facebook
Artificial Intelligence Research

How to solve an MDP incrementally: Approximate Algorithms - Policy Gradient

Matteo Pirotta
Facebook AI Research

Learning in an MDP

Goal: learning the optimal policy $\pi^\star$ of an MDP

- Tabular MDP, known dynamics
- Tabular MDP, unknown dynamics
- Large or Continuous MDP, known dynamics
- Large or Continuous MDP, unknown dynamics

Learning in an MDP

Goal: learning the optimal policy $\pi^\star$ of an MDP

- Tabular MDP, known dynamics
  Dynamic Programming
- Tabular MDP, unknown dynamics
- Large or Continuous MDP, known dynamics
- Large or Continuous MDP, unknown dynamics

Learning in an MDP

Goal: learning the optimal policy $\pi^\star$ of an MDP

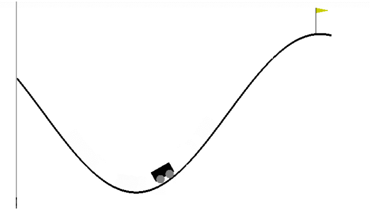
- Tabular MDP, known dynamics
  Dynamic Programming
- Tabular MDP, unknown dynamics
  Q-Learning, SARSA
- Large or Continuous MDP, known dynamics
- Large or Continuous MDP, unknown dynamics

Learning in an MDP

Goal: learning the optimal policy $\pi^\star$ of an MDP

- Tabular MDP, known dynamics
  Dynamic Programming
- Tabular MDP, unknown dynamics
  Q-Learning, SARSA
- Large or Continuous MDP, known dynamics
  ?
- Large or Continuous MDP, unknown dynamics
  ?

Example: Mountain Car



State : $(x, \dot{x}) \in [-1.2; 0.6] \times [-0.07; 0.07]$

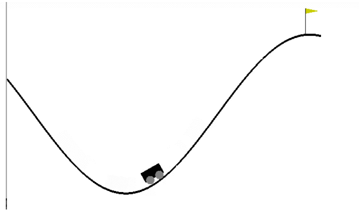
Actions : $\mathcal{A} = \{-1, 0, 1\}$: full throttle reverse / zero throttle / full throttle forward

Reward : always -1 except in the terminal (goal) state $x_\star = 0.6$

Dynamics : when doing action a_t in state $s_t = (x_t, v_t)$, the next state $s_{t+1} = (x_{t+1}, v_{t+1})$ is

$$\begin{cases} v_{t+1} & = & \max\{\min\{v_t + \epsilon_t + 0.001 a_t - 0.0025 \cos(3x_t), 0.07\}, -0.07\}, \\ x_{t+1} & = & \max\{\min\{x_t + v_t, 0.6\}, -1.2\}. \end{cases}$$

Example: Mountain Car



State : $(x, \dot{x}) \in [-1.2; 0.6] \times [-0.07; 0.07]$

Actions : $\mathcal{A} = \{-1, 0, 1\}$: full throttle reverse / zero throttle / full throttle forward

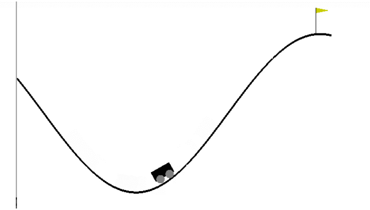
Reward : always -1 except in the terminal (goal) state $x_\star = 0.6$

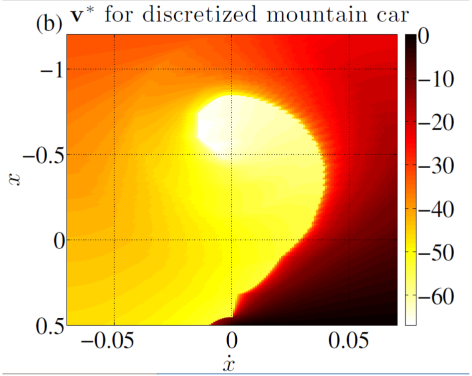
Dynamics : when doing action a_t in state $s_t = (x_t, v_t)$, the next state $s_{t+1} = (x_{t+1}, v_{t+1})$ is

$$\begin{cases} v_{t+1} &= \max\{\min\{v_t + \epsilon_t + 0.001 a_t - 0.0025\cos(3x_t), 0.07\}, -0.07\}, \\ x_{t+1} &= \max\{\min\{x_t + v_t, 0.6\}, -1.2\}. \end{cases}$$

$\Longrightarrow$ continuous MDP with known model

Example: Mountain Car





From Exact to Approximate RL

- DP and RL algorithms require an *exact* representation of value functions and policies

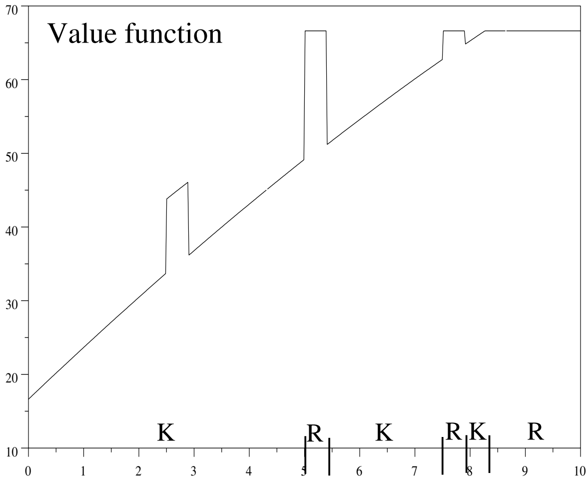
From Exact to Approximate RL

- DP and RL algorithms require an *exact* representation of value functions and policies
- This is often *impossible* since their shape is too "complicated" (e.g., large or continuous state space).

