# Contents

**Part I**

# Numerical Analysis

# 1 Rootfinding

# 2 Factorizations

# 3 Eigenvalue Computation

# 4 Exploiting Sparsity

# 5 Interpolation and Quadrature

# Part II

# Nonconvex Optimization

## 6 Unconstrained Optimization

**First-Order Necessary Conditions**

If $x^*$ is a local minimizer and $f \in C^1$ locally, then $\nabla f(x^*) = 0$; in other words, any local minimizer is a *stationary point*.

**Second-Order Necessary Conditions**

If $x^*$ is a local minimizer and $f \in C^2$ locally, then $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*) \succcurlyeq 0$.

**Second-Order Sufficient Conditions**

If $f \in C^2$ locally, $\nabla f(x^*) = 0$, and $\nabla^2 f(x^*) \succ 0$, then $x^*$ is a *strict* local minimizer.

*Remark.* The proofs of all three preceeding conditions are direct by Taylor expansion. ■

# 7 Line-Search Methods

**Line-Search Methods**

Each iteration of a line search method computes a search direction $p_k$ and then updates via $x_{k+1} = x_k + \alpha_k p_k$, where $\alpha_k > 0$ is the *step size*.

- Most algorithms require $p_k$ to be a descent direction ($p_k^\top \nabla f(x_k) < 0$) for convergence.
- The search direction usually has the form $p_k = -B_k^{-1} \nabla f(x_k)$ for an appropriate $B_k \succ 0$ symmetric and nonsingular.

For example, $B_k = I$ in gradient descent while $B_k = \nabla^2 f(x_k)$ in Newton's method.

TODO: Nocedal and Wright, *Numerical Optimization*.

# 8 Trust-Region Methods

**Trust-Region Methods**

# 9 Optimizers for Deep Learning

## 9.1 Gradient Descent

**Minibatch Stochastic Gradient Descent**

For $f : \mathbb{R}^d \to \mathbb{R}$ differentiable, stochastic gradient descent updates $w_k = w_{k-1} - \alpha \widehat{\nabla} f(w_k)$ for some $\alpha > 0$. TODO. Learning rate schedulers, preconditioning, sampling with/without replacement, batch sizing, etc.

## 9.2 Momentum

**Momentum**

Momentum updates as follows, where $g_k$ can be conceptualized as "negative velocity":

$$\begin{aligned} g_k &= \beta g_{k-1} - \nabla f(w_{k-1}) \\ w_k &= w_{k-1} - \alpha g_{k-1} \end{aligned}$$

It is equivalent to *heavy-ball momentum*, $w_k = w_{k-1} - \alpha \nabla f(w_{k-1}) + \beta(w_{k-1} - w_{k-2})$, and reduces to gradient descent when $\beta = 0$. Intuitively, in regions of low curvature, momentum mitigates stalled training, while in regions of high curvature, it smooths out fluctuations.

*Remark.* Algebraic equivalence to the heavy-ball form follows from taking

$$w_k = w_{k-1} - \alpha g_k \implies g_k = \frac{w_{k-1} - w_k}{\alpha},\ g_{k-1} = \frac{w_{k-2} - w_{k-1}}{\alpha}$$

from which it follows

$$\begin{aligned} g_k = \beta g_{k-1} - \nabla f(w_{k-1}) &\implies \frac{w_{k-1} - w_k}{\alpha} = \beta \frac{w_{k-2} - w_{k-1}}{\alpha} - \nabla f(w_{k-1}) \\ &\implies w_k = w_{k-1} - \alpha \nabla f(w_{k-1}) + \beta(w_{k-1} - w_{k-2}) \end{aligned}$$

which is precisely the heavy-ball update. ■

*Remark.* The key insights of momentum are that it increases the convergence window for the step size $\alpha$ while speeding up convergence of ill-conditioned problems. The simplest nontrivial model is the convex quadratic, $f(w) = \frac{1}{2} w^\top A w - b^\top w$ for $A \succcurlyeq 0$ real-symmetric. Let's compare gradient descent and momentum. For gradient descent,

