Policy gradient

Simplicity at the cost of variance

Cathy Wu

6.7920: Reinforcement Learning: Foundations and Methods

Readings

1. *Josh Achiam. Spinning Up. Part 3: Intro to Policy Optimization. OpenAI, 2018.*
2. *NDP §6.1: Generic issues – from parameters to policies*
3. *SB Chapter 13: Policy Gradient Methods*

Outline

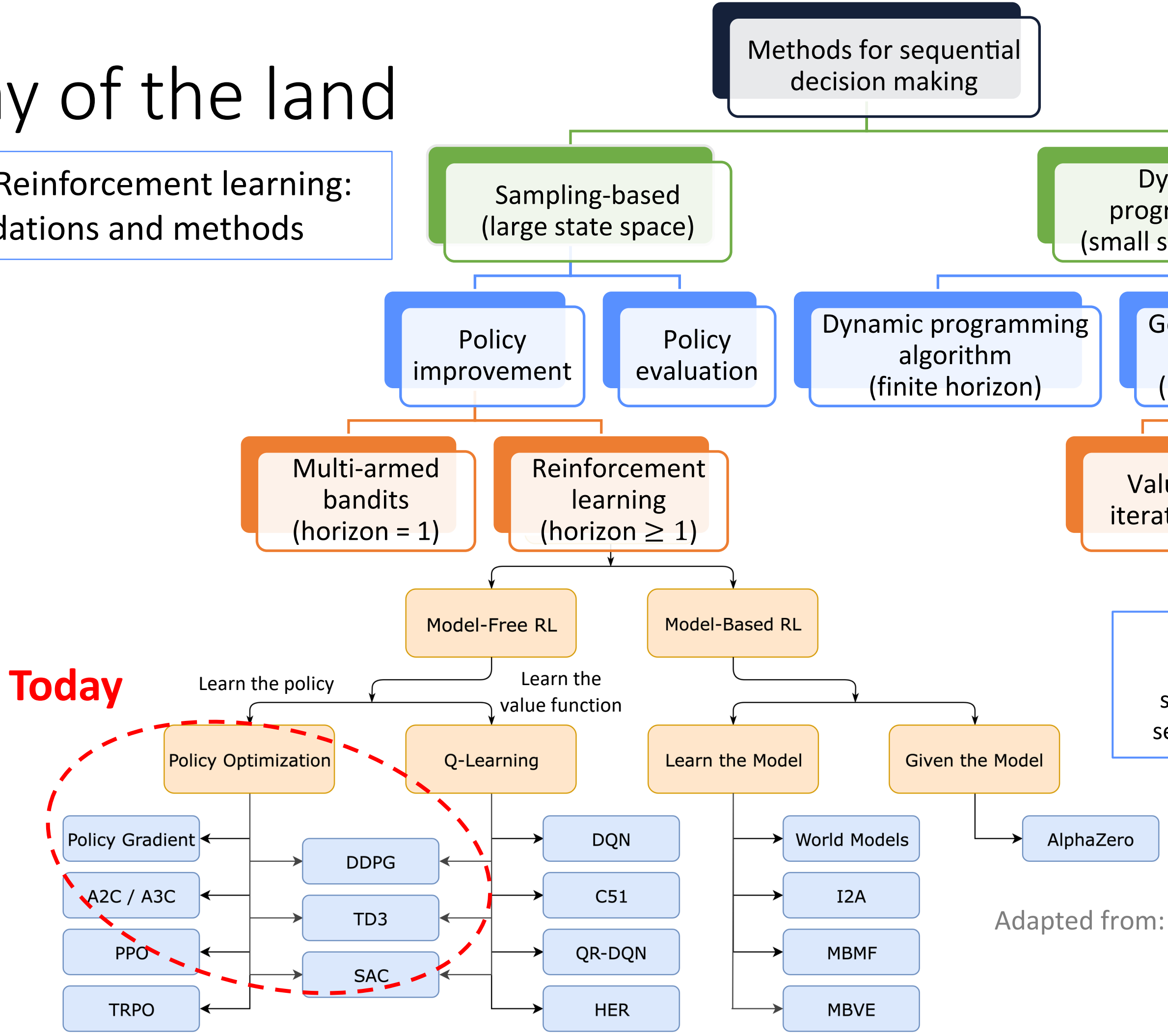
1. From Policy Iteration to Policy Search
2. Policy gradient methods
3. Actor-critic

Outline

1. **From Policy Iteration to Policy Search**
2. Policy gradient methods
3. Actor-critic

Lay of the land

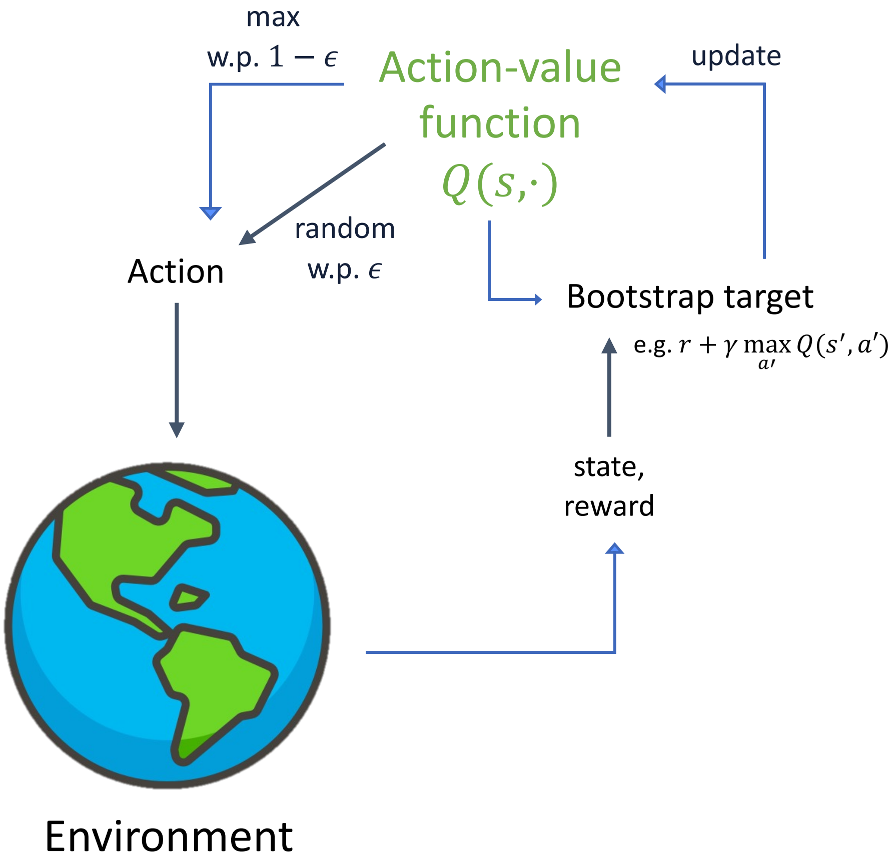
6.7920: Reinforcement learning: foundations and methods



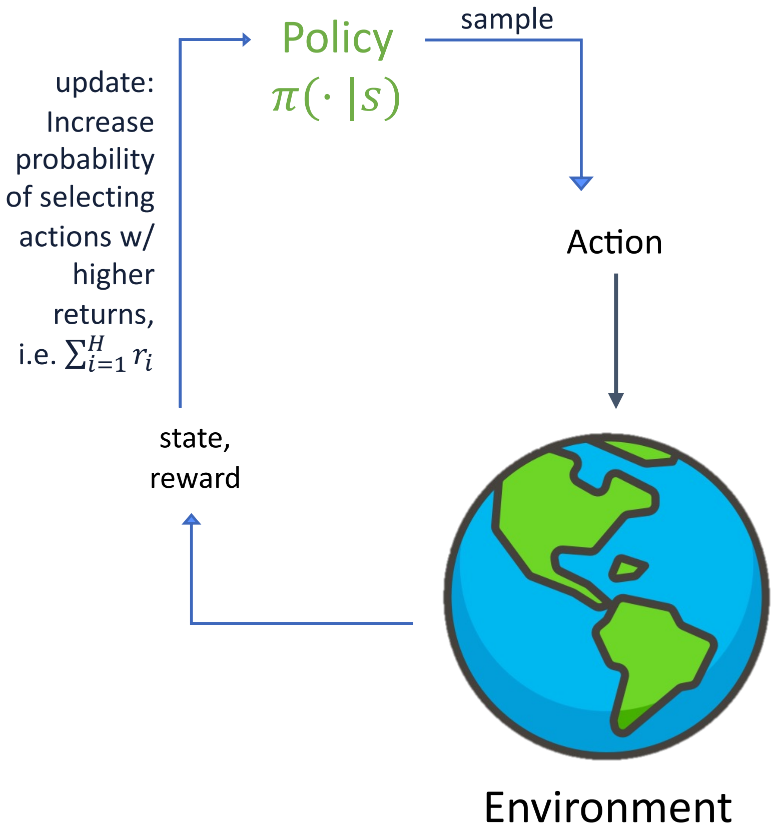
De-facto definition:
Reinforcement learning = sampling-based policy improvement for sequential decision making (horizon ≥ 1)