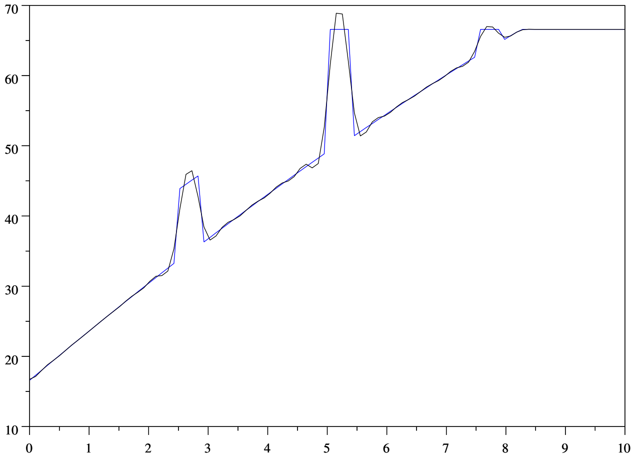
From Exact to Approximate RL

- DP and RL algorithms require an *exact* representation of value functions and policies
- This is often *impossible* since their shape is too "complicated" (e.g., large or continuous state space).
- **Can we use approximations?**

From Exact to Approximate RL



From Exact to Approximate RL



Approximated by a Fourier basis expansion

What to approximate?

- Value Function

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

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

- Policy

$$\pi : \mathcal{S} \to \Delta(\mathcal{A})$$

How? Value function approximation

From an estimate of $V^\star$ to an estimate of $Q^\star$

$$Q^\star \to V^\star(s) = \max_a Q^\star(s,a) \qquad \text{easy}$$

$$V^\star \to Q^\star(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) V^\star(s') \qquad \text{possibly complicated}$$

Policy Computation

$$\pi(s) = \arg\max_a Q(s,a)$$

$$\pi(s) = \arg\max_a r(s,a) + \gamma \sum_{s'} p(s'|s,a) V^\star(s')$$

👉 decide when to approximate $V^\star$ or $Q^\star$
($Q^\star$ **is more handy to get a policy, but more parameter to learn**)

How? Value function approximation

Problem: Often $\mathcal{S}$ is too large to store a vector V or a table Q in memory...

Solution: look for estimates V (resp. Q) of $V^\star$ (resp. $Q^\star$) in an approximation space $\mathcal{F}_V$ (resp. $\mathcal{F}_Q$)

Parametric approximation

$$\mathcal{F}_V = \{s \mapsto V_\theta(s) | \theta \in \Theta\} \qquad \mathcal{F}_Q = \{(s,a) \mapsto Q_\theta(s,a) | \theta \in \Theta\}$$

only requires to store a parameter θ (typically $\theta \in \mathbb{R}^d$, $d << S$)

☞ Smooth parameterization if $\nabla_\theta V_\theta(s)$ (resp. $\nabla_\theta Q_\theta(s,a)$) can be computed

How? Value function approximation

Linear function approximation

$$\mathcal{F}_V = \left\{ s \mapsto V_\theta(s) = \sum_{i=1}^{d} \theta_i \phi_i(s) | \theta \in \mathbb{R}^d \right\}$$

Let $\phi(s)$ be the *feature vector* of a state s

$$\phi(s) = (\phi_1(s), \ldots, \phi_d(s))^\mathsf{T} \in \mathbb{R}^d$$

then

$$V_\theta(s) = \theta^\mathsf{T} \phi(s)$$

Remarks:

- smooth parameterization with $\nabla_\theta V_\theta(s) = \phi(s)$
- if $\mathcal{S} = \{s_1, \ldots, s_S\}$, we recover the tabular case with $\phi_i(s) = \mathbb{1}(s = s_i)$ for $s \in [S]$

How? Value function approximation

Linear function approximation

$$\mathcal{F}_Q = \left\{ (s,a) \mapsto Q_\theta(s) = \sum_{i=1}^{d} \theta_i \phi_i(s,a) | \theta \in \mathbb{R}^d \right\}$$

Let $\phi(s)$ be the *feature vector* of a state s

$$\phi(s,a) = (\phi_1(s,a), \dots, \phi_d(s,a))^\mathsf{T} \in \mathbb{R}^d$$

then

$$Q_\theta(s,a) = \theta^\mathsf{T} \phi(s,a)$$

Remarks:

- smooth parameterization with $\nabla_\theta Q_\theta(s,a) = \phi(s,a)$
- we can still recover the tabular case

How? Value function approximation

Non-Linear function approximation

Linear function approximation requires to design (meaningful) features, which can be hard . . .