$$w_k = w_{k-1} - \alpha \nabla f(w_{k-1}) = w_{k-1} - \alpha A(w_{k-1} - w^*)$$

for unique minimizer $w^* = A^{-1} b$. Let $A = Q \Lambda Q^\top$ be the spectral decomposition with eigenvalues $\lambda_1 \geqslant \cdots \geqslant \lambda_n > 0$. Under a change of basis $v_k = Q^\top (w_k - w^*)$,

$$v_k = v_{k-1} - \alpha \Lambda Q^\top (w_{k-1} - w^*) = v_{k-1} - \alpha \Lambda v_{k-1} \implies v_k = (I - \alpha \Lambda)^k v_0$$

The convergence rate is therefore $\rho = \max_i |1 - \alpha \lambda_i| = \max\{|1 - \alpha \lambda_1|, |1 - \alpha \lambda_n|\}$. Notably, convergence occurs whenever $\rho < 1$, i.e. when $\alpha \lambda_1 < 2$. By a perturbation argument (if the rates are

not equal, you can decrease the larger one and increase the smaller one, thereby decreasing the maximum), the optimal step size $\alpha^*$ comes from equating the two rates,

$$|1-\alpha^*\lambda_1| = |1-\alpha^*\lambda_n| \implies 1-\alpha^*\lambda_1 = -1+\alpha^*\lambda_n \implies \alpha^* = \frac{2}{\lambda_1+\lambda_n},\ \rho^* = \frac{\kappa-1}{\kappa+1}$$

where $\kappa = \lambda_1/\lambda_n \geqslant 1$ is the condition number of $A$. In comparison, consider the momentum updates under a similar change of basis $v_k = Q^\top g_k$ and $x_k = Q^\top(w_k - w^*)$,

$$\begin{aligned} v_k &= \beta v_{k-1} + \Lambda x_{k-1} \\ x_k &= x_{k-1} - \alpha v_k = (I-\alpha\Lambda)x_{k-1} - \alpha\beta v_{k-1} \end{aligned}$$

This relation can be expressed as a linear recurrence,

$$\begin{bmatrix} v_k \\ x_k \end{bmatrix} = \begin{bmatrix} \beta I & \Lambda \\ -\alpha\beta I & I-\alpha\Lambda \end{bmatrix} \begin{bmatrix} v_{k-1} \\ x_{k-1} \end{bmatrix} \implies \begin{bmatrix} v_k(i) \\ x_k(i) \end{bmatrix} = \underbrace{\begin{bmatrix} \beta & \lambda_i \\ -\alpha\beta & 1-\alpha\lambda_i \end{bmatrix}}_{R_i} \begin{bmatrix} v_{k-1}(i) \\ x_{k-1}(i) \end{bmatrix} = R_i^k \begin{bmatrix} v_0(i) \\ x_0(i) \end{bmatrix}$$

In other words, the convergence in the $i^{\text{th}}$ eigenspace is governed by the spectral radius $r_i$ of $R_i$, whose characteristic polynomial and spectrum are, respectively,

$$\begin{aligned} \chi_i(z) &= z^2 - (1-\alpha\lambda_i+\beta)z + \beta \\ \sigma_i &= \frac{1}{2}\left(1-\alpha\lambda_i+\beta \pm \sqrt{(1-\alpha\lambda_i+\beta)^2 - 4\beta}\right) \end{aligned}$$

Consider the discriminant $D_i = (1-\alpha\lambda_i+\beta)^2 - 4\beta$. If $D_i \leqslant 0$, then $\sqrt{D_i}$ is purely imaginary, so the spectrum is a singleton with magnitude $r_i = \sqrt{\beta} < 1$, and convergence always occurs. On the other hand if $D_i > 0$, the convergence criterion is $r_i < 1$; equivalently $\chi_i(z)$ must have roots inside the open unit disk. Checking boundary conditions $\chi_i(\pm 1) = 0$, we find $\alpha\lambda_i = 0$ and $\alpha\lambda_i = 2+2\beta$; the former trivially requires $\alpha > 0$, but the latter shows convergence for $\alpha\lambda_i < 2+2\beta$. Comparison with the $\alpha\lambda_i < 2$ bound for gradient descent shows that *momentum nearly doubles the effective window of convergence* whenever $\beta$ is large!

