facebook
Artificial Intelligence Research

Policy Search: Actor-Critic Methods

Matteo Pirotta

Facebook AI Research

Reinforcement Learning Summer School (RLSS)

I will add the parts presented on the whiteboard soon.

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

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^{π_k} $\implies$ Is it still converging?
Continuous actions?

Approximate Policy Iteration

Issue: is no longer guaranteed to converge!

Proposition

The asymptotic performance of the policies π_k generated by the API algorithm is related to the approximation error as:

$$\limsup_{k\to+\infty} \underbrace{\|v^\star - v^{\pi_k}\|_\infty}_{\text{performance loss}} \leq \frac{2\gamma}{(1-\gamma)^2} \limsup_{k\to+\infty} \underbrace{\|v_k - v^{\pi_k}\|_\infty}_{\text{approximation error}}$$

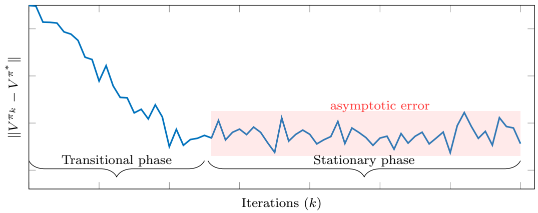
Approximate Policy Iteration

Issue: is no longer guaranteed to converge!

Proposition

The asymptotic performance of the policies π_k generated by the API algorithm is related to the approximation error as:

$$\limsup_{k\to+\infty} \underbrace{\|v^\star - v^{\pi_k}\|_\infty}_{\text{performance loss}} \leq \frac{2\gamma}{(1-\gamma)^2} \limsup_{k\to+\infty} \underbrace{\|v_k - v^{\pi_k}\|_\infty}_{\text{approximation error}}$$



Approximate Policy Iteration: Issues

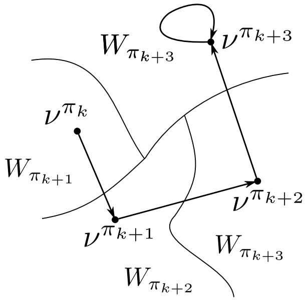
Potential pathologies in policy-iteration with function approximation

1. Exploration
2. Policy evaluation: bias, simulation bias/error
3. Policy improvement: policy oscillation
   - *local attractors*, e.g., local maxima

Approximate Policy Iteration: Issues

Potential pathologies in policy-iteration with function approximation

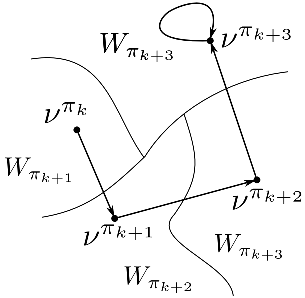
1. Exploration
2. Policy evaluation: bias, simulation bias/error
3. Policy improvement: policy oscillation
   - *local attractors*, e.g., local maxima

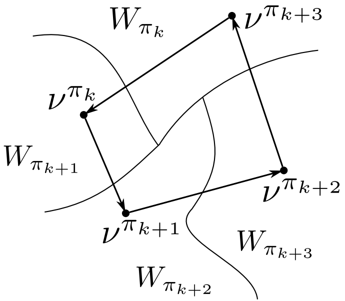


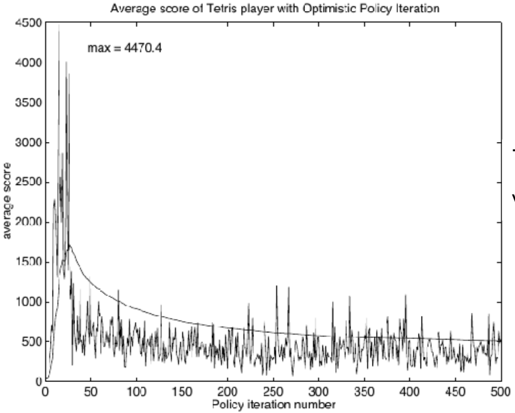
Approximate Policy Iteration: Issues

Potential pathologies in policy-iteration with function approximation

1. Exploration
2. Policy evaluation: bias, simulation bias/error
3. Policy improvement: policy oscillation
   - *local attractors*, e.g., local maxima