Modeling V (or Q) as a neural network can be more powerful :

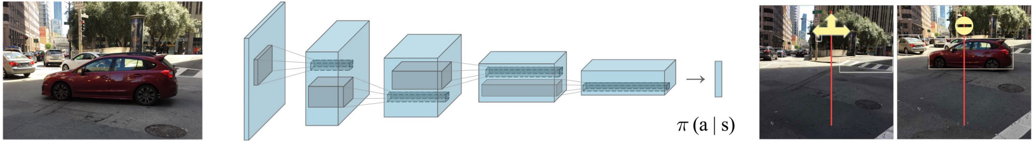
- neural networks are known to be universal approximators
- they "learn features" from the data
- and gradient can be computed efficiently

☞ using neural networks for RL is an old idea, making it work is art
(**main problem is to provide good samples, i.e., explore the system**)

How? Policy approximation

$$\mathcal{F}_{\Pi} = \left\{ (s,a) \mapsto \pi_\theta(a|s) \mid \theta \in \Theta \right\}$$

- deterministic vs. stochastic policy
- discrete actions vs. continuous actions



How? Policy approximation

Normal Policy

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

with

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

Softmax Policy (κ inverse temperature)

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

with

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

Approximate Policy-Based Algorithms

Policy Learning

Outline

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

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} $\implies$ Is it still converging?
Continuous actions?

From Policy Iteration to Policy Search

- Approximate a *stochastic policy* directly using function approximation

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

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

› *Policy optimization problem*

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

From Policy Iteration to Policy Search

- Approximate a *stochastic policy* directly using function approximation

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

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

> *Policy optimization problem*

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

Solution 1: **Policy Search/Black-box optimization:**

Use global optimizers or gradient by finite-difference methods

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

From Policy Iteration to Policy Search

- Approximate a *stochastic policy* directly using function approximation

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

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

> *Policy optimization problem*

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

Solution 1: **Policy Search/Black-box optimization:**

Use global optimizers or gradient by finite-difference methods

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

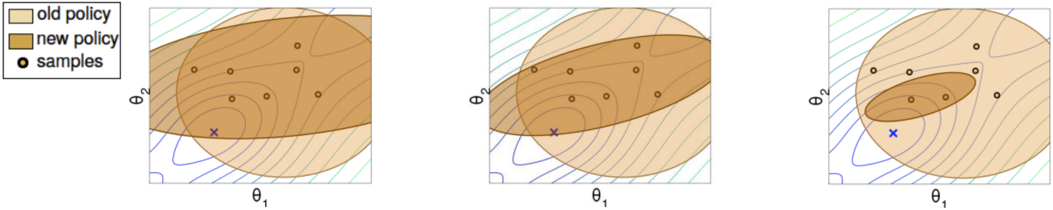
Solution 2: **Policy gradient optimization:**

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

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

Desired Properties for the Policy Update

- *Invariance* to parameter or reward transformations
- *Regularized policy update*
  - Update is computed based on data
    $\Longrightarrow$ *stay close to data!*
  - Smooth learning progress
- *Controllable exploration-exploitation trade-off*



Conservative Update
Small "step size"

Moderate Update,
Moderate "step size"

Greedy update
Large "step size"

Policy Gradient as Policy Update

Approximate Policy Iteration

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

Unstable (fast)

Policy Gradient

$$\theta_{k+1} = \theta_k + \alpha_k \nabla J(\theta_k)$$

Smooth, fine control (slow)

How do we compute $\nabla_\theta J(\theta)$?

Outline

Policy Gradient: finite-horizon

Given an MDP $M = (\mathcal{S}, \mathcal{A}, p, r, H, \rho)$ and a policy π

$$J(\pi) = \mathbb{E}\left[\sum_{t=1}^{H} r_t | \pi, M\right] = \mathbb{E}_{\tau \sim \mathbb{P}(\tau|\pi,M)}\left[\mathcal{R}(\tau)\right]$$

where $\tau = (s_1, a_1, r_1, \ldots, s_{H+1})$ is a trajectory and $R(\tau)$ its return (sum of returns).

Policy Gradient: finite-horizon

Theorem ([Williams, 1992, Sutton et al., 2000])

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

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

Proof

- The objective is an *expectation*. Want to compute the gradient w.r.t. θ

$$
\begin{aligned}
\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_\tau[R(\tau)] &= \nabla_\theta \int \mathbb{P}(\tau|\theta) R(\tau) \mathrm{d}\tau \\
&= \int \nabla_\theta \mathbb{P}(\tau|\theta) R(\tau) \mathrm{d}\tau \\
&= \int \mathbb{P}(\tau|\theta) \; \nabla_\theta \log \mathbb{P}(\tau|\theta) \; R(\tau) \mathrm{d}\tau \\
&= \mathbb{E}_\tau[R(\tau) \nabla_\theta \log \mathbb{P}(\tau|\theta)]
\end{aligned}
$$

log trick

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

Proof

- The objective is an *expectation*. Want to compute the gradient w.r.t. θ

$$
\begin{aligned}
\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_\tau[R(\tau)] &= \nabla_\theta \int \mathbb{P}(\tau|\theta) R(\tau) \mathrm{d}\tau \\
&= \int \nabla_\theta \mathbb{P}(\tau|\theta) R(\tau) \mathrm{d}\tau \\
&= \int \mathbb{P}(\tau|\theta) \ \nabla_\theta \log \mathbb{P}(\tau|\theta) \ R(\tau) \mathrm{d}\tau \\
&= \mathbb{E}_\tau[R(\tau) \nabla_\theta \log \mathbb{P}(\tau|\theta)]
\end{aligned}
$$

log trick

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

- Last expression is an *unbiased* gradient estimator.
  Just sample $\tau_i \sim \mathbb{P}(\tau|\theta)$, and compute $\widehat{g}_i = R(\tau_i) \nabla_\theta \log \mathbb{P}(\tau|\theta)$