Now we seek the optimal global convergence rate $\rho^*$, for which we must choose $\alpha^*$ and $\beta^*$. By a complex-valued analog to perturbation argument from gradient descent, the maximum convergence rate is minimized when

$$D_i \leqslant 0 \iff -2\sqrt{\beta} \leqslant 1-\alpha\lambda_i+\beta \leqslant 2\sqrt{\beta} \iff \lambda_i \in \left[\frac{(1-\sqrt{\beta})^2}{\alpha}, \frac{(1+\sqrt{\beta})^2}{\alpha}\right]$$

The optimum occurs when this window is just large enough, i.e.

$$\lambda_1 = \frac{(1+\sqrt{\beta^*})^2}{\alpha^*},\ \lambda_n = \frac{(1-\sqrt{\beta^*})^2}{\alpha^*} \implies \beta^* = \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^2,\ \alpha^* = \left(\frac{2}{\sqrt{\lambda_1}+\sqrt{\lambda_n}}\right)^2$$

and the global convergence rate follows from $\rho^* = \sqrt{\beta^*}$. The insight is: *for ill-conditioned problems ($\kappa \gg 1$), momentum effectively square-roots the condition number* compared to gradient descent. ■

## Part III

# Convex Optimization

### Strong Convexity

Let $f : K \to \mathbb{R}$ differentiable for $K \subseteq \mathbb{R}^n$. Then $f$ is $\alpha$-strongly convex if it has at least some curvature everywhere:

$$f(y) \geqslant f(x) + \nabla f(x)^\top (y - x) + \frac{1}{2}\alpha \|y - x\|^2$$

Informally, the function cannot be "too flat" anywhere, which is useful since small sub-optimalities in output-space then automatically translate into to small suboptimalities in input-space: $f(x) - f(x^*) \geqslant \alpha \|x^* - x\|^2/2$. If $f \in C^2$, an equivalent characterization is that the spectrum of the Hessian is bounded from below by $\alpha$, i.e. $\nabla^2 f(x) \geqslant \alpha I$.

### Smoothness

Let $f : K \to \mathbb{R}$ differentiable for $K \subseteq \mathbb{R}^n$. Then $f$ is $\beta$-smooth if its curvature is not too sharp anywhere:

$$f(y) \leqslant f(x) + \nabla f(x)^\top (y - x) + \frac{1}{2}\beta \|y - x\|^2$$

Equivalently, its gradient is $\beta$-Lipschitz, $\|\nabla f(y) - \nabla f(x)\| \leqslant \beta \|y - x\|$. Similarly, if $f \in C^2$, another characterization is $\nabla^2 f(x) \leqslant \beta I$.

*Remark.* If $f : K \to \mathbb{R}$ is $\alpha$-strongly convex *and* $\beta$-smooth, then $\alpha I \preccurlyeq \nabla^2 f(x) \preccurlyeq \beta I$ so we have a local characterization of the function's spectrum. In this case we denote $\kappa = \beta/\alpha \geqslant 1$ the **condition number** of $f$, analogous to the matrix condition number $\kappa = \lambda_{\max}(A)/\lambda_{\min}(A)$ for $A > 0$. In fact, when $f(x) = \frac{1}{2}x^\top Ax$ for $A > 0$, these definitions agree. ■

### Euclidean Projection

If $K \subseteq \mathbb{R}^n$ is convex and closed, then it has a Euclidean projection

$$\Pi_K(x) = \operatorname*{argmin}_{y \in K} \|y - x\|$$

The generalized Pythagorean theorem states that for such a set $K$,

$$\|y - x\| \geqslant \|\|y - \Pi_K(x)\| \quad (y \in K, x \in \mathbb{R}^n)$$

# 10 Linear Programming

# 11 Convex Programming

# 12 Online Convex Optimization

**The OCO Protocol**

The following defines the OCO protocol, which can be framed as a repeated game between a *decision maker* and an *adversary*. Notably, the losses may be adversarially selected for each round after knowing the decision maker's action.