Tetris [Bertsekas and Ioffe, 1996]
very pathological [e.g., Scherrer et al., 2015]

Policy oscillation with linear function approximation [Koller and Parr, 2000, Lagoudakis and Parr, 2003a]

☞ poor policies

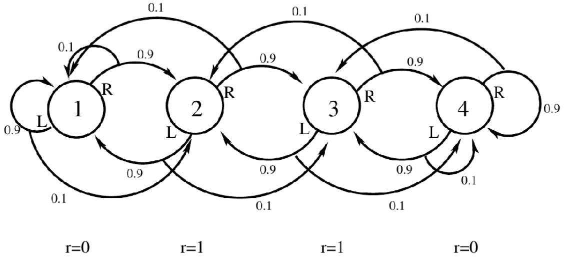


Figure 9: The problematic MDP.

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

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

- 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

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

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 $\nabla_\theta J(\theta)$?

(recap on optimality criteria)

Finite Horizon

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

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

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

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 $\widehat{g}_i = R(\tau_i) \nabla_\theta \log \mathbb{P}(\tau|\theta)$
- 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 $k = 1, \ldots, K$
   - Sample m trajectory $\tau_i = (s_1, a_1, r_1, s_2, \ldots, s_T, a_T, r_T, s_{T+1})$ 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

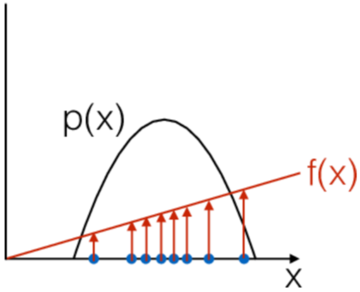
$$\theta_{k+1} = \theta_k + \alpha_k \widehat{\nabla_\theta J}(\pi_{\theta_k})$$

3. Return last policy π_{θ_K}

REINFORCE: Intuition

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

- $R(\tau_i)$ 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

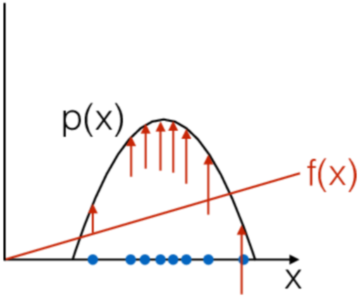


[Schulman, 2016]

REINFORCE: Intuition

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

- $R(\tau_i)$ 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

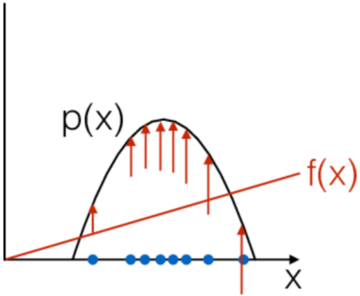


[Schulman, 2016]

REINFORCE: Intuition

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

- $R(\tau_i)$ 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 (TO REMOVE)
- Trial and Error approach

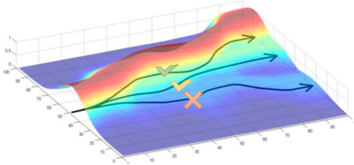


image from "CS 294-112: Deep Reinforcement Learning" slides by S. Levine

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, 2008b]

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)

Intuition: $b(s_t)$ does not depend on the action thus

$$\mathbb{E}_{a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(s_t, a) b(s_t) | \tau_{1:t-1}] = 0$$

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

Infinite Horizon

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

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

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

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

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

Policy Gradient: proof

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

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

Policy Gradient: proof

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

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

Policy Gradient: proof

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

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

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

Policy Gradient: example

$$\overline{\nabla_\theta J}(\pi_\theta) := \frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{T}\gamma^{t-1}\nabla_\theta \log \pi_\theta(s_{i,t}, a_{i,t}) \cdot \widehat{q}_{i,t}$$

How do we represent a policy?