Proof

- The objective is an *expectation*. Want to compute the gradient w.r.t. θ

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_\tau[R(\tau)] = \nabla_\theta \int \mathbb{P}(\tau|\theta) R(\tau) \mathrm{d}\tau$$
$$= \int \nabla_\theta \mathbb{P}(\tau|\theta) R(\tau) \mathrm{d}\tau$$
$$= \int \mathbb{P}(\tau|\theta)\ \nabla_\theta \log \mathbb{P}(\tau|\theta)\ R(\tau) \mathrm{d}\tau$$
$$= \mathbb{E}_\tau[R(\tau) \nabla_\theta \log \mathbb{P}(\tau|\theta)]$$

log trick

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

- Last expression is an *unbiased* gradient estimator.
  Just sample $\tau_i \sim \mathbb{P}(\tau|\theta)$, and compute $\widehat{g}_i = R(\tau_i)\nabla_\theta \log \mathbb{P}(\tau|\theta)$
- Need to be able to *compute and differentiate the density* $\mathbb{P}(\tau|\theta)$ w.r.t. θ

Proof

Likelihood (*with stochastic policies*)

$$\mathbb{P}(\tau|\pi, M) = \rho(s_1) \prod_{i=1}^{H} \pi(s_i, a_i) p(s_{i+1}|s_i, a_i)$$

$$\log \mathbb{P}(\tau|\pi, M) = \log \rho(s_1) + \sum_{i=1}^{H} \log \pi(s_i, a_i) + \log p(s_{i+1}|s_i, a_i)$$

$$\nabla_\theta \log \mathbb{P}(\tau|\pi, M) = \cancelto{0}{\nabla_\theta \log \rho(s_1)} + \sum_{i=1}^{H} \left(\nabla_\theta \log \pi(s_i, a_i) + \cancelto{0}{\nabla_\theta \log p(s_{i+1}|s_i, a_i)} \right)$$

REINFORCE

1. Let π_{θ_1} be an arbitrary policy
2. At each iteration $k = 1, \dots, K$
   - Sample m trajectory $\tau_i = (s_1, a_1, r_1, s_2, \dots, s_T, a_T, r_T, s_{T+1})$ following π_k
   - Compute unbiased gradient estimate

$$\widehat{\nabla_\theta J}(\pi_{\theta_k}) = \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=1}^{H} r_t^i \right) \left(\sum_{t=1}^{H} \nabla_\theta \log \pi_{\theta_k}(s_t, a_t) \right)$$

   - Update parameters

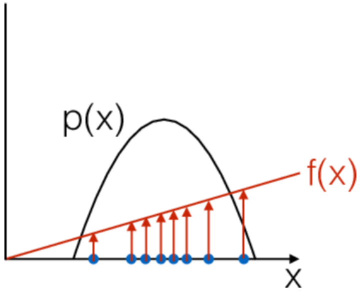
$$\theta_{k+1} = \theta_k + \alpha_k \widehat{\nabla_\theta J}(\pi_{\theta_k})$$

3. Return last policy π_{θ_K}

REINFORCE as *Supervised Learning*

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

- $R(\tau_i)$ measures how *good* is sample τ_i
- Moving in the direction of $\widehat{g}_i$ pushes up the log probability of the sample, in proportion to how good it is

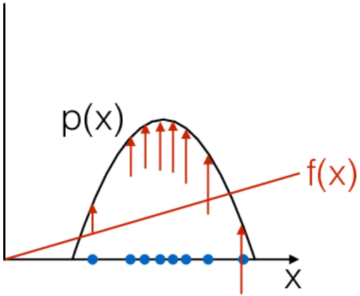


[Schulman, 2016]

REINFORCE as *Supervised Learning*

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

- $R(\tau_i)$ measures how *good* is sample τ_i
- Moving in the direction of $\widehat{g}_i$ pushes up the log probability of the sample, in proportion to how good it is

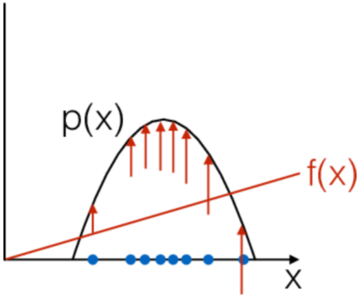


[Schulman, 2016]

REINFORCE as *Supervised Learning*

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

- $R(\tau_i)$ measures how *good* is sample τ_i
- Moving in the direction of $\widehat{g}_i$ pushes up the log probability of the sample, in proportion to how good it is



[Schulman, 2016]

