \mathbb{E}_{s \sim d^{\pi_{\theta'}}, a \sim \pi'_{\theta}(\cdot | s)} [A^{\pi_{\theta}}(s, a)] = \mathbb{E}_{s \sim d^{\pi_{\theta}}, a \sim \pi_{\theta}(\cdot | s)} \left[\frac{d^{\pi_{\theta'}}(s)}{d^{\pi_{\theta}}(s)} \frac{\pi_{\theta'}(a | s)}{\pi_{\theta}(a | s)} A^{\pi_{\theta}}(s, a) \right]$$

So, the surrogate is making the approximation $\frac{d^{\pi_{\theta'}}(s)}{d^{\pi_{\theta}}(s)} \approx 1$.

Issues. The two largest problems are that monotonic improvement is only guaranteed for mixture policies (which deep neural networks violate) and that the constant c_2 is extraordinarily large, thus requiring a massive surrogate objective to guarantee positive improvement. ■

Trust Region Policy Optimization (Schulman et. al., 2015)

TRPO provides a *theory of monotonic improvement for general policy update rules*. It frames the problem as a constrained optimization, building on the ideas of CPI and NPG:

$$\begin{aligned} & \underset{\theta'}{\text{maximize}} \quad \tilde{L}_{\pi_{\theta}}(\pi_{\theta'}) = \mathbb{E}_{s \sim d^{\pi_{\theta}}, a \sim \pi_{\theta}(\cdot | s)} \left[\frac{\pi_{\theta'}(a | s)}{\pi_{\theta}(a | s)} A^{\pi_{\theta}}(s, a) \right] \\ & \text{subject to} \quad \tilde{\mathbb{D}}_{\text{KL}}[\pi_{\theta} \| \pi_{\theta'}] \leq \delta \end{aligned}$$

As with natural policy gradients, a first-order approximation reduces the problem to

$$\begin{aligned} & \underset{\Delta\theta}{\text{maximize}} \quad \underbrace{(\nabla_{\theta'} L_{\pi_{\theta}}(\pi_{\theta'})|_{\theta'=\theta})^\top \Delta\theta}_{g_{\theta}} \\ & \text{subject to} \quad \frac{1}{2} \Delta\theta^\top \mathcal{I}(\pi_{\theta}) \Delta\theta \leq \delta \end{aligned}$$

The ideal direction is $s_{\theta} = \mathcal{I}(\pi_{\theta})^{-1} g_{\theta}$ which is computed by the conjugate-gradient method (requiring only $\mathcal{O}(|\theta|)$ per matvec thanks to **Pearlmutter's trick!**). The step size is computed by backtracking line search, starting from the natural policy gradient step size (the optimal size assuming the conjugate-gradient solution and first-order approximation are exact):

$$\theta' = \theta + \alpha^j \sqrt{\frac{2\delta}{s_{\theta}^\top \mathcal{I}(\pi_{\theta}) s_{\theta}}} s_{\theta} = \theta + \alpha^j \sqrt{\frac{2\delta}{g_{\theta}^\top s_{\theta}}} s_{\theta}$$

where $j \in \{0, \dots, J\}$ is the smallest positive integer such that both the *performance difference is positive* and the *KL divergence constraint is satisfied*. If no such $j \leq J$ exists, then the policy remains constant while more data is collected at the next iteration. (For the analysis, take $J = \infty$). TRPO guarantees that

$$J(\theta') \geq L_{\pi_{\theta}}(\pi_{\theta'}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} \max_{s \in \mathcal{S}} \mathbb{D}_{\text{KL}}[\pi_{\theta}(\cdot | s) \| \pi_{\theta'}(\cdot | s)]$$

Notably, $\pi_{\theta'}$ does *not* have to be a mixture policy.

Remark. **Notes.** Pearlmutter's trick computes Hessian-vector products using only two forward/back-

ward passes and thus in $\mathcal{O}(n)$ time! Suppose $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $v \in \mathbb{R}^n$; we seek $\nabla^2 f(x)v$. Note

$$\nabla f(x)^\top v = \sum_{j=1}^n \frac{\partial f}{\partial x_j} v_j \implies \nabla(\nabla f(x)^\top v) = \left(\sum_{j=1}^n \frac{\partial^2 f}{\partial x_i \partial x_j} v_j \right)_{1 \leq i \leq n} = \nabla^2 f(x)v$$

So, we require only two gradient operations to compute the product, making each iteration of conjugate gradient $\mathcal{O}(n)$.