Adapted from: OpenAI Spinning Up

Value-based methods



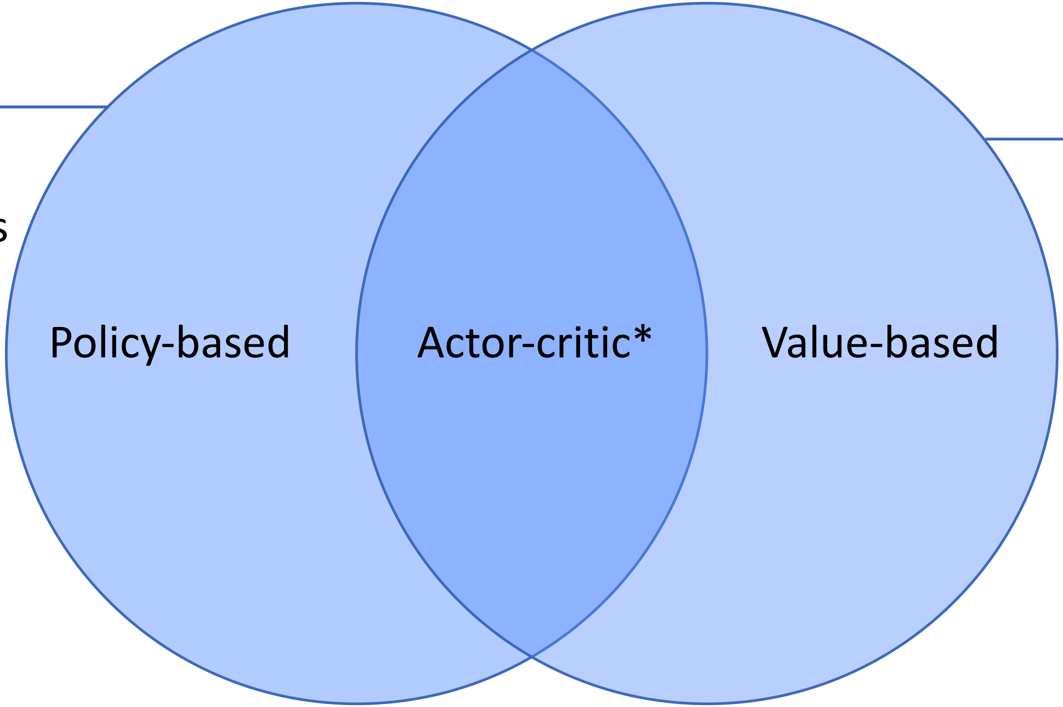
Policy-based methods



RL methods overview

Optimize policies
(and only policies)

Policy gradients
REINFORCE
Genetic algorithms



Optimize value functions
(and only value functions)

Value iteration
Q-learning
SARSA
DQN
Rainbow DQN

Elements of both

DDPG, PPO, TD3, SAC

The best methods use elements of both.

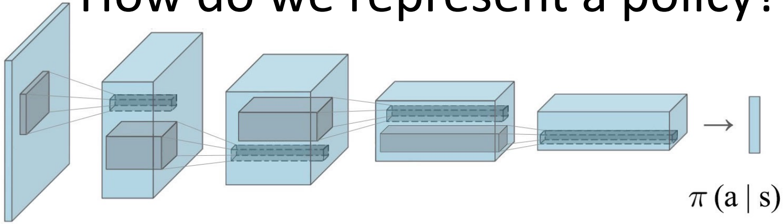
* Warning: precise definitions may vary

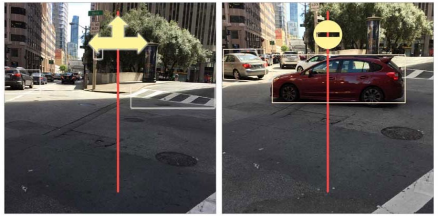
Example: Frozen Lake (Gymnasium)

- Aim:
  - Make it to the goal
  - Don’t fall into the holes
- Slippery (stochastic actions)
- Observation: current location

Example: Parameterized Policy

How do we represent a policy?





Normal Policy

$$\pi(a|s) = \frac{1}{\sigma_\omega(s)\sqrt{2\pi}} e^{-\frac{(a-\mu_\theta(s))^2}{2\sigma_\omega^2(s)}}$$

Continuous actions

Gibbs (softmax) Policy

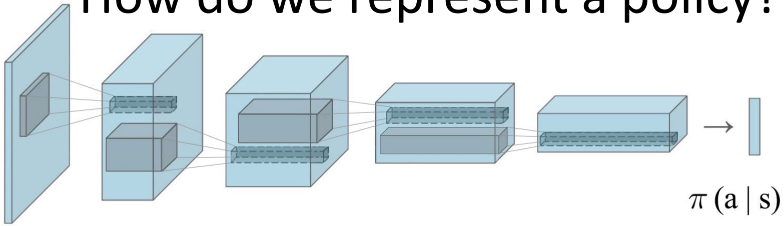
$$\pi(a|s) = \frac{e^{\mathcal{K} Q_\theta(s,a)}}{\sum_{a' \in \mathcal{A}} e^{\mathcal{K} Q_\theta(s,a')}}$$

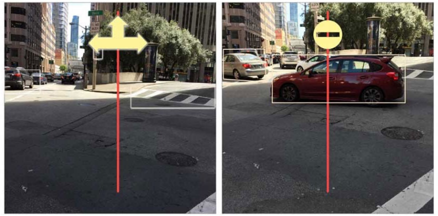
Discrete actions

Differentiable! → autodiff via PyTorch

Example: Parameterized Policy

How do we represent a policy?





Normal Policy

$$\pi(a|s) = \frac{1}{\sigma_\omega(s)\sqrt{2\pi}} e^{-\frac{(a-\mu_\theta(s))^2}{2\sigma_\omega^2(s)}}$$

Then:

$$\nabla_\theta \log \pi(a|s) = \frac{(a-\mu_\theta(s))}{\sigma_\omega^2(s)} \nabla_\theta \mu_\theta(s)$$

$$\nabla_\omega \log \pi(a|s) = \frac{(a-\mu_\theta(s))^2 - \sigma_\omega^2(s)}{\sigma_\omega^3(s)} \nabla_\omega \mu_\omega(s)$$

Continuous actions

Gibbs (softmax) Policy

$$\pi(a|s) = \frac{e^{\mathcal{K}Q_\theta(s,a)}}{\sum_{a'\in\mathcal{A}} e^{\mathcal{K}Q_\theta(s,a')}}$$

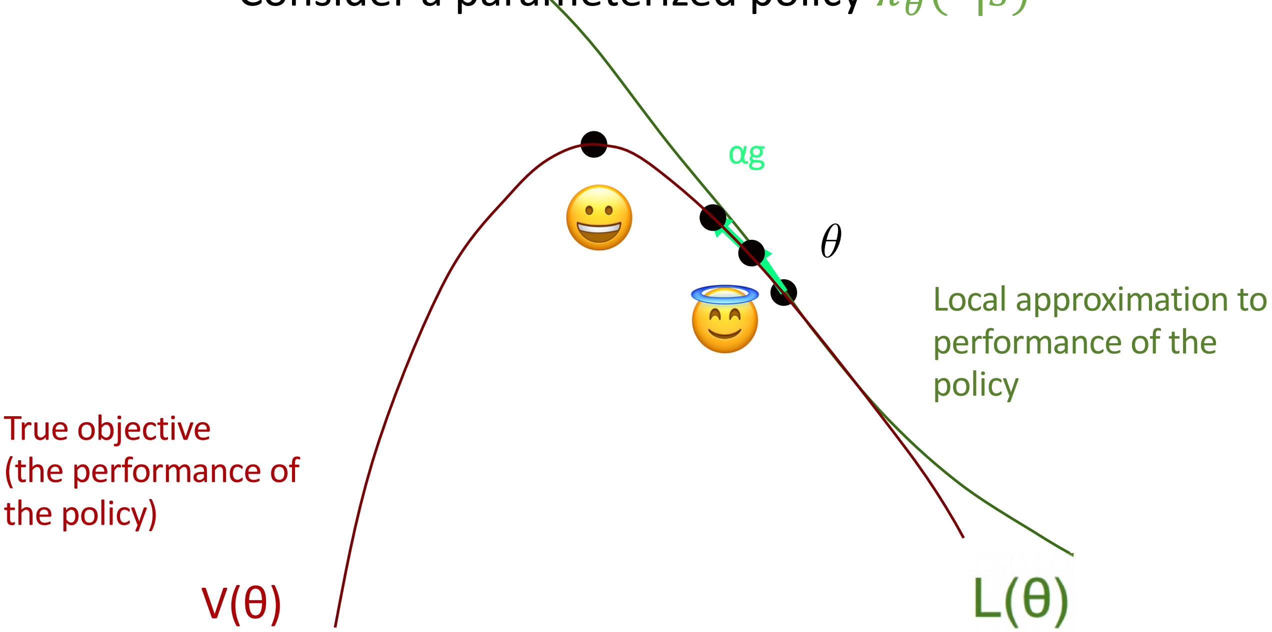
Then:

$$\nabla_\theta \log \pi(a|s) = \mathcal{K}\nabla_\theta Q_\theta(s,a) - \mathcal{K}\sum_{a'\in\mathcal{A}} \pi(a'|s)\nabla_\theta Q_\theta(s,a')$$

Discrete actions

Policy gradient = gradient ascent for MDPs

Consider a parameterized policy $\pi_\theta(\cdot\,|s)$



Adapted from Matteo Pirotta

Policy Gradient = gradient ascent for MDPs

$$V(\pi_{\theta_k}) = \mathbb{E}\left[\sum_{t=0}^{T-1} r_t | \pi_{\theta_k}, M\right] = \mathbb{E}_{\tau \sim \mathbb{P}(\tau | \pi_{\theta_k}, M)}[\mathcal{R}(\tau)]$$

Policy Gradient

$$\theta_{k+1} = \theta_k + \alpha_k \nabla_\theta V(\theta_k)$$

1. **How do we compute $\boldsymbol{\nabla_\theta V(\theta)}$?**
2. How quickly do we update (i.e. α_k)?

REINFORCE, variance reduction, baselines, generalized advantage estimation (GAE)

NPG, TRPO, PPO

Function approximation

Last time: Add function approximation to value iteration

This time: Add function approximation to policy iteration. Sorta.