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

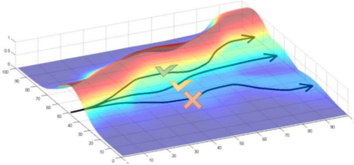


image from "CS 294-112: Deep Reinforcement Learning" slides by S. Levine

REINFORCE

Pros

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

REINFORCE

Pros

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

Issues

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

Policy Gradient: temporal structure

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}\left[\sum_{t=1}^{H} \nabla_\theta \log \pi_\theta(s_t, a_t) \sum_{t'=t}^{H} r_{t'}\right]$$

Policy Gradient: temporal structure

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}\left[\sum_{t=1}^{H} \nabla_\theta \log \pi_\theta(s_t, a_t) \sum_{t'=t}^{H} r_{t'}\right]$$

$$\begin{aligned}
\mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s_t, a) \sum_{t'=1}^{t-1} r_i \middle| \tau_{1:t-1}\right] &= \left(\sum_{t'=1}^{t-1} r_i\right) \int \pi_\theta(s_t, a) \nabla_\theta \log \pi(s_t, a) \mathrm{d}a \\
&= \left(\sum_{t'=1}^{t-1} r_i\right) \int \nabla_\theta \pi(s_t, a) \mathrm{d}a \\
&= \left(\sum_{t'=1}^{t-1} r_i\right) \nabla_\theta \underbrace{\int \pi(s_t, a) \mathrm{d}a}_{:=1} = 0
\end{aligned}$$

in literature known as G(PO)MDP [Peters and Schaal, 2008]

Policy Gradient: baseline

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

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

- The gradient estimate is unbiased
- *"Near optimal choice"* that minimize the variance is the expected sum of returns

$$b^\star(s_t) = \mathbb{E}\left[\sum_{t=1}^{T} r_t | s_1 = s_t, \pi, M\right]$$

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

Baseline derivation

Rough idea

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

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

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

baseline is unbiased in expectation

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

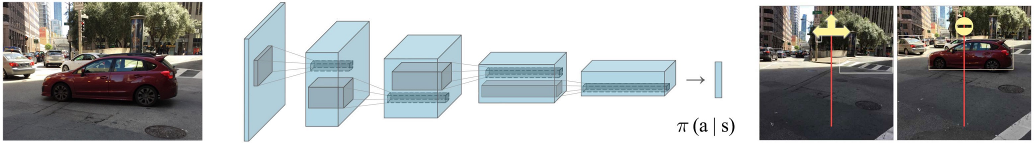
$$= \cancelto{0}{\frac{\partial}{\partial b}\mathbb{E}_\tau[g(\tau)^2 R(\tau)^2]} - 2\frac{\partial}{\partial b}\mathbb{E}_\tau[g(\tau)^2 R(\tau)\ b] + \frac{\partial}{\partial b}\mathbb{E}_\tau[b^2 g(\tau)^2]$$

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

Expected return weighted by the magnitude of the gradient

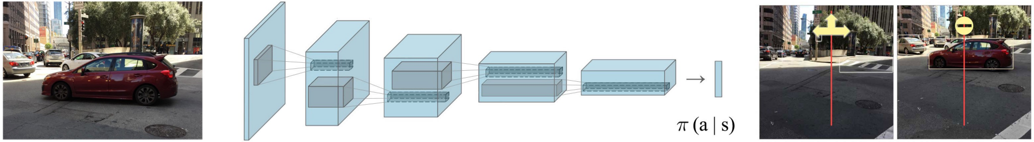
Policy Gradient: example

How do we represent a policy?



Policy Gradient: example

How do we represent a policy?



Normal Policy

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

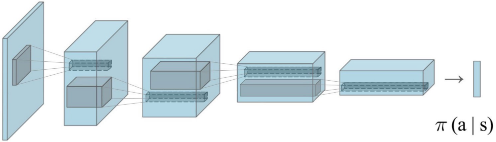
then

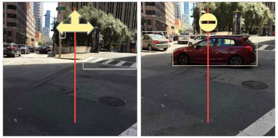
$$\nabla_\theta \log \pi(a|s) = \frac{(a-\mu_\theta(s))}{\sigma_\omega^2(s)} \nabla_\theta \mu_\theta(s)$$

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

Policy Gradient: example

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

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

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

Gibbs (softmax) policy

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

then

$$\begin{aligned}\nabla_\theta \log \pi(a|s) =& \kappa \nabla_\theta Q_\theta(s,a) \\ & - \kappa \sum_{a' \in \mathcal{A}} \pi(a'|s) \nabla_\theta Q_\theta(s,a')\end{aligned}$$

Policy Gradient via Automatic Differentiation

- Manually code the derivative can be tedious
  $\implies$ use auto diff
- Define a graph such that its gradient is the policy gradient

"Pseudo loss": weighted maximum likelihood

$$\widetilde{J} = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{T} \log \pi_\theta(s_{i,t}, a_{i,t}) \widehat{q}_{i,t}$$

where $\widehat{q}_{i,t} = \sum_{k=1}^{T_i} r_{i,k}$ for REINFORCE and $\widehat{q}_{i,t} = \sum_{k=t}^{T_i} r_{i,k}$ for G(PO)MDP.