**Online Convex Optimization Protocol**

1: Fix $\mathcal{K}$ convex and compact.
2: **for** $t = 1, \dots, T$ **do**
3: Decision maker selects action $x_t \in \mathcal{K}$ according to algorithm $\mathcal{A}$.
4: Adversary reveals loss function $f_t : \mathcal{K} \to \mathbb{R}$.
5: Decision maker incurs loss $f_t(x_t)$.
6: **end for**

Because the losses are adversarial, there is no meaningful way to absolutely bound the cumulative loss of the decision maker. Instead, the relevant metric of performance is the performance against the best fixed action in hindsight, i.e. the *regret* function

$$\text{Regret}_T(\mathcal{A}) = \sum_{t=1}^{T} f_t(x_t) - \min_{x \in \mathcal{K}} \sum_{t=1}^{T} f_t(x) = \sum_{t=1}^{T} [f_t(x_t) - f_t(x^*)]$$

for $x^* \in \operatorname{argmin}_{x \in \mathcal{K}} \sum_t f_t(x)$. The goal is *sublinear regret* so that the average regret vanishes asymptotically, i.e. $\frac{1}{T}\text{Regret}_T(\mathcal{A}) \to 0$ as $T \to \infty$.

*Remark.* For this framework to make sense, several assumptions must be made:

- The adversary's losses must be uniformly bounded in time; otherwise, (s)he can assign an arbitrarily large loss to the decision-maker's first action (and provide zero loss to all other actions), making it impossible to bound regret from above.
- Compactness, with some regularity assumptions on $f_t$, prevents the adversary from setting aside some strategies with zero loss while increasing the decision-maker's losses indefinitely.

Surprisingly, much can be derived with not much more than these restrictions. ■

## 12.1 Prediction from Expert Advice

**The Experts Setting**

In this special case of OCO, $\mathcal{K} = \Delta_n$ is the probability simplex over $n$ experts, each of whom presents a recommendation at time $t$ represented by a basis vector $e_i \in \Delta_n$. Therefore the experts' loss vector is $l_t = (f_t(e_i))_{i=1}^n \in \mathbb{R}^n$.

**Hedge Algorithm**

The Hedge algorithm exponentially downweights the experts each round depending on their latest losses.

**Hedge Algorithm**

Initialize $W_1 = \mathbf{1}_n$.
**for** $t = 1, \ldots, T$ **do**
    Initialize distribution $x_t = \left[\frac{W_t(i)}{\sum_j W_t(j)}\right]_{i=1}^{n}$.
    Decision maker selects expert $i_t \sim x_t$.
    Adversary reveals loss function $f_t$, and DM incurs loss $l_t(i_t)$.
    Update weights $W_{t+1}(i) = W_t(i)\exp(-\eta l_t(i))$.
**end for**

# Part IV
# Reinforcement Learning

## 13 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 $V^\pi : \mathcal{S} \to \mathbb{R}$, while the *action-value function* is $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The functions are defined by ($\tau$ is a trajectory)