Policy Gradient: example

$$\overline{\nabla_\theta J}(\pi_\theta) := \frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{T}\gamma^{t-1}\nabla_\theta \log \pi_\theta(s_{i,t}, a_{i,t}) \cdot \widehat{q}_{i,t}$$

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

$$\overline{\nabla_\theta J}(\pi_\theta) := \frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{T}\gamma^{t-1}\nabla_\theta \log \pi_\theta(s_{i,t}, a_{i,t}) \cdot \widehat{q}_{i,t}$$

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

$$\overline{\nabla_\theta J}(\pi_\theta) := \frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{T}\gamma^{t-1}\nabla_\theta \log \pi_\theta(s_{i,t}, a_{i,t}) \cdot \widehat{q}_{i,t}$$

- 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

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

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]:

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

(Example?)

Convergence Results

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 $\min_{\pi} J_{\mathsf{LQ}}(\pi)$ 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

Convergence Results: Structural Properties

[Bhandari and Russo, 2019]

Let $\Pi_\theta = \{\pi_\theta | \theta \in \Theta\}$ being the space of parametrized policies

1. Closure under policy improvement

$$\forall \pi \in \Pi_\theta, \quad \exists \pi^+ \in \Pi_\theta \qquad \text{s.t.} \qquad \pi^+ \in \arg\max q^\pi$$

2. Convexity of policy improvement steps

$$q^\pi(s,a) \text{ is convex in } a$$

3. Convexity of the policy class Π_θ
   soft policy-iteration update $(1-\alpha)\pi + \alpha\pi^+$ is feasible
4. Regularity conditions
   e.g., compactness of $\mathcal{S}$, existence and continuity of derivatives w.r.t. θ, etc.

Global convergence

- Consider the structural properties
- Consider infinite-horizon discounted problems

Global convergence

- Consider the structural properties
- Consider infinite-horizon discounted problems

No suboptimal stationary points by following a specific ascent direction

$\Longrightarrow$ **global convergence** [Bhandari and Russo, 2019]

Global convergence

- Consider the structural properties
- Consider infinite-horizon discounted problems

No suboptimal stationary points by following a specific ascent direction

$\Longrightarrow$ **global convergence** [Bhandari and Russo, 2019]

Idea:

$$\pi_{\theta_\alpha} := (1-\alpha)\pi_\theta + \alpha\pi_{\theta'} \in \Pi_\theta$$

$\alpha \in [0, 1]$ defines a line in the policy space
What is the direction to follow in the parameter space?

Global convergence

- Consider the structural properties
- Consider infinite-horizon discounted problems

No suboptimal stationary points by following a specific ascent direction

$\implies$ **global convergence** [Bhandari and Russo, 2019]

Idea:

$$\pi_{\theta_\alpha} := (1-\alpha)\pi_\theta + \alpha\pi_{\theta'} \in \Pi_\theta$$

$\alpha \in [0,1]$ defines a line in the policy space

What is the direction to follow in the parameter space?

find u such that the *directional derivative* of π' points in the direction of π' (smooth curve in the parameter space)

Follow the directional derivative between π_{θ_k} and π_k^+

Global convergence

- Consider the structural properties
- Consider infinite-horizon discounted problems

No suboptimal stationary points by following a specific ascent direction

$\implies$ **global convergence** [Bhandari and Russo, 2019]

Idea:

$$\pi_{\theta_\alpha} := (1-\alpha)\pi_\theta + \alpha\pi_{\theta'} \in \Pi_\theta$$

$\alpha \in [0,1]$ defines a line in the policy space
What is the direction to follow in the parameter space?
find u such that the *directional derivative* of π' points in the direction of π' (smooth curve in the parameter space)
Follow the directional derivative between π_{θ_k} and π_k^+
Forward connection: conservative policy iteration and adaptive gradient