Note that $\mathbb{E}\left[\nabla_\theta \widetilde{J}\right] = \nabla_\theta J(\pi_\theta)$

Outline

Going beyond the finite-horizon case

Theorem

For an infinite horizon MDP (average or discounted), the policy gradient is

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim\pi_\theta(s,\cdot)}\left[\nabla_\theta \log \pi_\theta(s,a) q^\pi(s,a)\right]$$

- d^π is the stationary distribution
- q^π is the state-action value function

Infinite-horizon discounted

- Define a *distribution* ρ over $\mathcal{S}$
- The *γ-discounted visitation frequency* for policy π is

$$d^\pi(s) = \lim_{T\to+\infty} \sum_{t=1}^{T} \gamma^{t-1} \mathbb{P}(s_t = s|\pi, M, \rho)$$

- Then

$$q^\pi(s,a) = \lim_{T\to+\infty} \mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} r(s_t,a_t)|s_1 = s, a_1 = a, \pi, M\right]$$

$$v^\pi(s) = \lim_{T\to+\infty} \mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} r(s_t,a_t)|s_1 = s, \pi, M\right] = \sum_a \pi(s,a) q^\pi(s,a)$$

$$J(\pi) = \lim_{T\to+\infty} \mathbb{E}[\sum_{t=1}^{T} \gamma^{t-1} r(s_t,a_t)|\pi, M, \rho]$$

$$= \sum_s d^\pi(s) \sum_a \pi(s,a) r(s,a) = \sum_s \rho(s) v^\pi(s)$$

Policy Gradient: proof

Bellman Equation

$$q^\pi(s,a) = r(s,a) + \sum_y p(y|s,a) v^\pi(y)$$

$$\boxed{\begin{aligned} \nabla_\theta\ v^\pi(s) &= \sum_a q^\pi(s,a)\nabla_\theta \pi(s,a) + \pi(s,a)\nabla_\theta q^\pi(s,a) \\ &= \sum_a q^\pi(s,a)\nabla_\theta \pi(s,a) + \underbrace{\gamma \sum_a \pi(s,a) \sum_y p(y|s,a)\nabla_\theta v^\pi(y)}_{\text{B}} \end{aligned}}$$

Bellman equation for the gradient!

Policy Gradient: proof

Multiply by $d^\pi(s)$ and sum over states

$$\text{Ⓑ} = \sum_s d^\pi(s)\gamma \sum_{a,y} \pi(s,a) p(y|s,a) \nabla_\theta v^\pi(y)$$

$$= \sum_s \sum_{k=0}^{+\infty} \gamma^k \mathbb{P}(s_1 \to s, k, \pi)\gamma \sum_{a,y} \pi(s,a) p(y|s,a) \nabla_\theta v^\pi(y)$$

Policy Gradient: proof

Multiply by $d^\pi(s)$ and sum over states

$$\text{Ⓑ} = \sum_s d^\pi(s)\gamma \sum_{a,y} \pi(s,a)p(y|s,a)\nabla_\theta v^\pi(y)$$

$$= \sum_s \sum_{k=0}^{+\infty} \gamma^k \mathbb{P}(s_1 \to s, k, \pi)\gamma \sum_{a,y} \pi(s,a)p(y|s,a)\nabla_\theta v^\pi(y)$$

Policy Gradient: proof

Multiply by $d^\pi(s)$ and sum over states

$$
\begin{aligned}
Ⓑ &= \sum_s d^\pi(s) \gamma \sum_{a,y} \pi(s,a) p(y|s,a) \nabla_\theta v^\pi(y) \\
&= \sum_s \sum_{k=0}^{+\infty} \gamma^k \mathbb{P}(s_1 \to s, k, \pi) \gamma \sum_{a,y} \pi(s,a) p(y|s,a) \nabla_\theta v^\pi(y) \\
&= \sum_y \left(\sum_{k=0}^{+\infty} \gamma^{k+1} \mathbb{P}(s_1 \to y, k+1, \pi) \right) \nabla_\theta v^\pi(y)
\end{aligned}
$$

Policy Gradient: proof

Multiply by $d^\pi(s)$ and sum over states

$$
\begin{aligned}
\text{Ⓑ} &= \sum_s d^\pi(s)\gamma \sum_{a,y} \pi(s,a)p(y|s,a)\nabla_\theta v^\pi(y) \\
&= \sum_s \sum_{k=0}^{+\infty} \gamma^k \mathbb{P}(s_1 \to s, k, \pi)\gamma \sum_{a,y} \pi(s,a)p(y|s,a)\nabla_\theta v^\pi(y) \\
&= \sum_y \left(\sum_{k=0}^{+\infty} \gamma^{k+1}\mathbb{P}(s_1 \to y, k+1, \pi) \pm \mathbb{P}(s_1 \to y, 0, \pi) \right) \nabla_\theta v^\pi(y)
\end{aligned}
$$