Policy Iteration: Recap

Let π_0 be an arbitrary stationary policy.

while $k = 1, \dots, K$ **do**

Policy Evaluation: given π_k compute $V_k = V^{\pi_k}$

Policy Improvement: find π_{k+1} that is better than π_k

- e.g. compute the *greedy* policy:

$$\pi_{k+1}(s) \in \arg\max_{a \in \mathcal{A}} \left\{ r(s,a) + \gamma \sum_{y} p(y|s,a) V^{\pi_k}(y) \right\}$$

return the last policy π_K

end

- Convergence is finite and monotonic [Bertsekas, 2007] (in exact settings)

? Issues: Function approximation for V^{π_k} $\Longrightarrow$ Does it still converge?

Continuous Actions?

Approximate Policy Iteration with Q Functions

Recall the state-action cost-to-go function: $Q_\pi(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) Q_\pi(s', \pi(s'))$

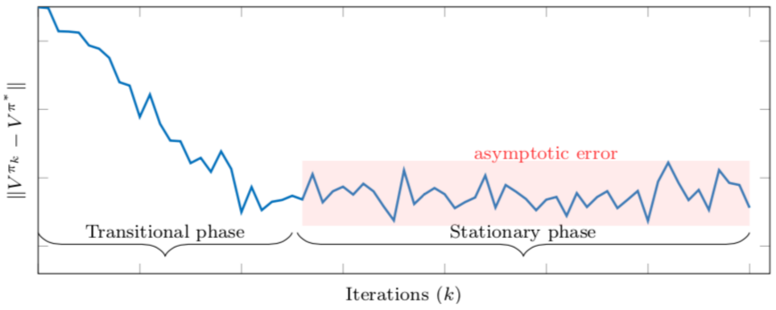
Approximate PI:

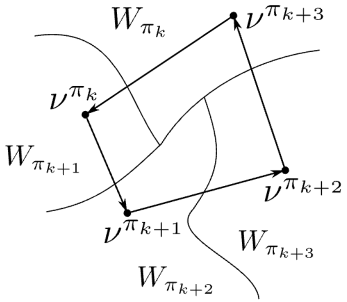
- For $k = 0, 1, 2, \ldots$
  1. Approximate the value under π_k: $Q_{\theta_k} \approx Q_{\pi_k}$
  2. Solve for an improved policy

$$\pi_{k+1}(s) \in \operatorname*{argmin}_{a \in A(s)} Q_{\theta_k}(s,a) \qquad \forall s \in \mathcal{S}$$

Q_{π_k} can be approximated by either TD or Monte Carlo methods.

Same story as fitted Q-iteration. No longer guaranteed to converge.





From Policy Iteration to Policy Search

- Approximate a stochastic policy directly using function approximation

$$\pi_\theta : S \to \mathcal{P}(\mathcal{A}) \text{ with } \theta \in \mathbb{R}^d$$

- Let $V(\pi_\theta)$ denote the policy performance of policy π_θ

➢ Policy optimization problem

$$\max_{\pi_\theta} V(\pi_\theta)$$

Solution 1: **Policy Search/Blackbox optimization**:

Use global optimizers or gradient by finite-difference methods

Policy π_θ can also be not differentiable w.r.t. θ

Solution 2: **Policy gradient optimization**:

Compute the gradient $\nabla_\theta V(\theta)$ and follow the ascent direction

$\nabla_\theta \pi_\theta(s,a)$ should exist

Policy Gradient as Policy Update

Approximate Policy Iteration

$$\pi_{\theta_{k+1}} = \arg\max_{\pi_\theta} Q^{\pi_\theta}\big(s, \pi_\theta(s)\big)$$

Unstable (fast)

No convergence

Policy Gradient

$$\theta_{k+1} = \theta_k + \alpha_k \nabla_\theta V(\theta_k)$$

Smooth, fine control (slow)

Convergence to local optima

1. **How do we compute $\boldsymbol{\nabla_\theta V(\theta)}$?**
2. How quickly do we update (i.e. α_k)?

Outline

1. From Policy Iteration to Policy Search
2. **Policy gradient methods**
   a. REINFORCE
   b. Representing a policy (discrete and continuous!)
   c. Variance reduction (temporal structure and baselines)
3. Actor-critic

Assume: finite-horizon setting

Discount γ excluded to simplify notation.

Policy Gradient (Finite-Horizon)

Given an MDP $M = (\mathcal{S}, \mathcal{A}, p, r, T, \mu)$ and a policy π_{θ_0}. For k = 1,2,…

1. Use π_{θ_k} to collect data τ.
2. Use τ to approximate gradient of:

Maximizing this is ultimately what we desire

$$V(\pi_{\theta_k}) = \mathbb{E}\left[\sum_{t=0}^{T-1} r_t | \pi_{\theta_k}, M\right] = \mathbb{E}_{\tau \sim \mathbb{P}(\tau|\pi_{\theta_k}, M)}[\mathcal{R}(\tau)]$$

where

- μ is an initial state distribution
- $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$ (includes terminal reward) is a trajectory
- $\mathcal{R}(\tau)$ its return (sum of rewards).

Main issue: MDP is a complex object to differentiate through, i.e. $\nabla_\theta \mathbb{P}(\tau|\pi_\theta, M)$. (Example: Frozen Lake)

3. Update $\theta_{k+1} = \theta_k + \alpha_k \widehat{\nabla_\theta V}(\pi_{\theta_k})$

How?

Policy Gradient (Finite-Horizon)

Policy Gradient Theorem [Williams, 1992; Sutton et al., 2000]

For any finite-horizon MDP $M = (\mathcal{S}, \mathcal{A}, p, r, T, \mu)$ and differentiable policy π_θ

$$\nabla_\theta V(\pi_\theta) = \mathbb{E}_{\tau \sim \mathbb{P}(\cdot|\pi, M)} \left[R(\tau) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta (s_t, a_t) \right]$$

Gradient is now on the inside! We can differentiate through (differentiable) policies.

- Model-free! Why?
- Compare: taking gradient through trajectory-space is difficult

$$\nabla_\theta V(\pi_\theta) = \nabla_\theta \mathbb{E}_\tau [R(\tau)] = \nabla_\theta \int \mathbb{P}(\tau|\pi_\theta, M) R(\tau) d\tau$$

Proof

- The objective is an expectation. Want to compute the gradient w.r.t. θ (simplify notation from: $V(\pi_\theta)$ to $V(\theta)$). First, bring the gradient to the inside.

$$\nabla_\theta V(\theta) = \nabla_\theta \mathbb{E}_\tau[R(\tau)] = \nabla_\theta \int \mathbb{P}(\tau|\pi_\theta, M)R(\tau)d\tau$$

Log trick

$$\nabla_\theta \log \mathbb{P}(\tau|\pi_\theta, M) = \frac{\nabla_\theta \mathbb{P}(\tau|\pi_\theta, M)}{\mathbb{P}(\tau|\pi_\theta, M)}$$

$$= \int \nabla_\theta \mathbb{P}(\tau|\pi_\theta, M)R(\tau)d\tau$$

$$= \int \mathbb{P}(\tau|\pi_\theta, M)\nabla_\theta \log \mathbb{P}(\tau|\pi_\theta, M)\, R(\tau)d\tau$$

$$= \mathbb{E}_\tau[R(\tau)\nabla_\theta \log \mathbb{P}(\tau|\pi_\theta, M)]$$

- Last expression is an unbiased gradient estimator
  Just sample $\tau_t \sim \mathbb{P}(\tau|\pi_\theta, M)$, and compute $\hat{g}_t = R(\tau_t)\nabla_\theta \log \mathbb{P}(\tau_t|\pi_\theta, M)$
- Issue: Need to be able to compute & differentiate the density $\mathbb{P}(\tau|\pi_\theta, M)$ w.r.t θ

Proof

Likelihood (with stochastic policies)

$$\mathbb{P}(\tau|\pi_\theta, M) = \mu(s_0) \prod_{t=0}^{T-1} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$$

$$\log \mathbb{P}(\tau|\pi_\theta, M) = \log \mu(s_0) + \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t) + \log p(s_{t+1}|s_t, a_t)$$

$$\nabla_\theta \log \mathbb{P}(\tau|\pi_\theta, M) = \cancelto{0}{\nabla_\theta \log \mu(s_0)} + \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) + \cancelto{0}{\nabla_\theta \log p(s_{t+1}|s_t, a_t)}$$

→ model free

Alternative proof: likelihood rescaling

- Interested in policy gradient: $\nabla_\Delta V(\theta + \Delta)|_{\Delta=0}$
- Likelihood rescaling

$$V(\theta + \Delta) = \mathbb{E}_{\tau(\theta)}\left[R(\tau(\theta))\frac{\prod_t \pi_{\theta+\Delta}(a_t|s_t)}{\prod_t \pi_\theta(a_t|s_t)}\right]$$

- Apply chain rule to get

$$\nabla_\Delta V(\theta + \Delta)\Big|_{\Delta=0} = \mathbb{E}_{\tau(\theta)}\left[R(\tau(\theta))\sum_t \frac{\nabla\, \pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)}\right]$$
$$= \mathbb{E}_\tau[R(\tau)\textstyle\sum_t \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

Policy Gradient (Finite-Horizon)

Policy Gradient Theorem [Williams, 1992; Sutton et al., 2000]

For any finite-horizon MDP $M = (\mathcal{S}, \mathcal{A}, p, r, T, \mu)$ and differentiable policy π_θ

$$\nabla_\theta V(\pi_\theta) = \mathbb{E}_{\tau \sim \mathbb{P}(\cdot|\pi, M)}\left[R(\tau) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(s_t, a_t)\right]$$

Gradient is now on the inside! We can differentiate through (differentiable) policies.