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

For simplicity of analysis, we often assume $T = \infty$. Note $\|V^\pi\|_\infty, \|Q^\pi\|_\infty \leqslant \frac{1}{1-\gamma}$, 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 $\mathbb{P}_{(s,a),s'} = \mathbb{P}(s' \mid s, a)$. Let $\Pi_\pi \in \mathbb{R}^{S\times SA}$ be the policy matrix with $(\Pi_\pi)_{s,(s',a)} = \mathbf{1}[s = s']\pi(a \mid s)$, and let $r \in \mathbb{R}^{SA}$ be the reward vector. Then the Bellman consistency equations become

$$Q^\pi = r + \gamma \mathbb{P}\Pi_\pi Q^\pi, \qquad V^\pi = \Pi_\pi Q^\pi$$

Define the *Bellman evaluation operator* $\mathcal{T}^\pi : \mathbb{R}^{SA} \to \mathbb{R}^{SA}$ by $\mathcal{T}^\pi Q = r + \gamma\mathbb{P}\Pi_\pi Q$. Then $Q^\pi$ is the unique fixed point of $\mathcal{T}^\pi$, and since $\mathcal{T}^\pi$ is a $\gamma$-contraction in $\|\cdot\|_\infty$, the solution is

$$Q^\pi = (I - \gamma\mathbb{P}\Pi_\pi)^{-1} r$$

Similarly, the *Bellman optimality operator* $\mathcal{T}^* Q = r + \gamma\mathbb{P}\max_\pi \Pi_\pi Q$ has unique fixed point $Q^*$, and the optimal policy is $\pi^*(s) = \operatorname{argmax}_a Q^*(s,a)$. Value iteration is precisely the repeated application $Q_{k+1} = \mathcal{T}^* Q_k$, which converges at rate $\gamma^k$. ■

## Temporal Difference Learning

TD learning updates the value function using *bootstrapped* estimates rather than waiting for complete trajectory returns. The **TD(0)** update rule is

$$V(s_t) \leftarrow V(s_t) + \alpha \underbrace{(r(s_t, a_t) + \gamma V(s_{t+1}) - V(s_t))}_{\delta_t \text{ (TD error)}}$$

More generally, **TD($\lambda$)** interpolates between TD(0) and Monte Carlo via eligibility traces. Define the $n$-step return

$$G_t^{(n)} = \sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n V(s_{t+n})$$

Then the $\lambda$-return is an exponentially-weighted average of all $n$-step returns,

$$G_t^{\lambda} = (1-\lambda) \sum_{n=1}^{T-t} \lambda^{n-1} G_t^{(n)}$$

TD(0) is biased but low-variance (it bootstraps from $V$, which may be inaccurate), while Monte Carlo ($\lambda = 1$) is unbiased but high-variance. TD methods are *off-policy compatible* and converge under standard stochastic approximation conditions (Robbins-Monro). With function approximation $V_\phi$, the update becomes semi-gradient:

$$\phi \leftarrow \phi + \alpha \delta_t \nabla_\phi V_\phi(s_t)$$

where crucially the gradient is *not* taken through the bootstrap target $V_\phi(s_{t+1})$.

## SARSA and Expected SARSA

**SARSA** (State-Action-Reward-State-Action) is the on-policy analogue of TD(0) for the action-value function. At each step, the agent observes $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$ and updates

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right)$$

Since $a_{t+1} \sim \pi(\cdot \mid s_{t+1})$, SARSA evaluates the *behavior policy* including exploration (e.g., $\varepsilon$-greedy), making it on-policy. This causes SARSA to learn more conservative policies in stochastic environments, since it accounts for the cost of exploratory actions.

**Expected SARSA** replaces the sampled next action with an expectation over the policy,

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_t + \gamma \mathbb{E}_{a' \sim \pi(\cdot|s_{t+1})}[Q(s_{t+1}, a')] - Q(s_t, a_t)\right)$$

This eliminates the variance from sampling $a_{t+1}$ while remaining on-policy. When $\pi$ is greedy, Expected SARSA reduces to Q-learning; when $\pi$ is $\varepsilon$-greedy, it provides a strictly lower-variance alternative to SARSA. In tabular settings, Expected SARSA converges under the same conditions as Q-learning but with empirically faster convergence.