Policy Gradient: proof

Multiply by $d^\pi(s)$ and sum over states

$$
\begin{aligned}
Ⓑ &= \sum_s d^\pi(s)\gamma \sum_{a,y} \pi(s,a)p(y|s,a)\nabla_\theta v^\pi(y) \\
&= \sum_s \sum_{k=0}^{+\infty} \gamma^k \mathbb{P}(s_1 \to s, k, \pi)\gamma \sum_{a,y} \pi(s,a)p(y|s,a)\nabla_\theta v^\pi(y) \\
&= \sum_y \left(\sum_{k=0}^{+\infty} \gamma^{k+1} \mathbb{P}(s_1 \to y, k+1, \pi) \pm \mathbb{P}(s_1 \to y, 0, \pi) \right) \nabla_\theta v^\pi(y) \\
&= \sum_y \Big(d^\pi(y) - \underbrace{\mathbb{P}(s_1 \to y, 0, \pi)}_{:=\rho(y)} \Big) \nabla_\theta v^\pi(y)
\end{aligned}
$$

Policy Gradient: proof

Multiply by $d^\pi(s)$ and sum over states

$$
\begin{aligned}
Ⓑ &= \sum_s d^\pi(s)\gamma \sum_{a,y} \pi(s,a)p(y|s,a)\nabla_\theta v^\pi(y) \\
&= \sum_s \sum_{k=0}^{+\infty} \gamma^k \mathbb{P}(s_1 \to s, k, \pi)\gamma \sum_{a,y} \pi(s,a)p(y|s,a)\nabla_\theta v^\pi(y) \\
&= \sum_y \left(\sum_{k=0}^{+\infty} \gamma^{k+1}\mathbb{P}(s_1 \to y, k+1, \pi) \pm \mathbb{P}(s_1 \to y, 0, \pi) \right) \nabla_\theta v^\pi(y) \\
&= \sum_y \left(d^\pi(y) - \underbrace{\mathbb{P}(s_1 \to y, 0, \pi)}_{:=\rho(y)} \right) \nabla_\theta v^\pi(y)
\end{aligned}
$$

Summing up everything

$$
\cancel{\sum_s d^\pi(s)\nabla_\theta v^\pi(s)} = \sum_{s,a} d^\pi(s)\nabla_\theta \pi(s,a) q^\pi(s,a) + \cancel{\sum_y d^\pi(y)\nabla_\theta v^\pi(y)} - \underbrace{\nabla_\theta \sum_y \rho(y) v^\pi(y)}_{\nabla_\theta J(\pi)}
$$

REINFORCE for infinite horizon

1. Collect m trajectories for policy π starting from $s_1 \sim \rho$
2. For each time t

$$\widehat{q}_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$$

(*almost*) unbiased estimate $\rightarrow \mathbb{E}[\widehat{q}|s_t, a_t] = q^\pi(s_t, a_t)$

Then

$$\overline{\nabla_\theta J}(\pi_\theta) := \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{T} \gamma^{t-1} \nabla_\theta \log \pi_\theta(s_{i,t}, a_{i,t}) \sum_{t'=t}^{T} \gamma^{t'-t} r_{i,t'}$$

REINFORCE for infinite horizon

- Define $F_t := \widehat{q}_t \nabla_\theta \log \pi_\theta(s_t, a_t)$

$$
\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^{+\infty} \gamma^{t-1} F_t\right] &= \sum_{t=1}^{+\infty} \gamma^{t-1} \sum_{s} \mathbb{E}[F_t | s_t = s] \mathbb{P}(s_t = s | s_1 \sim \rho) \\
&= \sum_{s,a} q^\pi(s,a) \nabla_\theta \pi(s,a) \underbrace{\sum_{t=1}^{+\infty} \gamma^{t-1} \mathbb{P}(s_t = s | s_1 \sim \rho)}_{:= d^\pi(s)} \\
&= \nabla_\theta J(\pi)
\end{aligned}
$$

- *Almost unbiased* (T vs. $+\infty$)
- We can introduce a *baseline* $b(s_t)$ also in this case

Gradient in Practice

Finite-Horizon γ-discounted setting

$$J_\gamma(\pi) = \mathbb{E}\left[\sum_{t=1}^{H} \gamma^{t-1} r_t\right]$$

$$\nabla_\theta J_\gamma(\pi) = \mathbb{E}\left[\sum_{t=1}^{H} \gamma^{t-1} \nabla_\theta \log \pi_\theta(s_t, a_t) q^\pi(s_t, a_t)\right]$$

Gradient in Practice

Finite-Horizon γ-discounted setting

$$J_\gamma(\pi) = \mathbb{E}\left[\sum_{t=1}^{H} \gamma^{t-1} r_t\right]$$

$$\nabla_\theta J_\gamma(\pi) = \mathbb{E}\left[\sum_{t=1}^{H} \gamma^{t-1} \nabla_\theta \log \pi_\theta(s_t, a_t) q^\pi(s_t, a_t)\right]$$

In practice

$$\nabla_\theta J^?(\pi) = \mathbb{E}\left[\sum_{t=1}^{H} \cancelto{1}{\gamma^{t-1}} \nabla_\theta \log \pi_\theta(s_t, a_t) q^\pi(s_t, a_t)\right]$$

$\nabla_\theta J^?(\pi)$ is a **semi-gradient** of the *undiscounted* objective $J(\pi)$