Issues. There are three salient issues.

- The quadratic approximation of the KL divergence is still fragile; if the geometry is highly nonlinear, line search may repeatedly reject updates, leading to stalled training.
- Backtracking line search and conjugate gradient (even with Pearlmutter's trick) are computationally expensive.
- Actor-critic methods, which share parameters between π_θ and V^{π_θ} , cannot be used since the actor (policy network) is updated by a constrained optimization, while the critic (value network) is unconstrained. So, even mixing a value update with a policy update can violently upset the local geometry.

As it turns out, proximal policy optimization will abandon the second-order Fisher information matrix computation and the hard KL constraint, addressing these issues. ■

2.3 Actor-Critic Methods

Generalized Advantage Estimation (Schulman et. al., 2016)

GAE provides a family of advantage estimators that interpolate between bootstrapped and Monte Carlo estimates. Define the **temporal difference residual** for a value function V_ϕ as

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

Notably, δ_t is an unbiased one-step estimate of the advantage:

$$\mathbb{E}_{\tau \sim \pi_\theta} [\delta_t | s_t, a_t] = A^{\pi_\theta}(s_t, a_t)$$

The *generalized advantage estimate* is an exponentially-weighted average of multi-step advantage estimates:

$$\hat{A}_t = \sum_{t'=t}^T (\gamma \lambda)^{t'-t} \delta_{t'} \quad (\lambda \in [0, 1])$$

where λ is the *bias-variance tradeoff parameter*, and conventionally $0^0 = 1$. Notably, if $\lambda = 0$, then $\hat{A}_t = \delta_t$, the one-step TD residual; if $\lambda = 1$, then $\hat{A}_t = R_t(\tau) - V(s_t)$, the Monte Carlo advantage. While GAE is mostly an empirical method, one theoretical guarantee is that if $\|V_\phi - V^{\pi_\theta}\|_\infty \leq \epsilon$, then the bias is bounded by an expression that vanishes as $\lambda \rightarrow 1$:

$$\left| \mathbb{E}_{\tau \sim \pi_\theta} [\hat{A}_t - A^{\pi_\theta}(s_t, a_t)] \right| \leq \frac{2\epsilon\gamma(1-\lambda)}{1-\gamma\lambda}$$

Actor-Critic

An actor-critic architecture maintains two components:

- **Actor** π_θ : the policy network, updated by policy gradient methods.
- **Critic** V_ϕ : a value function approximator with parameters ϕ , trained by regression on observed returns.

The critic provides advantage estimates (e.g., via GAE) that are used in place of Monte Carlo returns in the policy gradient,

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T \gamma^t \hat{A}_t \nabla_\theta \log \pi_\theta(a_t | s_t) \right]$$

The critic is updated by minimizing the squared error loss

$$\mathcal{L}(V_\phi) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T (V_\phi(s_t) - R_t(\tau))^2 \right]$$

Unlike REINFORCE, the actor-critic framework allows for lower-variance gradient estimates at the cost of bias introduced by the learned value function.

Remark. **Notes.** The baseline subtraction used in REINFORCE can be viewed as a special case: using $b(s) = V^{\pi_\theta}(s)$ gives the advantage $A^{\pi_\theta}(s, a) = Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s)$, which is exactly what the critic estimates. The key distinction is that in actor-critic, the critic is actively learned and used for bootstrapping (via TD residuals), not just as a passive baseline.