Actor-Critic

REINFORCE

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

REINFORCE

- Monte-Carlo policy gradient is unbiased but *still* has high variance
- Define an alternative 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

- Actor-critic algorithms maintain two sets of parameters: $\theta \mapsto \pi$, $\omega \mapsto q^{\pi}$
- *Critic can use TD(0)*

---

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

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

Compute temporal difference

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

Update q estimate

$$\omega = \omega + \beta \delta_t \nabla_\omega q_\omega(x_t, a_t)$$

Update policy

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

end

TD(0) is a semi-gradient approach [Baird, 1995, Sutton, 2015]

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 $\nabla_\theta J(\pi_\theta)$

Solution:

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

Compatible Function Approximation

Theorem

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

$$q_\omega(s,a) = \omega^\intercal \nabla_\theta \log \pi_\theta(s,a)$$

If ω minimizes the squared Bellman residual

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

Then

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

Actor-Critic with a baseline

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

- $b(s)$ 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

- It is possible to estimate v^π and q^π *independently* (e.g., by TD(0))

Actor-Critic with advantage function

- It is possible to estimate v^π and q^π *independently* (e.g., by TD(0))
- $A^\pi = q_\omega - v_\nu$ is a **biased** and **unstable** estimate

Actor-Critic with advantage function

- It is possible to estimate v^π and q^π *independently* (e.g., by TD(0))
- $A^\pi = q_\omega - v_\nu$ is a **biased** and **unstable** estimate

Solution:

- Consider the temporal difference error

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

Actor-Critic with advantage function

- It is possible to estimate v^{π} and q^{π} *independently* (e.g., by TD(0))
- $A^{\pi} = q_{\omega} - v_{\nu}$ 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*

$$\mathbb{E}[\delta^{\pi_\theta}|s,a] = \mathbb{E}[r(s,a) + \gamma v^{\pi_\theta}(s')|s,a] - v^{\pi_\theta}(s) = q^{\pi_\theta}(s,a) - v^{\pi_\theta}(s)$$

Actor-Critic with advantage function

- Estimate **only** $v_\nu \mapsto \delta_\nu = r + \gamma v_\nu(s') - v_\nu(s)$

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

for $t = 1, \dots, T$ **do**

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

Compute temporal difference

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

Update v estimate

$$\nu = \omega + \beta \delta_t \nabla_\nu v_\nu(s_t)$$

Update policy

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

end

State-Action baseline (side note)

Several recent methods [Gu et al., 2017, Thomas and Brunskill, 2017, Grathwohl et al., 2018, Liu et al., 2018, Wu et al., 2018] have extended to **state-action baselines**

$$b(s) \to b(s, a)$$

👍 unbiased when *compatible function* approximation is used (proof?)

Is really working? See [Tucker et al., 2018] for complete investigation!

From online to batch actor-critic

- So far we have observed fully online actor-critic approaches
- In some case it can be *inefficient* (e.g., for training approximators)

$$\Longrightarrow \textit{batching}$$

From online to batch actor-critic

- So far we have observed fully online actor-critic approaches
- In some case it can be *inefficient* (e.g., for training approximators)

$\implies$ *batching*

1. Sample trajectories $\tau_i = \{s_1, a_1, r_1, \ldots, s_{T+1}\}$ using π_θ

$$\hat{v}(s_{i,t}) = \sum_{k=t}^{t+p} \gamma^{k-t} r_k + \gamma^p v_\nu(s_{t+p+1}) \qquad \textbf{bootstrapping}$$

From online to batch actor-critic

- So far we have observed fully online actor-critic approaches
- In some case it can be *inefficient* (e.g., for training approximators)

$\implies$ *batching*

1. Sample trajectories $\tau_i = \{s_1, a_1, r_1, \ldots, s_{T+1}\}$ using π_θ

$$\hat{v}(s_{i,t}) = \sum_{k=t}^{t+p} \gamma^{k-t} r_k + \gamma^p v_\nu(s_{t+p+1})$$ **bootstrapping**