- Model-free! Why?
- Compare: taking gradient through trajectory-space is difficult

$$\nabla_\theta V(\pi_\theta) = \nabla_\theta \mathbb{E}_\tau[R(\tau)] = \nabla_\theta \int \mathbb{P}(\tau|\pi_\theta, M) R(\tau) d\tau$$

REINFORCE [Williams, 1992]

1. Let π_{θ_1} be an arbitrary policy.
2. At each iteration $k = 1, \dots, K$
   - Sample m trajectories $\tau_i = (s_0, a_0, r_0, s_1, \dots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$ following π_k
   - Compute unbiased gradient estimate:

$$\widehat{\nabla_\theta V}(\pi_{\theta_k}) = \frac{1}{m}\sum_{i=1}^{m}\left(\sum_{t=0}^{T-1} r_t^i\right)\left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi_{\theta_k}(a_t^i | s_t^i)\right)$$

   - Update parameters:

$$\theta_{k+1} = \theta_k + \alpha_k \widehat{\nabla_\theta V}(\pi_{\theta_k})$$

Monte Carlo approximation of policy gradient

3. Return last policy π_{θ_K}

REINFORCE [Williams, 1992]

1. Let π_{θ_1} be an arbitrary policy.
2. At each iteration $k = 1, \ldots, K$
   - Sample m trajectories $\tau_i = (s_0, a_0, r_0, s_1, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$ following π_k
   - Compute unbiased gradient estimate:
   $$\widehat{\nabla_\theta V}(\pi_{\theta_k}) = \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=0}^{T-1} r_t^i \right) \left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi_{\theta_k}(a_t^i | s_t^i) \right)$$
   - Update parameters:
   $$\theta_{k+1} = \theta_k + \alpha_k \widehat{\nabla_\theta V}(\pi_{\theta_k})$$
3. Return last policy π_{θ_K}

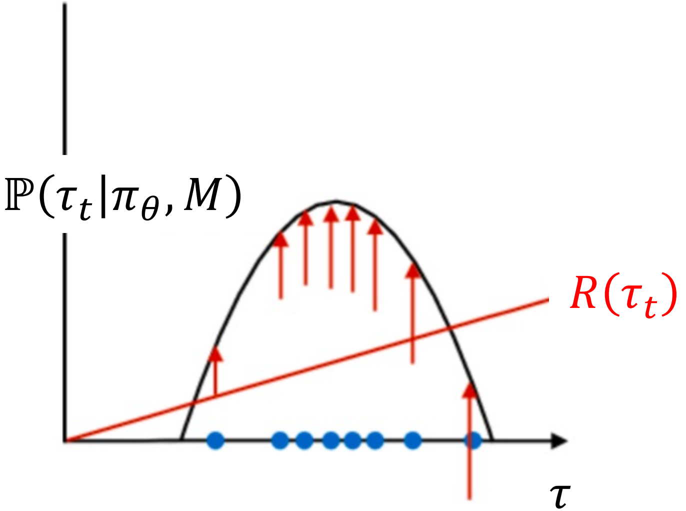
Monte Carlo approximation of policy gradient

Wu

REINFORCE as Supervised Learning

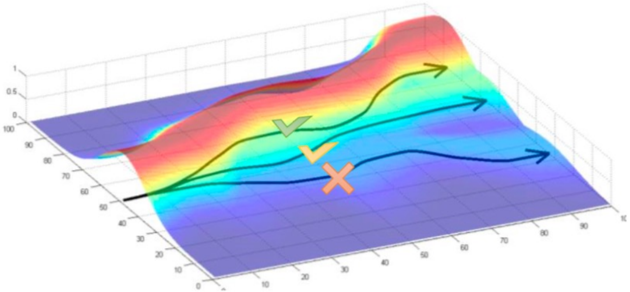
$$\hat{g}_t = R(\tau_t)\nabla_\theta \log \mathbb{P}(\tau_t|\pi_\theta, M)$$

- $R(\tau_t)$ measures how good is sample τ_t
- Moving in the direction of $\hat{g}_t$ pushes up the log probability of the sample in proportion to how good it is.



Interpretation: uses good trajectories as supervised examples

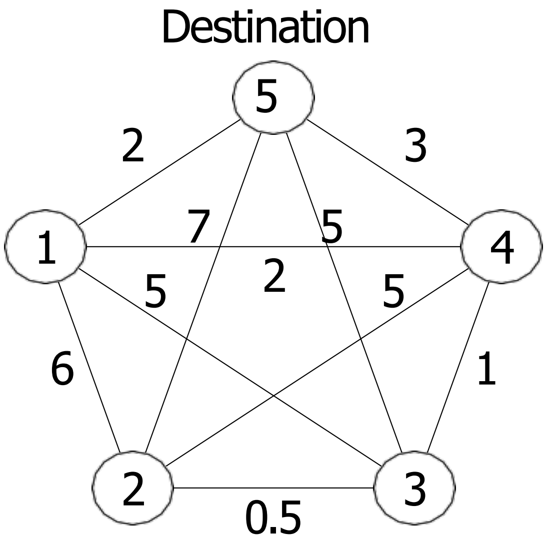
- Like maximum likelihood in supervised learning
- Good stuff are made more likely while bad less
- Trial and Error approach



From "CS 294-112: Deep Reinforcement Learning" slides by S. Levine

Dynamic programming vs policy gradient

How would policy gradient solve shortest path?



Destination is node 5.

REINFORCE

Pros

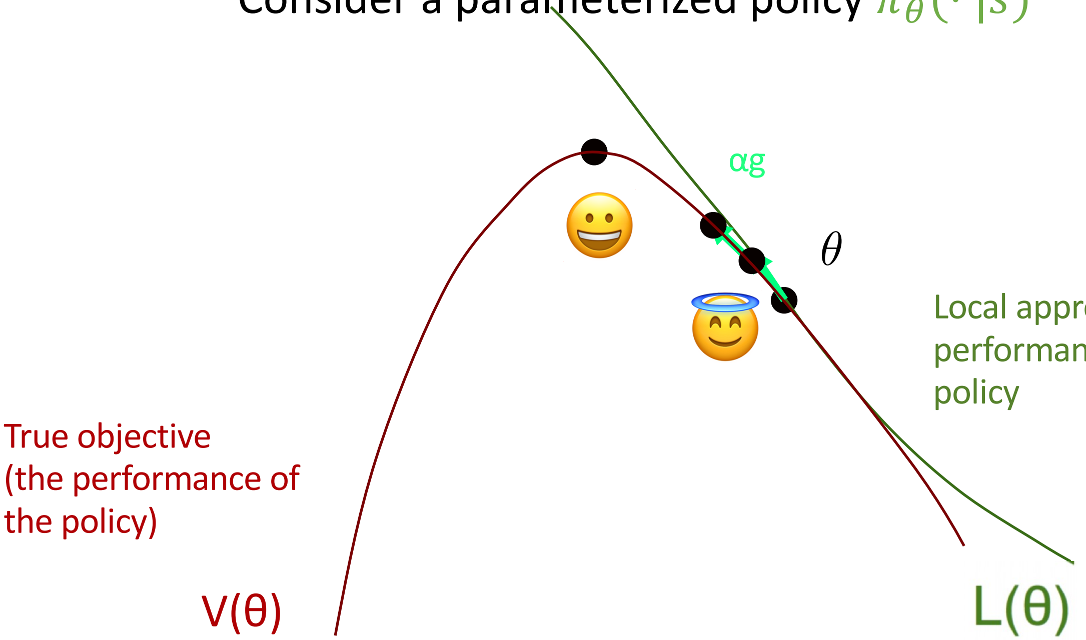
- Easy to compute
- Does not use Markov property!
- Can be used in partially observable MDPs without modification

Issues

- Use a MC estimate of $Q(s, a)$
- It has possibly a very large variance
- Needs many samples to converge

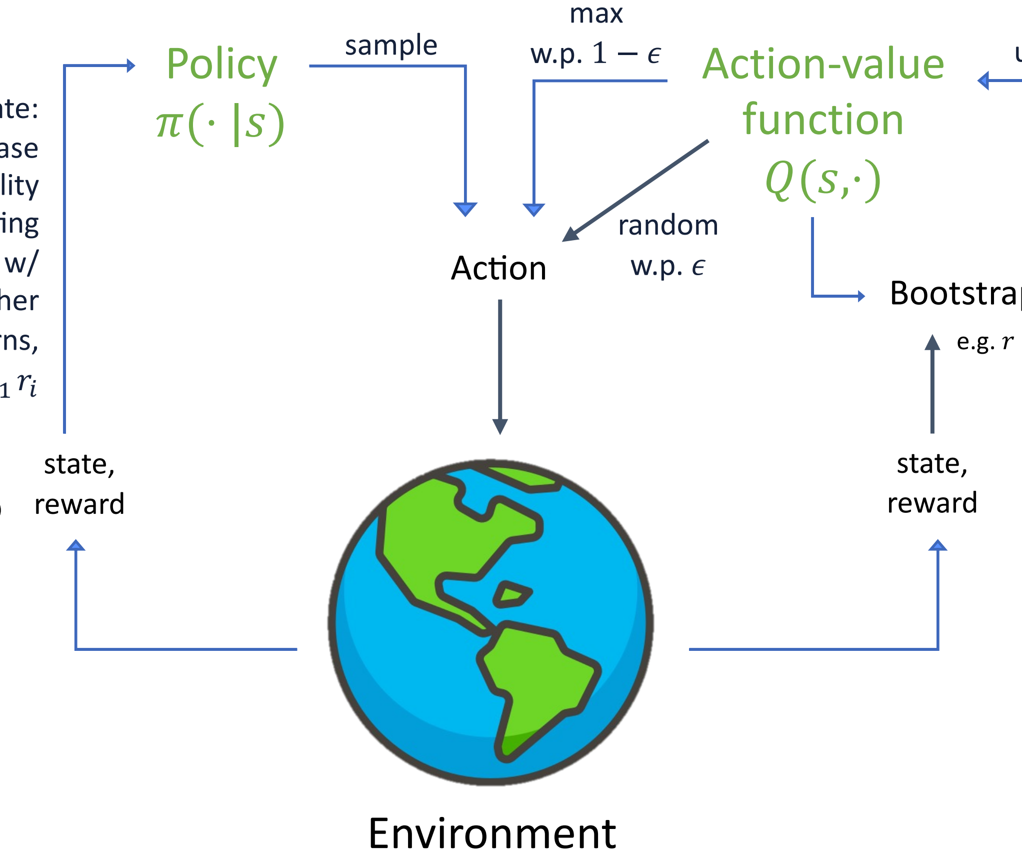
Policy gradient = gradient ascent for MDPs