Issues. When parameters are shared between the actor and critic (e.g. in a common feature extractor), constrained policy updates like TRPO become problematic: the trust region governs the actor, but the critic's unconstrained regression updates can perturb the shared representations, violating the local geometry assumptions. PPO's clipping mechanism, as we will see, is far more amenable to shared architectures. ■

2.4 Proximal Policy Optimization

Proximal Policy Optimization (Schulman et. al., 2017)

PPO replaces TRPO's constrained optimization with a *clipped surrogate objective* that implicitly enforces a trust region, achieving comparable or superior empirical performance at dramatically lower implementation and computational cost. Define the probability ratio

$$\rho_t(\theta') = \frac{\pi_{\theta'}(a_t | s_t)}{\pi_\theta(a_t | s_t)}$$

The actor loss is the **PPO-Clip** objective; for some clipping hyperparameter $\varepsilon > 0$,

$$\mathcal{L}^{\text{CLIP}}(\theta') = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T \min \left(\rho_t(\theta') \hat{A}_t, \text{clip}(\rho_t(\theta'), 1 - \varepsilon, 1 + \varepsilon) \hat{A}_t \right) \right]$$

The critic loss is the usual sum-of-squares error of the value function,

$$\mathcal{L}^{\text{VF}}(\phi) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T (V_\phi(s_t) - R_t(\tau))^2 \right]$$

Finally, we define the **entropy bonus for exploration**,

$$\mathcal{H}(\theta') = \mathbb{E}_{s \sim d^{\pi_\theta}} [H(\pi_{\theta'}(\cdot | s))]$$

The full PPO loss, combining the actor, critic, and an entropy bonus for exploration, is

$$\mathcal{L}^{\text{PPO}}(\theta', \phi) = -\mathcal{L}^{\text{CLIP}}(\theta') + c_1 \mathcal{L}^{\text{VF}}(\phi) - c_2 \mathcal{H}(\theta')$$

for $\alpha, \beta > 0$ weighting coefficients.

Remark. Mechanics of Clipping. By case analysis on the sign of \hat{A}_t :

- If $\hat{A}_t > 0$ (action was better than expected): increasing ρ_t improves the surrogate up to a factor of $1 + \varepsilon$. The minimum selects the more conservative term, so

$$\min(\rho_t \hat{A}_t, (1 + \varepsilon) \hat{A}_t) = \min(\rho_t, 1 + \varepsilon) \hat{A}_t$$

preventing the policy from moving too aggressively toward good actions.

- If $\hat{A}_t < 0$ (action was worse than expected): decreasing ρ_t improves the surrogate up to a factor of $1 - \varepsilon$. Here,

$$\min(\rho_t \hat{A}_t, (1 - \varepsilon) \hat{A}_t) = \max(\rho_t, 1 - \varepsilon) \hat{A}_t$$

preventing the policy from moving too far from bad actions.

In both cases, the gradient is zero outside the trust region $\rho_t \in [1 - \varepsilon, 1 + \varepsilon]$, providing a “soft wall” analogous to TRPO’s hard KL constraint. Crucially, the clipped objective is a *pessimistic* lower bound on the unclipped surrogate, so PPO never exploits overconfident advantage estimates.

Notes. We discuss several important details.

- *Minibatch updates.* Unlike TRPO (which performs a single constrained step per batch of data), PPO performs multiple epochs of minibatch SGD on the same batch of trajectories. This is the primary source of its sample efficiency gains. Clipping ensures that each epoch does not stray too far from the data-collecting policy, even without an explicit constraint.
- *Entropy bonus.* The term $c_2 \mathcal{H}(\pi_{\theta'})$ encourages exploration by penalizing deterministic policies. Without it, the policy can prematurely converge to a suboptimal deterministic action. This is especially important in environments with sparse or deceptive rewards.
- *Value function clipping.* In some implementations (e.g., the original PPO paper), the value function loss is also clipped:

$$\mathcal{L}^{\text{VF,clip}}(\phi) = \mathbb{E} [\max ((V_\phi(s_t) - R_t)^2, (V_{\phi_{\text{old}}}(s_t) + \text{clip}(\Delta V, -\varepsilon, \varepsilon) - R_t)^2)]$$

where $\Delta V = V_\phi(s_t) - V_{\phi_{\text{old}}}(s_t)$. However, subsequent empirical studies have found this can hurt performance, and it is often omitted.

- *Shared parameters.* Since both the actor and critic losses are unconstrained and differentiable, they can be combined into a single loss function and optimized jointly on a shared network backbone. This resolves the fundamental incompatibility between TRPO and actor-critic architectures noted earlier.