**Deep Q Networks (Mnih et. al., 2015)**

DQN extends Q-learning to high-dimensional state spaces by approximating $Q^*$ with a neural network $Q_\phi$. Two key innovations stabilize training:

- **Experience replay.** Transitions $(s_t, a_t, r_t, s_{t+1})$ are stored in a replay buffer $\mathcal{D}$ and sampled uniformly at random for training. This breaks temporal correlations between consecutive samples and improves data efficiency.
- **Target network.** A separate target network $Q_{\phi^-}$ with parameters $\phi^-$ is used to compute TD targets, with $\phi^-$ updated slowly (either by periodic copying or Polyak averaging $\phi^- \leftarrow \tau\phi + (1-\tau)\phi^-$).

The loss is the mean squared Bellman error over minibatches from the replay buffer,

$$\mathcal{L}(\phi) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(r + \gamma \max_{a'} Q_{\phi^-}(s', a') - Q_\phi(s, a)\right)^2\right]$$

The agent acts $\varepsilon$-greedily with respect to $Q_\phi$, typically with $\varepsilon$ annealed from 1.0 to 0.1.

*Remark.* **Notes.** Without target networks, the TD target $r + \gamma \max_{a'} Q_\phi(s', a')$ moves with every gradient step, creating a "chasing a moving target" phenomenon that causes oscillation or divergence. The target network provides a quasi-stationary objective over many updates.

**Double DQN.** Vanilla DQN suffers from *maximization bias*: the $\max_{a'} Q_{\phi^-}(s', a')$ operator systematically overestimates values since $\mathbb{E}[\max_i X_i] \geqslant \max_i \mathbb{E}[X_i]$. Double DQN (van Hasselt et. al., 2016) decouples action selection from evaluation,

$$y = r + \gamma Q_{\phi^-}(s', \operatorname*{argmax}_{a'} Q_\phi(s', a'))$$

The online network $Q_\phi$ selects the best action, but the target network $Q_{\phi^-}$ evaluates it. This significantly reduces overestimation and improves stability.

**Issues.** DQN is limited to discrete action spaces (the max over actions is intractable for continuous $\mathcal{A}$), does not naturally encourage exploration beyond $\varepsilon$-greedy, and the replay buffer introduces off-policy bias that can compound with function approximation errors (the "deadly triad" of function approximation, bootstrapping, and off-policy learning). ■

**Rainbow (Hessel et. al., 2018)**

Rainbow combines six orthogonal improvements to DQN into a single integrated agent:

- **Double Q-learning**: decouples action selection from evaluation to reduce overestimation bias.
- **Prioritized experience replay**: samples transitions with probability proportional to their TD error magnitude $|\delta_t|$, focusing learning on surprising or poorly-predicted transitions. Importance sampling weights correct for the induced bias.
- **Dueling networks**: decomposes the Q-function as $Q_\phi(s, a) = V_\psi(s) + A_\xi(s, a) - \frac{1}{|\mathcal{A}|}\sum_{a'} A_\xi(s, a')$, allowing the network to separately learn state value and action ad-

vantage. The centering ensures identifiability.

- **Multi-step returns**: replaces the one-step TD target with an $n$-step return $G_t^{(n)} = \sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n \max_{a'} Q_{\phi^-}(s_{t+n}, a')$, propagating reward signal faster at the cost of some off-policy bias.
- **Distributional RL (C51)**: models the full return distribution $Z(s,a)$ rather than its expectation $Q(s,a) = \mathbb{E}[Z(s,a)]$, representing the distribution as a categorical over a fixed set of atoms and minimizing the projected KL divergence to the distributional Bellman target.
- **Noisy networks**: replaces $\varepsilon$-greedy exploration with learned stochastic linear layers $y = (\mu_w + \sigma_w \odot \varepsilon_w)x + \mu_b + \sigma_b \odot \varepsilon_b$, where $\varepsilon$ is sampled noise. The noise parameters $\sigma$ are learned, enabling state-dependent exploration that adapts over training.