2. Use supervised regression on $D = \{(s_{i,t}, \hat{v}(s_{i,t}))\}$

$$\arg\min_\nu \frac{1}{2} \sum_{(s,\hat{v})\in D} (v_\nu(s) - \hat{v})^2$$

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

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

- Use samples from other policies via *importance sampling* (*not very stable*)
- *Conservative approaches*
- Variance reduction techniques
- Newton or Quasi-newton methods

Off-policy Policy Gradient

- Usual approach [Wang et al., 2017]
  - Store observed samples (a.k.a. replay buffer)
  - *Off-policy* *policy evaluation* is "easy" (cf. LSTDQ [Lagoudakis and Parr, 2003a])
    $\pi_k \mapsto v^{\pi_k}$

Off-policy Policy Gradient

- Usual approach [Wang et al., 2017]
  - Store observed samples (a.k.a. replay buffer)
  - *Off-policy policy evaluation* is "easy" (cf. LSTDQ [Lagoudakis and Parr, 2003a])
    $\pi_k \mapsto v^{\pi_k}$

Issue:

- The estimate of the gradient requires samples from π_θ
- Use *importance ratios* to avoid introducing additional bias

Importance Weighting

$$\mathbb{E}_{x\sim p}[f(x)] = \mathbb{E}_{x\sim q}\left[\frac{p(x)}{q(x)}f(x)\right] \approx \mu_q = \frac{1}{N}\sum_{i=1}^{N}\frac{p(x_i)}{q(x_i)}f(x_i), \quad x_i \sim q$$

Importance Weighting

$$\mathbb{E}_{x\sim p}[f(x)] = \mathbb{E}_{x\sim q}\left[\frac{p(x)}{q(x)}f(x)\right] \approx \mu_q = \frac{1}{N}\sum_{i=1}^{N}\frac{p(x_i)}{q(x_i)}f(x_i), \quad x_i \sim q$$

Variance

$$\begin{aligned}\text{var}(\mu_q) &= \frac{1}{N}\text{var}\left(\frac{p(x)}{q(x)}f(x)\right)\\ &= \frac{1}{N}\left(\mathbb{E}_{x\sim p}\left[\frac{p(x)}{q(x)}f(x)^2\right] - \mathbb{E}_{x\sim p}[f(x)]^2\right)\end{aligned}$$

! The term in red may explode!

Importance Weighting in Policy Gradient

[Jurcícek, 2012, Degris et al., 2012]

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\beta}\left[\frac{\mathbb{P}(\tau|\pi_\theta)}{\mathbb{P}(\tau|\beta)}\sum_{t=1}^{T}\gamma^{t-1}\nabla_\theta \log \pi_\theta(s_t, a_t) q^{\pi_\theta}(s_t, a_t)\right]$$

❓ what's the issue?

Importance Weighting in Policy Gradient

[Jurcícek, 2012, Degris et al., 2012]

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\beta}\left[\frac{\mathbb{P}(\tau|\pi_\theta)}{\mathbb{P}(\tau|\beta)}\sum_{t=1}^{T}\gamma^{t-1}\nabla_\theta \log \pi_\theta(s_t, a_t) q^{\pi_\theta}(s_t, a_t)\right]$$