Consider a parameterized policy $\pi_\theta(\cdot\,|s)$



Adapted from Matteo Pirotta

Policy-based vs value-based methods



Policy-based methods

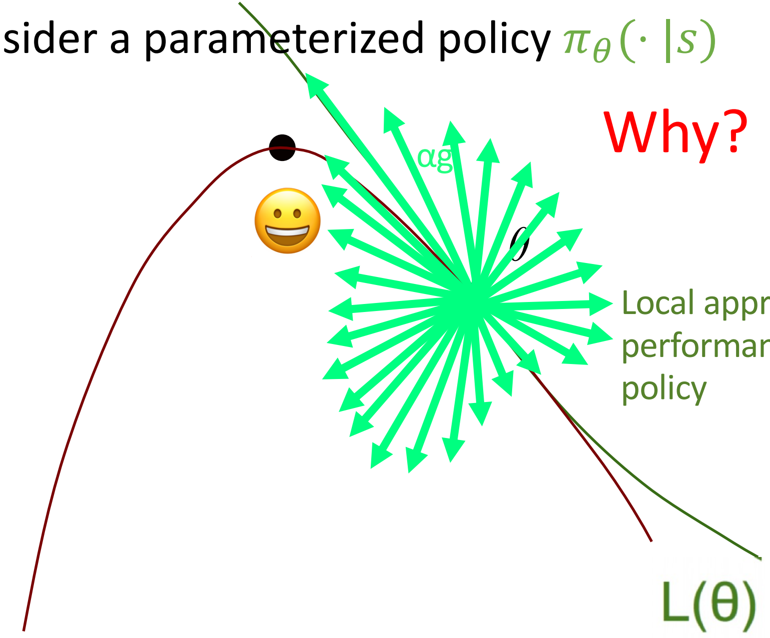
- Sensible & simple ☺
- Local convergence guarantees only
- Also "works" for partial observation, nonstationary settings
- Doesn't work that well (on its own) ☹

Value-based methods

- Conceptually more complicated
- Global convergence guarantees for tabular discounted infinite horizon MDPs ☺
- Doesn't work that well outside of its comfort zone (on its own) ☹

Key challenge: Policy gradient has high variance

Consider a parameterized policy $\pi_\theta(\cdot\,|s)$



$$\widehat{\nabla_\theta V}(\pi_{\theta_k}) = \frac{1}{m}\sum_{i=1}^{m}\left(\sum_{t=0}^{T-1} r_t^i\right)\left(\sum_{t=0}^{T-1}\nabla_\theta \log \pi_{\theta_k}(a_t^i|s_t^i)\right)$$

Adapted from Matteo Pirotta

Policy Gradient: Temporal Structure (Causality)

$$\nabla_\theta V(\pi_\theta) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t'=t}^{T-1} r_{t'}\right]$$

Discuss: Why does this help with variance?

Because $\forall t$:

Actions don't affect past rewards

$$\mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s_t) \sum_{t'=0}^{t-1} r_i \,|\tau_{0:t-1}\right] = \left(\sum_{t'=0}^{t-1} r_i\right) \int \pi_\theta(s_t, a) \nabla_\theta \log \pi_\theta(a|s_t) da$$

$$= \left(\sum_{t'=0}^{t-1} r_i\right) \int \nabla_\theta \pi_\theta(a|s_t) da$$

$$= \left(\sum_{t'=0}^{t-1} r_i\right) \nabla_\theta \underbrace{\int \pi_\theta(a|s_t) da}_{:=1} = 0$$

In literature known as G(PO)MDP [Peters and Schaal, 2008b].

Policy Gradient: Baseline

- Further reduce the variance by introducing a baseline $b(s)$

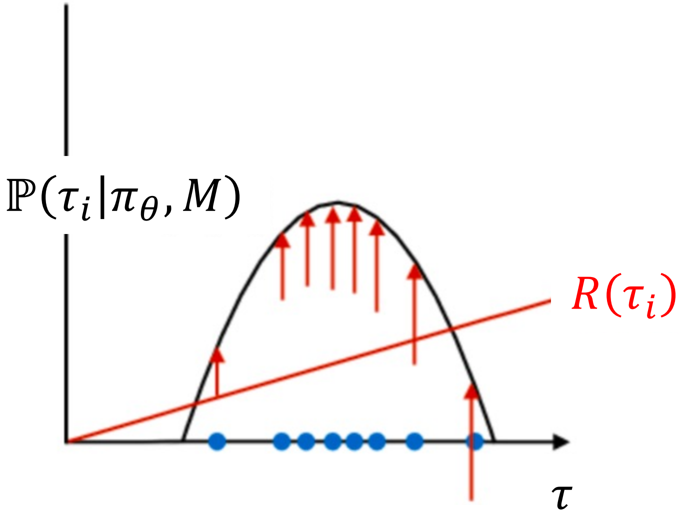
$$\nabla_\theta V(\pi_\theta) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(s_t, a_t)\left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t)\right)\right]$$

- The gradient estimate is still unbiased.
- Proof: State-dependent baselines do not introduce bias (zero mean).

Solution: Baseline

Regular policy gradient

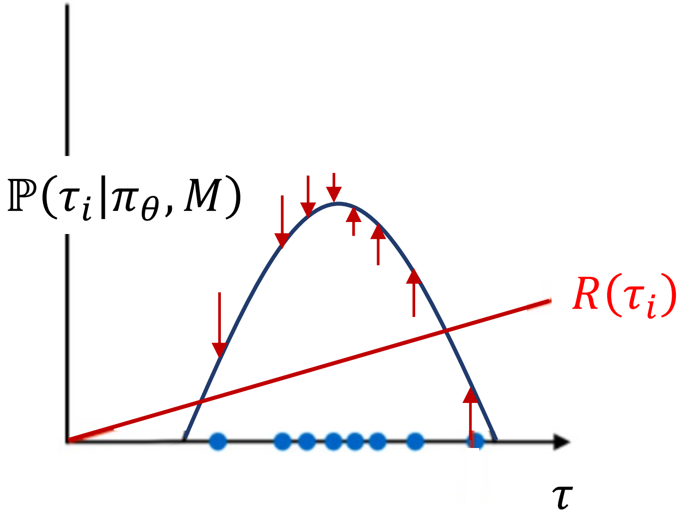
- $\hat{g}_i = R(\tau_i)\nabla_\theta \log \mathbb{P}(\tau_i|\pi_\theta, M)$



Encourage all trajectories

Policy gradient with baseline

- $\hat{g}_i = (R(\tau_i) - V(\tau_i))\nabla_\theta \log \mathbb{P}(\tau_i|\pi_\theta, M)$



Encourage trajectories that are better than average

Variance reduction via baseline?

- Baseline

$$\nabla_\theta V(\pi_\theta) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(s_t, a_t)\left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t)\right)\right]$$

- "Near optimal choice" that minimize the variance is the expected sum of returns:

$$b^\star(s) \approx \mathbb{E}\left[\sum_{t=0}^{T-1} r_t \middle| s_0 = s, \pi_\theta, M\right] = V^{\pi_\theta}(s)$$

  - *Interpretation*: increase the log probability of an action a_t proportionally to how much returns are **better than expected** (relative values).

- Intuition: To reduce variance, try to maximize the covariance between x and y

$$\mathrm{Var}(x - y) = \mathrm{Var}(x) - 2\mathrm{Cov}(x, y) + \mathrm{Var}(y)$$

Optimal Baseline Derivation

$$\nabla_{\theta_i} V(\pi_\theta) = \mathbb{E}_\tau\Big[\underbrace{\nabla_{\theta_i} \log \mathbb{P}(\tau|\pi_\theta)}_{:= g(\tau)}(R(\tau) - b)\Big]$$

$$\mathrm{Var} = \mathbb{E}_\tau[(g(\tau)(R(\tau) - b))^2] - (\mathbb{E}_\tau[g(\tau)(R(\tau) - b)])^2$$

$$\Longrightarrow \mathbb{E}_\tau[g(\tau)R(\tau)]^2$$

[Baseline is unbiased in expectation]

$$\frac{\partial}{\partial \mathrm{b}}\mathrm{Var} = \frac{\partial}{\partial \mathrm{b}}\mathbb{E}_\tau[g(\tau)^2(R(\tau) - b)^2]$$

$$= \cancel{\frac{\partial}{\partial \mathrm{b}}\mathbb{E}_\tau[g(\tau)^2 R(\tau)^2]}^{\,0} - 2\frac{\partial}{\partial \mathrm{b}}\mathbb{E}_\tau[g(\tau)^2 R(\tau) b] + \frac{\partial}{\partial \mathrm{b}}\mathbb{E}_\tau[b^2 g(\tau)^2]$$

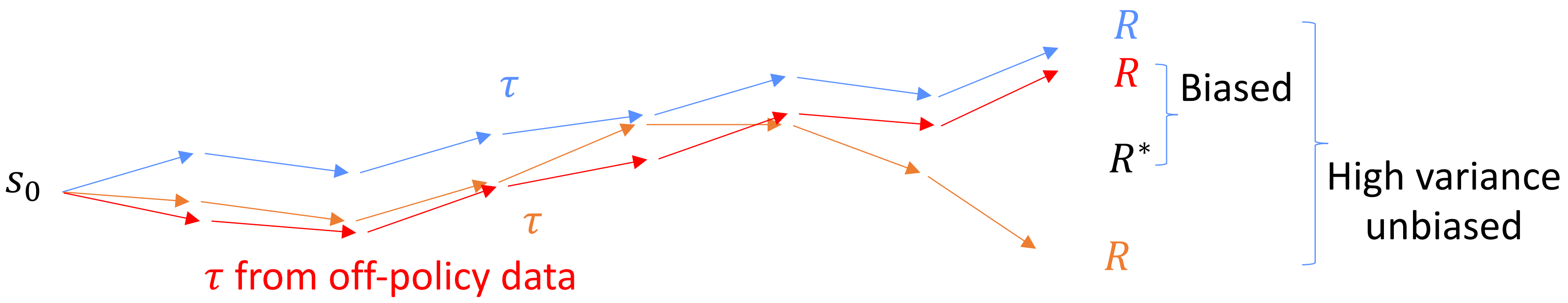
$$\Longrightarrow b^\star(\tau) = \frac{\mathbb{E}_\tau[g(\tau)^2 R(\tau)]}{\mathbb{E}_\tau[g(\tau)^2]}$$

Expected return weighted by the magnitude of the gradient.