Gradient in practice

$$J(\pi) = \mathbb{E}\left[\sum_{t=1}^{H} r_t\right] \quad \mapsto \quad \nabla_\theta J(\pi) = \underbrace{\sum_s d_\gamma^\pi(s) \frac{\partial}{\partial\theta} v_\gamma^\pi(s)}_{:=\nabla_\theta J^?(\pi)} + \sum_s v_\gamma^\pi(s) \frac{\partial}{\partial\theta} d_\gamma^\pi(s)$$

! TD(0) step is also a semi-gradient of the mean squared Bellman error [Sutton and Barto, 2018, Chapter 9]

- In *tabular settings*, semi-gradient TD(0) converges to a minimum of the mean squared error [Jaakkola et al., 1994]
- Also *on-policy* TD with linear function approximatio [Sutton and Barto, 2018]

Gradient in practice

$$J(\pi) = \mathbb{E}\left[\sum_{t=1}^{H} r_t\right] \quad \mapsto \quad \nabla_\theta J(\pi) = \underbrace{\sum_s d_\gamma^\pi(s) \frac{\partial}{\partial \theta} v_\gamma^\pi(s)}_{:=\nabla_\theta J^?(\pi)} + \sum_s v_\gamma^\pi(s) \frac{\partial}{\partial \theta} d_\gamma^\pi(s)$$

! TD(0) step is also a semi-gradient of the mean squared Bellman error [Sutton and Barto, 2018, Chapter 9]

- In *tabular settings*, semi-gradient TD(0) converges to a minimum of the mean squared error [Jaakkola et al., 1994]
- Also *on-policy* TD with linear function approximatio [Sutton and Barto, 2018]

Semi-policy gradient may converge to a **BAD policy** w.r.t. both discounted and undiscounted objectives

Impossibility result [Nota and Thomas, 2019]:

$$\nexists f(\pi) \in C \text{ such that } \nabla_\theta J^?(\pi) = \frac{\partial}{\partial \theta} f(\pi)$$

Outline

Convergence Results

- Policy gradient is *stochastich gradient*

$$\theta_{k+1} = \theta_k + \alpha_k(\nabla J(\theta_k) + \text{noise})$$

- J is **non-convex**
- $\implies$ converge asymptotically to a stationary point or a **local minimum** (*under standard technical assumptions*)

Convergence Results

- Policy gradient is *stochastich gradient*

$$\theta_{k+1} = \theta_k + \alpha_k(\nabla J(\theta_k) + \text{noise})$$

- J is **non-convex**
- $\implies$ converge asymptotically to a stationary point or a **local minimum** (*under standard technical assumptions*)

what is the *quality* of this point?

Convergence Results

- Policy gradient is *stochastich gradient*

$$\theta_{k+1} = \theta_k + \alpha_k (\nabla J(\theta_k) + \text{noise})$$

- J is **non-convex**
- $\implies$ converge asymptotically to a stationary point or a **local minimum** (*under standard technical assumptions*)

what is the *quality* of this point?

Dynamics are linear (LQ systems) $\implies$ *global convergence [Fazel et al., 2018]*

Surprising since $\min_{\pi} J_{\mathrm{LQ}}(\pi)$ may be not convex, quasi-convex, and star-convex but (far from boundaries) J_{LQ} is "almost" smooth

Hints: use properties of functions that are gradient dominated

Convergence Results

Issues

- *Non-convexity of the loss function*
- *Unnatural policy parameterization:* parameters that are far in Euclidean distance may describe the same policy (*we will talk about this later*)
- *Insufficient exploration:* naive stochastic exploration
- *Large variance of stochastic gradients:* generally increases with the length of the horizon

Convergence Results

Issues

- *Non-convexity of the loss function*
- *Unnatural policy parameterization:* parameters that are far in Euclidean distance may describe the same policy (*we will talk about this later*)
- *Insufficient exploration:* naive stochastic exploration
- *Large variance of stochastic gradients:* generally increases with the length of the horizon

Solution:

$\Longrightarrow$ similar to LQ, we need structural assumptions [Bhandari and Russo, 2019]

See also [Zhang et al., 2019] for convergence results

Bibliography

Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. II*. Athena Scientific, 3rd edition, 2007.

Jalaj Bhandari and Daniel Russo. Global optimality guarantees for policy gradient methods. *CoRR*, abs/1906.01786, 2019.

Maryam Fazel, Rong Ge, Sham Kakade, and Mehran Mesbahi. Global convergence of policy gradient methods for the linear quadratic regulator. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 1466–1475. PMLR, 2018.

Tommi S. Jaakkola, Michael I. Jordan, and Satinder P. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6):1185–1201, 1994.

Chris Nota and Philip S. Thomas. Is the policy gradient a gradient? *CoRR*, abs/1906.07073, 2019.

Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21 (4):682–697, 2008.

John Schulman. Deep reinforcement learning: Policy gradients and q-learning. Technical report, Bay Area Deep Learning School, 2016.

Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*. MIT press Cambridge, 2 edition, 2018.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Kaiqing Zhang, Alec Koppel, Hao Zhu, and Tamer Başar. Global convergence of policy gradient methods to (almost) locally optimal policies. *arXiv preprint arXiv:1906.08383*, 2019.

Thank you!

facebook
Artificial Intelligence Research