The ablation study in the original paper demonstrates that each component contributes positively, with distributional RL and prioritized replay providing the largest individual gains. Rainbow achieved state-of-the-art performance on the Atari-57 benchmark at the time of publication.

# 14 Policy Methods

**Policy Optimization**

Model the policy using a neural network $\pi_\theta : \mathcal{S} \to \mathcal{A}$. 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 $\tau = (s_0, a_0, r_0, \ldots, s_T, a_T, r_T)$ is a trajectory, $R_t(\tau) = \sum_{t' \geqslant t} \gamma^{t'-t} r(s_{t'}, a_{t'})$ is a one-step estimate of $Q^{\pi_\theta}(s_t, a_t)$, and $\mu$ 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 $V^{\pi_\theta}$ 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 $\mathbb{P}(s' \mid s, a)$ are unknown. However, it turns out that

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

where the last equality follows from the identity $Q^{\pi_\theta}(s_t, a_t) = \mathbb{E}_{\tau\sim\pi_\theta}[R_t(\tau) \mid s_t, a_t]$. Notably, $\nabla_\theta \log \pi_\theta(a_t \mid s_t)$ 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 $\mu$ is the initial state distribution,

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

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

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

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:

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

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

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

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

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

which is precisely as desired. ■

**Discounted State Occupancy Measure (DSOM)**

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

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

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

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

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

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

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:

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

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

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

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 $x\sim p_1$.

## 14.1 REINFORCE

**REINFORCE (Williams, 1992)**

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

**REINFORCE Algorithm**

Initialize learning rate $\alpha$
Initialize weights $\theta$ of a policy network $\pi_\theta$
**for** *episode* $= 0, \ldots,$ MAX_EPISODE **do**
  Sample a trajectory $\tau = (s_0, a_0, r_0, \ldots, s_T, a_T, r_T) \sim \pi_\theta$
  Set $\nabla_\theta J(\theta) \leftarrow 0$
  **for** $t = 0, \ldots, T$ **do**
    $R_t(\tau) \leftarrow \sum_{t'=t}^{T}\gamma^{t'-t}r_{t'}$
    $\nabla_\theta J(\theta) \leftarrow \nabla_\theta J(\theta) + R_t(\tau)\nabla_\theta\log\pi_\theta(a_t\mid s_t)$
  **end for**
  $\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$
**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,

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

One option for $b$ is the value function $V^\pi$, 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 moreso 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! ■

## 14.2 Trust Region Policy Optimization

**Fisher Information Matrix**

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

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

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 $\mathcal{I}(\pi_\theta) = \nabla^2_{\theta'} \widetilde{\mathbb{D}}_{\mathrm{KL}}[\pi_\theta \parallel \pi_{\theta'}]|_{\theta'=\theta}$, where

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

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

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

(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 $\theta^i$ 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,

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

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

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

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 $\Delta J(\theta)$:

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

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

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

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 $\theta \leftarrow \theta + \alpha \widetilde{\nabla}_\theta J(\theta)$.

*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 $\theta$ 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 $\theta_i^{-1}$.

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 $\pi_\theta$ 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 $\pi_\theta$-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 $\mu$ 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*,

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

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

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

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

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

Thus for small enough $\alpha$, 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,

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

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 contrained optimization, building on the ideas of CPI and NPG:

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

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

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

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 $j \leqslant J$ exists, then the policy remains constant while more data is collected at the next iteration. (For the analysis, take $J = \infty$). TRPO guarantees that

$$J(\theta') \geqslant L_{\pi_\theta}(\pi_{\theta'}) - \frac{4\varepsilon\gamma}{(1-\gamma)^2}\max_{s\in\mathcal{S}} \mathbb{D}_{\mathrm{KL}}[\pi_\theta(\cdot\mid s) \parallel \pi_{\theta'}(\cdot\mid s)]$$

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 $f : \mathbb{R}^n \to \mathbb{R}$ and $v \in \mathbb{R}^n$; we seek $\nabla^2 f(x)v$. Note