Outline

1. From Policy Iteration to Policy Search
2. Policy gradient methods
3. **Actor-critic**
   a. Compatible function approximation
   b. Advantages and Advantage Actor-Critic (A2C)
   c. Asynchronous A2C (A3C)
   d. Deep Deterministic Policy Gradient (DDPG)
   e. Soft Actor-Critic (SAC)

Policy gradients & high variance: the saga continues



- Monte-Carlo policy gradient is unbiased but still has high variance

$$\nabla_\theta V(\pi_\theta) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t'=t}^{T-1} r_{t'}\right]$$

- Policy gradient is on-policy (doesn't re-use data → inefficient!)

Policy- and value-based methods → actor-critic

- Monte-Carlo policy gradient is unbiased but still has high variance

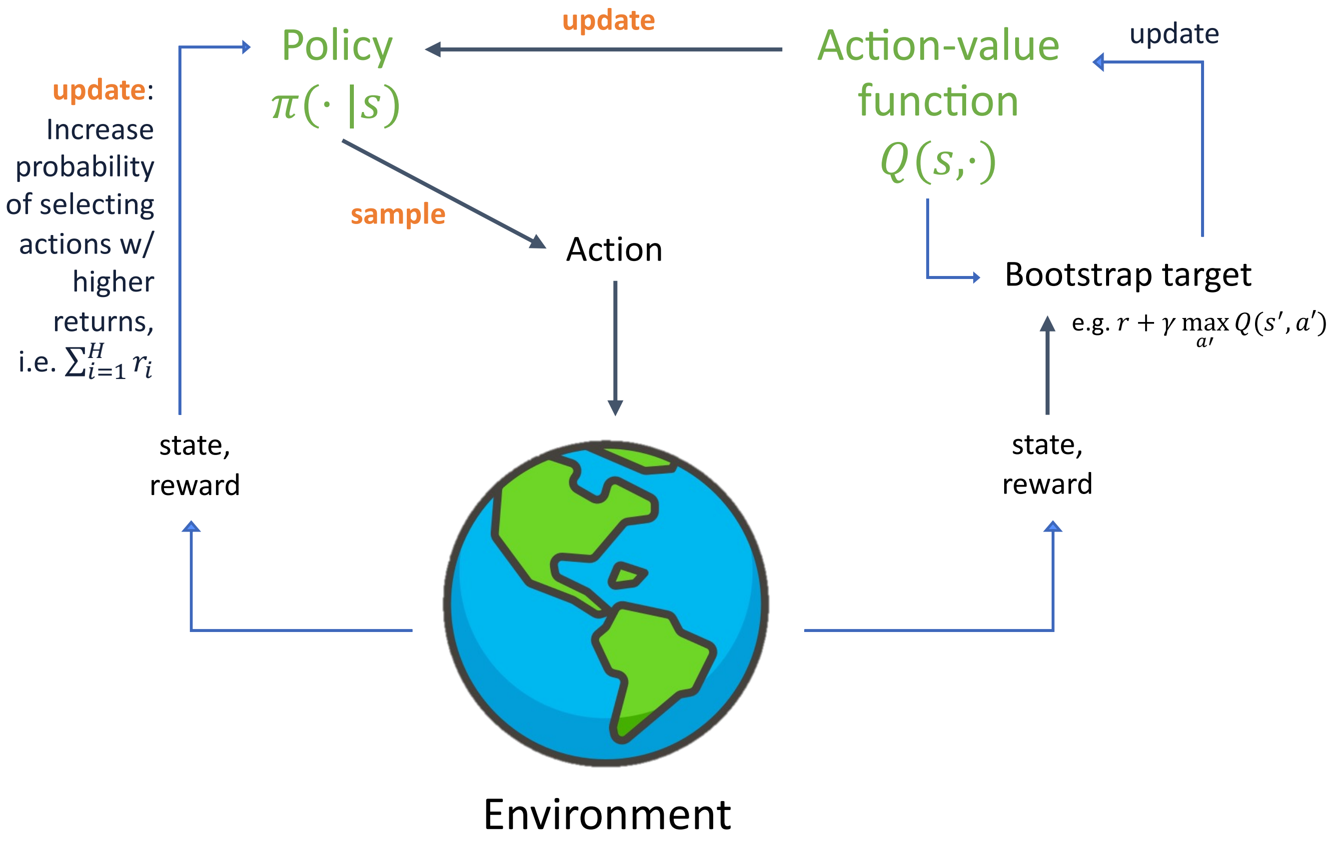
$$\nabla_\theta V(\pi_\theta) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t'=t}^{T-1} r_{t'}\right]$$

- Incorporate an estimate of $Q^\pi(s,a) \implies$ actor-critic
  - Critic: estimate the value function
  - Actor: update the policy in the direction suggested by the critic
- Actor-critic

$$\nabla_\theta V(\pi_\theta) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\, Q^{\pi_\theta}(s_t,a_t)\right]$$

- These are equivalent (see HW).

Actor-critic methods



Actor-Critic

- Algorithm maintains two sets of parameters: $\theta \longmapsto \pi_\theta, \omega \longmapsto Q_\omega$
- Critic can use $TD(0)$

---

for $t = 0, \ldots, T-1$ **do**

$a_t \sim \pi_\theta(s_t, \cdot)$ and observe r_t and s_{t+1}

Compute temporal difference

$$\delta_t = r_t + \gamma Q_\omega(s_{t+1}, a_{t+1}) - Q_\omega(s_t, a_t)$$

Update Q estimate

$$\omega = \omega + \beta \delta_t \nabla_\omega Q_\omega(s_t, a_t)$$

Update policy

$$\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(a_t | s_t) Q_\omega(s_t, a_t)$$

end

Actor-Critic

Issues:

- $Q_\omega(s,a)$ is a biased estimate of $Q^{\pi_\theta}(s,a)$
- The update of θ may not follow the gradient of $\nabla_\theta V(\pi_\theta)$

Solution:

- Choose the approximation space $Q_\omega(s,a)$ carefully
  $\Longrightarrow$ compatible function approximation between Q_ω and π_θ

Compatible Function Approximation

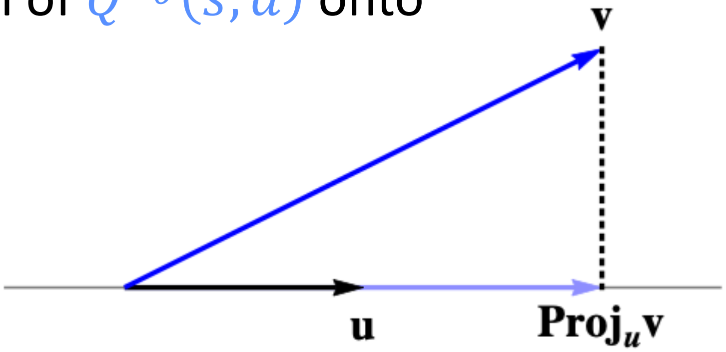
- Actor-critic

$$\nabla_\theta V(\pi_\theta) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\, Q^{\pi_\theta}(s_t, a_t)\right]$$

- Re-write using occupancy measures

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

- Interpretation (inner product): projection of $Q^{\pi_\theta}(s, a)$ onto subspace spanned by $\nabla_\theta \log \pi_\theta(a|s)$
- Let $Q_\omega(s, a) = \sum_i \alpha_i [\nabla_\theta \log \pi_\theta(s, a)]_i$ where $\omega = (\alpha_i)_{|\theta|}$



Compatible Function Approximation

Theorem (Silver, 2014)

An action value function space Q_ω is compatible with a policy space π_θ if:

1. **[Feature Selection]** $\nabla_\omega Q_\omega(s,a) = \nabla_\theta \log \pi_\theta(s,a)$
2. **[Least Squares Fitting]** And if ω minimizes the squared error

$$\omega = \arg\min_\omega \mathbb{E}_{s\sim d^{\pi_\theta}}\left[\sum_a \pi_\theta(a|s)\big(Q^{\pi_\theta}(s,a) - Q_\omega(s,a)\big)^2\right]$$

Then:

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

- Remark 1: conditions for which the policy gradient is exact.
- Remark 2: approximately satisfied by linear function approximation.