Comparison with TRPO. The core simplification is summarized as follows:

	TRPO	PPO-Clip
Constraint	Hard KL divergence	Implicit via clipping
Optimization	Conjugate gradient + line search	Minibatch SGD
Second-order	Fisher information matrix	Not required
Updates per batch	1	Multiple epochs
Actor-Critic	Incompatible (shared params)	Compatible

PPO achieves competitive or better empirical results by trading theoretical guarantees for practical robustness. Unlike TRPO, PPO does not come with a formal monotonic improvement bound; the clipping is a heuristic that works well in practice. ■

PPO-Clip Algorithm

The full algorithm operates as follows:

PPO-Clip Algorithm

```

1: Initialize policy parameters  $\theta$ , value function parameters  $\phi$ 
2: Initialize hyperparameters: learning rate  $\alpha$ , clip range  $\varepsilon$ , GAE parameter  $\lambda$ , number of
   epochs  $K$ , minibatch size  $M$ 
3: for  $iteration = 0, \dots, \text{MAX\_ITER}$  do
4:   Collect a batch of trajectories  $\mathcal{D} = \{\tau_i\}$  using  $\pi_\theta$ 
5:   Compute GAE advantages  $\hat{A}_t$  for all  $(s_t, a_t) \in \mathcal{D}$  using  $V_\phi$ 
6:   Normalize advantages:  $\hat{A}_t \leftarrow (\hat{A}_t - \mu_{\hat{A}})/\sigma_{\hat{A}}$  over the batch
7:   for  $epoch = 1, \dots, K$  do
8:     for each minibatch  $\mathcal{B} \subset \mathcal{D}$  of size  $M$  do
9:       Compute ratios  $\rho_t(\theta') = \pi_{\theta'}(a_t | s_t) / \pi_\theta(a_t | s_t)$  for all  $t \in \mathcal{B}$ 
10:      Compute clipped surrogate:  $\mathcal{L}^{\text{CLIP}} = \frac{1}{|\mathcal{B}|} \sum_{t \in \mathcal{B}} \min(\rho_t \hat{A}_t, \text{clip}(\rho_t, 1-\varepsilon, 1+\varepsilon) \hat{A}_t)$ 
11:      Compute value loss:  $\mathcal{L}^{\text{VF}} = \frac{1}{|\mathcal{B}|} \sum_{t \in \mathcal{B}} (V_\phi(s_t) - R_t)^2$ 
12:      Compute entropy bonus:  $\mathcal{H} = \frac{1}{|\mathcal{B}|} \sum_{t \in \mathcal{B}} H(\pi_{\theta'}(\cdot | s_t))$ 
13:       $(\theta', \phi) \leftarrow (\theta', \phi) - \alpha \nabla_{(\theta', \phi)} (-\mathcal{L}^{\text{CLIP}} + c_1 \mathcal{L}^{\text{VF}} - c_2 \mathcal{H})$ 
14:    end for
15:  end for
16:   $\theta \leftarrow \theta'$ 
17: end for

```

Remark. **Notes.** Advantage normalization (Line 6) is a critical implementation detail: it centers the advantages to have zero mean and unit variance across the batch, ensuring that roughly half the actions are reinforced and half are suppressed, regardless of the absolute scale of returns. This is conceptually related to the average reward baseline discussed in REINFORCE.

Standard hyperparameters (from the original paper and subsequent best practices) are $\varepsilon = 0.2$, $\lambda = 0.95$, $\gamma = 0.99$, $K = 3\text{--}10$ epochs, and $c_1 = 0.5$, $c_2 = 0.01$. The number of epochs K must be tuned carefully: too few epochs waste data, while too many can degrade performance as the ratio ρ_t drifts and the advantage estimates become stale.

PPO-Penalty. An alternative formulation replaces clipping with a soft KL penalty,

$$\mathcal{L}^{\text{KL}}(\theta') = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T \rho_t(\theta') \hat{\mathbb{A}}_t \right] - \beta \tilde{\mathbb{D}}_{\text{KL}}[\pi_\theta \parallel \pi_{\theta'}]$$

where $\beta > 0$ is an adaptive penalty coefficient. When the KL divergence exceeds 1.5δ , β is doubled; when it falls below $\delta/1.5$, β is halved. While this variant is closer to TRPO in spirit (directly penalizing distributional shift), PPO-Clip generally outperforms it empirically, suggesting that the pessimistic clipping mechanism is a more robust proxy for trust region enforcement than a tuned penalty. ■