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

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 $\pi_\theta$ and $V^{\pi_\theta}$, 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. ■

### 14.3 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 $V_\phi$ as

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

Notably, $\delta_t$ is an unbiased one-step estimate of the advantage:

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

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

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

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

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

**Actor-Critic**

An actor-critic architecture maintains two components:

- **Actor** $\pi_\theta$: the policy network, updated by policy gradient methods.
- **Critic** $V_\phi$: a value function approximator with parameters $\phi$, 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,

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

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. ■

## 14.4 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

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

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

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

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**,

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

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

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

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

*Remark.* **Mechanics of Clipping.** By case analysis on the sign of $\widehat{\mathbb{A}}_t$:

- If $\widehat{\mathbb{A}}_t > 0$ (action was better than expected): increasing $\rho_t$ improves the surrogate up to a factor of $1+\varepsilon$. The minimum selects the more conservative term, so
$$\min(\rho_t\widehat{\mathbb{A}}_t, (1+\varepsilon)\widehat{\mathbb{A}}_t) = \min(\rho_t, 1+\varepsilon)\widehat{\mathbb{A}}_t$$
preventing the policy from moving too aggressively toward good actions.
- If $\widehat{\mathbb{A}}_t < 0$ (action was worse than expected): decreasing $\rho_t$ improves the surrogate up to a factor of $1-\varepsilon$. Here,
$$\min(\rho_t\widehat{\mathbb{A}}_t, (1-\varepsilon)\widehat{\mathbb{A}}_t) = \max(\rho_t, 1-\varepsilon)\widehat{\mathbb{A}}_t$$
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:
$$\mathcal{L}^{\text{VF,clip}}(\phi) = \mathbb{E}\left[\max\left((V_\phi(s_t) - R_t)^2,\ (V_{\phi_{\text{old}}}(s_t) + \text{clip}(\Delta V, -\varepsilon, \varepsilon) - R_t)^2\right)\right]$$
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* $= 0, \ldots,$ MAX_ITER **do**
4: Collect a batch of trajectories $\mathcal{D} = \{\tau_i\}$ using $\pi_\theta$
5: Compute GAE advantages $\widehat{\mathbb{A}}_t$ for all $(s_t, a_t) \in \mathcal{D}$ using $V_\phi$
6: Normalize advantages: $\widehat{\mathbb{A}}_t \leftarrow (\widehat{\mathbb{A}}_t - \mu_{\widehat{\mathbb{A}}})/\sigma_{\widehat{\mathbb{A}}}$ over the batch
7: **for** *epoch* $= 1, \ldots, K$ **do**
8: **for** each minibatch $\mathcal{B} \subset \mathcal{D}$ of size $M$ **do**
9: Compute ratios $\rho_t(\theta') = \pi_{\theta'}(a_t \mid s_t)/\pi_\theta(a_t \mid s_t)$ for all $t \in \mathcal{B}$
10: Compute clipped surrogate: $\mathcal{L}^{\text{CLIP}} = \frac{1}{|\mathcal{B}|}\sum_{t\in\mathcal{B}} \min(\rho_t \widehat{\mathbb{A}}_t, \text{clip}(\rho_t, 1-\varepsilon, 1+\varepsilon)\widehat{\mathbb{A}}_t)$
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: $\mathcal{H} = \frac{1}{|\mathcal{B}|}\sum_{t\in\mathcal{B}} H(\pi_{\theta'}(\cdot \mid s_t))$
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 $\varepsilon = 0.2$, $\lambda = 0.95$, $\gamma = 0.99$, $K = 3$–$10$ 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 $\rho_t$ drifts and the advantage estimates become stale.

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

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

where $\beta > 0$ is an adaptive penalty coefficient. When the KL divergence exceeds $1.5\delta$, $\beta$ is doubled; when it falls below $\delta/1.5$, $\beta$ 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. ■

### 14.5 Group-Relative Policy Optimization

TODO.