Sample Efficiency in Actor-Critic

Issues:

- Sample efficiency is pretty poor
- All samples need to be generated by the current policy (on-policy learning)
- Samples are discarded after a single update

Solutions:

- Variance reduction techniques
- Asynchronous training (A3C)
- Use samples from other policies via importance sampling (not very stable) (next time)
- Conservative approaches (next time)
- Newton for Quasi-newton methods

Actor-Critic with a Baseline

$$\nabla_\theta V(\pi_\theta) = \mathbb{E}_{s \sim d^{\pi_\theta}}\left[\sum_a \nabla_\theta \pi_\theta(s,a)\left(Q^{\pi_\theta}(s,a) - b(s)\right)\right]$$

- $b(s)$ minimizes the variance
- $V^\pi(s)$ is a good choice as baseline
  - It minimizes the variance in average reward [Bhatnagar et al., 2009]
- $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$ is the advantage function

Actor-Critic with Advantage Function (A2C)

- It is possible to estimate V^{π} and Q^{π} independently (e.g. by $TD(0)$)
- $A^{\pi} = Q_{\omega} - V_{\mathcal{V}}$ is a biased and unstable estimate

Solution:

- Consider the temporal difference error

$$\delta^{\pi_\theta} = r(s,a) + \gamma V^{\pi_\theta}(s') - V^{\pi_\theta}(s)$$

- δ^{π_θ} is an unbiased estimate of the advantage

$$\begin{aligned}\mathbb{E}[\delta^{\pi_\theta}|s,a] &= \mathbb{E}[r(s,a) + \gamma V^{\pi_\theta}(s')|s,a] - V^{\pi_\theta}(s) \\ &= Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)\end{aligned}$$

Actor-Critic with Advantage Function (A2C)

- Estimate only $V_v \longmapsto \delta_v = r + \gamma V_v(s') - V_v(s)$

☞ Convergence results with compatible function approximation [Bhatnagar et al., 2009]

for $t = 0, \ldots, T$ **do**

$a_t \sim \pi^\theta(s_t, \cdot)$ and observer r_t and s_{t+1}

Compute temporal difference

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

Update V estimate

$$v = v + \beta \delta_t \nabla_v V_v(s_t)$$

Update policy

$$\theta = \theta + \alpha \delta_t \nabla_\theta \log \pi_\theta(a_t | s_t)$$

end

Compare (actor-critic):

$$\delta_t = r_t + \gamma Q_\omega(s_{t+1}, a_{t+1}) - Q_\omega(s_t, a_t)$$
$$\omega = \omega + \beta \delta_t \nabla_\omega Q_\omega(s_t, a_t)$$
$$\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(a_t | s_t) Q_\omega(s_t, a_t)$$

Generalized advantage estimation (GAE) (2016)

- A2C: Compute advantages in manner analogous to TD(0)
- GAE: Compute advantages in manner analogous to TD(λ)
- Can generally be used with actor-critic methods
  - Example algorithm: TRPO (next time)

[Generalized advantage estimation demo: learning to run and stand up]

[A Compilation of Robots Falling Down at the DARPA Robotics Challenge]

🤕

Asynchronous Advantage Actor-Critic (A3C)

- Multiple independent agents (networks) with their own weights, who interact with a different copy of the environment in parallel.
- The agents (or workers) train in parallel using a global network θ. They periodically update the global network with their $d\theta$.
- Remark: In practice, θ denotes the shared weights for the value function and the policy (multi-headed network)

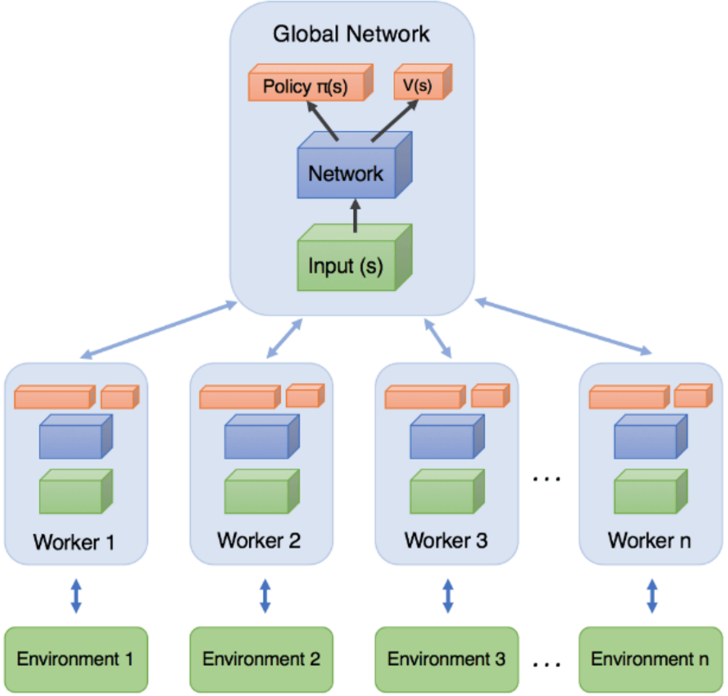
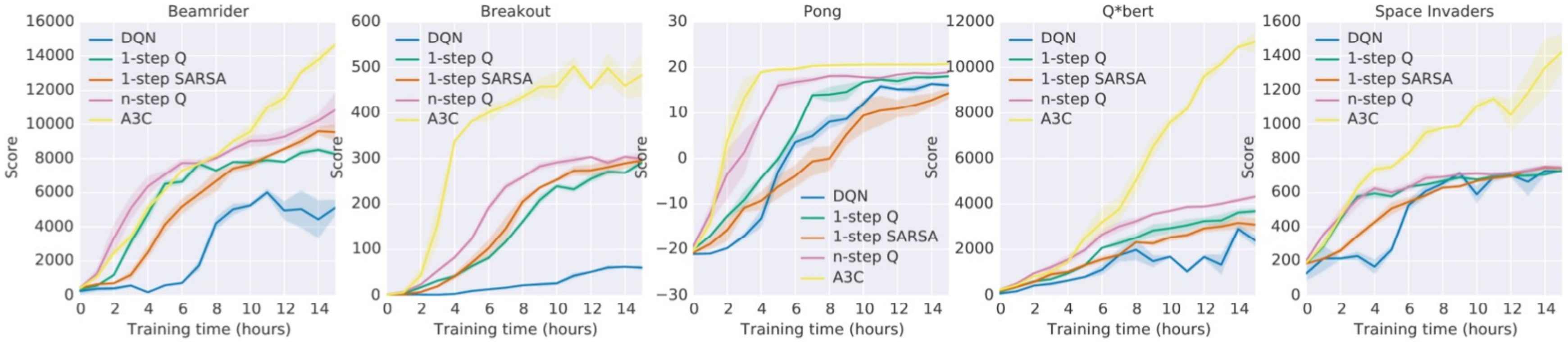


Figure from Atrisha Sarkar

Mnih, Volodymyr, et al. "Asynchronous methods for deep reinforcement learning." *ICML*, 2016.

Asynchronous Advantage Actor-Critic (A3C)

- Improved training exploration & stability.



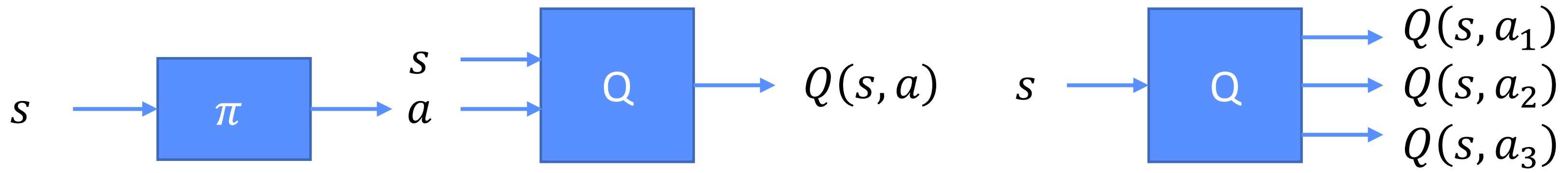
Mnih, Volodymyr, et al. "Asynchronous methods for deep reinforcement learning." *ICML*, 2016.

Outline

1. From Policy Iteration to Policy Search
2. Policy gradient methods
3. **Actor-critic**
   a. Compatible function approximation
   b. Advantages and Advantage Actor-Critic (A2C)
   c. Asynchronous A2C (A3C)
   d. **Deep Deterministic Policy Gradient (DDPG)**
   e. **Soft Actor-Critic (SAC)**

Bringing policies back to value-based methods

- Recall: value-based methods have trouble handling continuous actions/large action spaces
- Key idea: simplify Q using deterministic policies



Deep Deterministic Policy Gradient (DDPG) (2014)

- Recall: $V_D(\pi) = \mathbb{E}_{s\sim d^\pi}\left[r\left(s,\pi(s)\right)\right]$
- $\nabla_\theta V_D(\theta) = \sum_s d^\pi(s)\nabla_\theta \pi_\theta(s)\nabla_a Q^\pi(s,a)|_{a=\pi_\theta(s)} = \mathbb{E}_{s\sim d^\pi}\left[\nabla_\theta \pi_\theta(s)\nabla_a Q^\pi(s,a)|_{a=\pi_\theta(s)}\right]$

Plug it into an actor-critic framework

- Use $TD(0)$ to update a parametric representation of Q^π

$$\delta_t = R_t + \gamma Q_w(s_{t+1},a_{t+1}) - Q_w(s_t,a_t)$$; TD error in SARSA

$$w_{t+1} = w_t + \alpha_w \delta_t \nabla_w Q_w(s_t,a_t)$$

$$\theta_{t+1} = \theta_t + \alpha_\theta \nabla_a Q_w(s_t,a_t)\nabla_\theta \pi_\theta(s)\Big|_{a=\pi_\theta(s)}$$; Deterministic policy gradient theorem

- Issue: Need to explicitly force exploration, e.g. “behavior policy” $\beta(\cdot)\sim \mathcal{N}(\theta,\sigma\beta^2)$

Policy Iteration: Recap

Let π_0 be an arbitrary stationary policy.

while $k = 1, \ldots, K$ **do**

Policy Evaluation: given π_k compute $V_k = V^{\pi_k}$

Policy Improvement: find π_{k+1} that is better than π_k

- e.g. compute the *greedy* policy:

$$\pi_{k+1}(s) \in \arg\max_{a \in \mathcal{A}} \left\{ r(s,a) + \gamma \sum_{y} p(y|s,a) V^{\pi_k}(y) \right\}$$

return the last policy π_K

end

- Convergence is finite and monotonic [Bertsekas, 2007] (in exact settings)
- ? Issues: Function approximation for $V^{\pi_k} \Longrightarrow$ Does it still converge?
  Continuous Actions?

Recap: Approximate Policy Iteration with Q Functions

Recall the state-action cost-to-go function: $Q_\pi(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) Q_\pi(s', \pi(s'))$

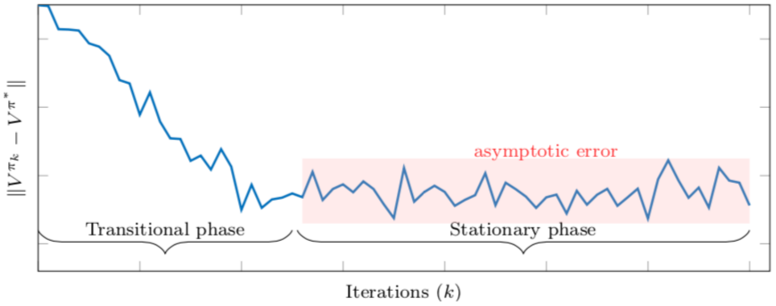
Approximate PI:

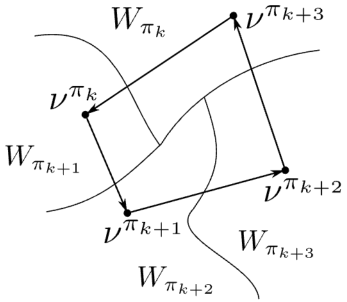
- For $k = 0, 1, 2, \ldots$
  1. Approximate the value under π_k: $Q_{\theta_k} \approx Q_{\pi_k}$
  2. Solve for an improved policy

$$\pi_{k+1}(s) \in \underset{a \in A(s)}{\operatorname{argmin}}\, Q_{\theta_k}(s,a) \qquad \forall s \in \mathcal{S}$$

Q_{π_k} can be approximated by either TD or Monte Carlo methods.

Same story as fitted Q-iteration. No longer guaranteed to converge.





Soft policy iteration [Haarnoja, 2018]

- **Soft policy evaluation**:

  We define the bellman backup operator for any $Q : \mathcal{S} \times \mathcal{A} \to \Re$:

  $$\mathcal{T}^\pi Q(s_t, a_t) \triangleq r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p}[V(s_{t+1})]$$

  where we have the soft state value function:

  entropy regularization

  $$V(s_t) = \mathbb{E}_{a_t \sim \pi}[Q(s_t, a_t) - \boxed{\alpha \log \pi(a_t|s_t)]}$$

- Under standard assumptions: Q^k will converge to the soft Q-value of π as $k \to \infty$

Soft policy iteration [Haarnoja, 2018]

- **Soft policy improvement**:

For each state, we do the following update:

$$\pi_{\text{new}} = \arg\min_{\pi' \in \Pi} D_{\text{KL}} \left(\pi'(\cdot|s_t) \Big\| \frac{1}{Z^{\pi_{\text{old}}}(s_t)} \cdot \exp(\frac{1}{\alpha} \cdot Q^{\pi_{\text{old}}}(s_t, \cdot)) \right).$$

Then we have

$$Q^{\pi_{\text{new}}}(s_t, a_t) \geq Q^{\pi_{\text{old}}}(s_t, a_t) \quad \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}, |\mathcal{A}| < \infty.$$

Soft policy iteration:

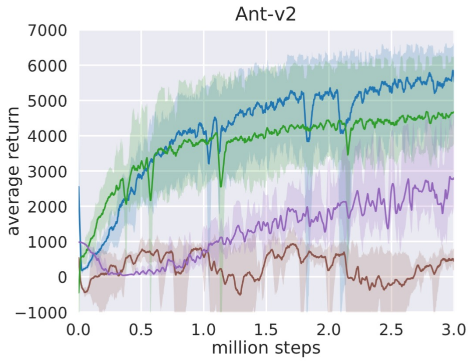
- Under standard assumptions: The sequence Q^{π_i} is monotonically increasing and bounded. So, it converges to some π^*.

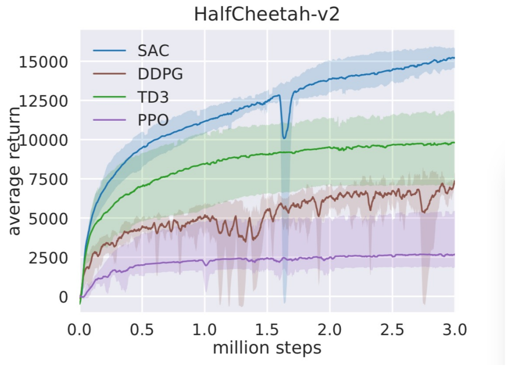
Soft actor-critic (SAC) [Haarnoja, 2018]

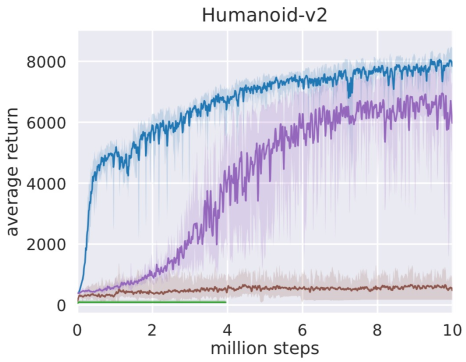
Soft policy iteration + function approximation

1. **[Soft policy evaluation]**
   Train the action-value function Q_θ, minimizing:
   $$\arg\min_{\theta} \mathbb{E}_{(s,a)\in H}\left[\frac{1}{2}\left(Q_\theta(s_t,a_t) - \left(r(s_t,a_t) + \gamma\mathbb{E}[V_{\bar{\psi}}(s')]\right)\right)^2\right]$$
   ! Fix the target network (e.g. DQN) → increase stability / break dependences
2. Train the (soft) value function V_ψ, minimizing:
   $$J_V(\psi) = \mathbb{E}_{s_t\sim D}\left[\frac{1}{2}\left(V_\psi(s_t) - \mathbb{E}_{a_t\sim\pi_\phi}[Q_\theta(s_t,a_t) \underbrace{- \log\pi_\phi(a_t|s_t)}_{\text{entropy regularization}}]\right)^2\right]$$
3. **[Soft policy improvement]**
   Fit the new (stochastic) policy π_ϕ:
   $$\arg\min_{\phi} \mathbb{E}_{s\in H}\left[D_{KL}\left(\pi_\phi || \frac{\exp[\eta Q_\theta]}{Z}\right)[s]\right]$$
   replace max with softmax

Soft actor-critic (SAC) [Haarnoja, 2018]





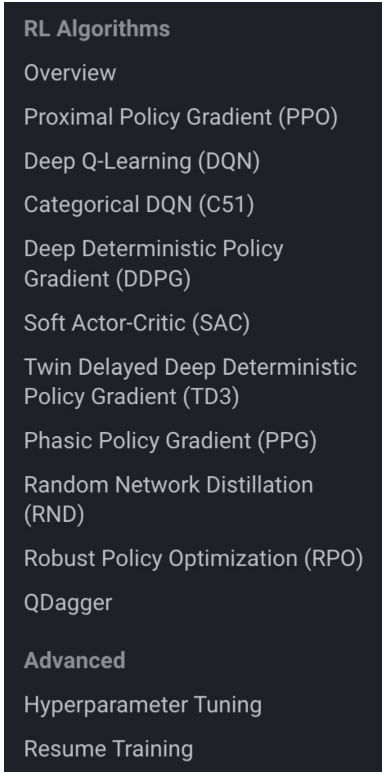


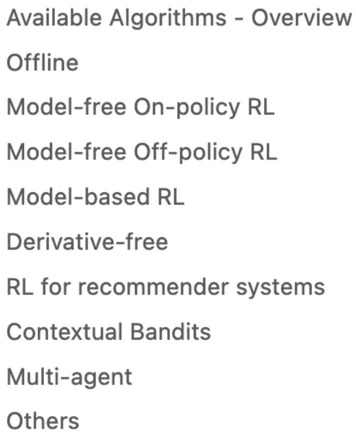
Further reading

- Soft policy iteration and soft actor-critic
  - T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *ICML, 2018*.
  - Blog post: https://yzhang1918.github.io/posts/sac/
- Soft Q-learning
  - Haarnoja T., Tang H., Abbeel P., Levine S, "Reinforcement Learning with Deep Energy-Based Policies," *ICML 2017*.
  - Blog post: https://bair.berkeley.edu/blog/2017/10/06/soft-q-learning/

Implementations of RL algorithms

- For research and prototyping:
  - CleanRL: https://docs.cleanrl.dev/
  - A Deep Reinforcement Learning library that provides high-quality single-file implementation with research-friendly features
- For scaling up:
  - RLlib: https://docs.ray.io/en/latest/rllib/index.html
  - Industry-grade reinforcement learning
  - Built on distributed execution engine Ray





Recap

- Policy gradient methods offer a conceptually simple class of methods for reinforcement learning.
- They work by directly optimizing the policy (rather than the value function) by approximating the gradient of the value function.
- Policy gradient methods attempt to maximize the likelihood of good trajectories.
- The policy gradient theorem enables us to estimate the gradient through Monte Carlo trajectory samples (REINFORCE algorithm).
- Advantages: no Markovian assumption, often effective for continuous action space problems.
- Disadvantages: high variance and on-policy (limited sample efficiency).
- A variety of approaches help to reduce variance: temporal structure, baselines, incorporate a critic.
- Core practical policy gradient / actor-critic methods: REINFORCE, SAC, TRPO, PPO (next lecture).

References

1. Matteo Pirotta. FAIR. Reinforcement Learning. 2019, Lecture 5.
2. Matteo Pirotta. Reinforcement Learning Summer School, 2019. Policy Search: Actor-Critic Methods.