❓ what's the issue? *Exploding or vanishing importance weights*

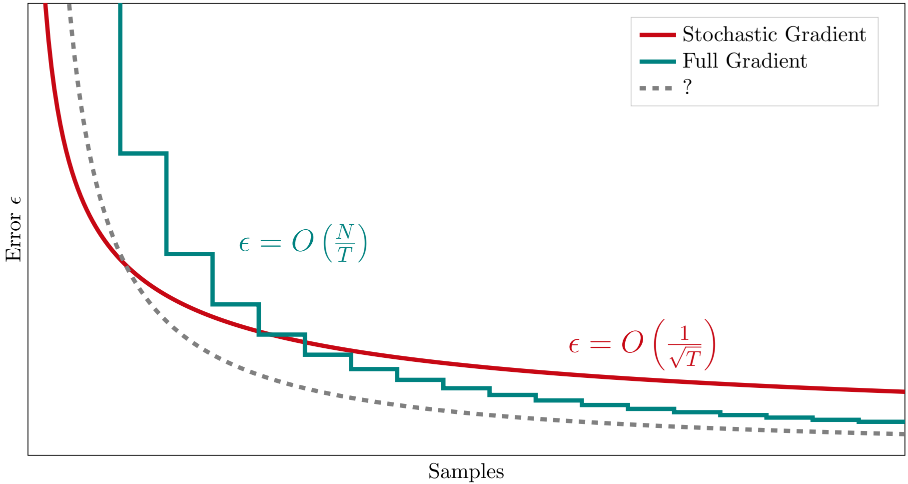
$$\omega(\beta, \pi_\theta|\tau) := \frac{\mathbb{P}(\tau|\pi_\theta)}{\mathbb{P}(\tau|\beta)} = \frac{\rho(s_1)\prod_{t=1}^{T} p(s_{t+1}|s_t, a_t)\pi_\theta(s_t, a_t)}{\rho(s_1)\prod_{t=1}^{T} p(s_{t+1}|s_t, a_t)\beta(s_t, a_t)} = \prod_{t=1}^{T}\frac{\pi_\theta(s_t, a_t)}{\beta(s_t, a_t)}$$

Partial fixes: clipping, normalization, etc.

❗ Off-policy RL is still a relevant open problem

Sample efficiency through variance-reduced gradient

Variance-reduced gradient estimator



Can we do something better?

Visualization idea from Bach [2016]

SVRG [Johnson and Zhang, 2013]

Stochastic Variance-Reduced Gradient

A solution from *finite-sum optimization*:

$$\max_{\theta} J(\theta) = \sum_{i=1}^{N} f_i(\theta)$$

epoch

iteration

$$\underbrace{\blacktriangledown J(\theta)}_{\text{SVRG estimator}} = \underbrace{\nabla J(\widetilde{\theta})}_{\text{FG } (\textit{snapshot})} + \underbrace{\nabla f_i(\theta)}_{\text{SG in current parameter}} - \underbrace{\nabla f_i(\widetilde{\theta})}_{\text{Correction term}}$$

- Unbiased
- Linear convergence
- More data-efficient than FG
- Supervised Learning (SL)

Algorithm 1 SVRG

Input: a dataset $\mathcal{D}_N$, number of epochs S, epoch size m, step size α, initial parameter $\boldsymbol{\theta}_m^0 := \widetilde{\boldsymbol{\theta}}^0$

for $s = 0$ **to** $S-1$ **do**

  $\boldsymbol{\theta}_0^{s+1} := \widetilde{\boldsymbol{\theta}}^s = \boldsymbol{\theta}_m^s$

  $\widetilde{\mu} = \nabla f(\widetilde{\boldsymbol{\theta}}^s)$

  for $t = 0$ **to** $m-1$ **do**

    $x \sim \mathcal{U}\left(\mathcal{D}_N\right)$

    $v_t^{s+1} = \widetilde{\mu} + \nabla z(x|\boldsymbol{\theta}_t^{s+1}) - \nabla z(x|\widetilde{\boldsymbol{\theta}}^s)$

    $\boldsymbol{\theta}_{t+1}^{s+1} = \boldsymbol{\theta}_t^{s+1} + \alpha v_t^{s+1}$

  end for

end for

Concave case: **return** $\boldsymbol{\theta}_m^S$

Non-Concave case: **return** $\boldsymbol{\theta}_t^{s+1}$ with (s,t) picked uniformly at random from $\{[0, S-1] \times [0, m-1]\}$

SVRG for RL: SVRPG

[Papini et al., 2018]

Issues in RL:

- non-concavity
- infinite dataset
- non-stationarity: $\tau \sim \pi_\theta$

Solution:

epoch

iteration

$$\underbrace{\blacktriangledown J(\theta)}_{\text{SVRPG estimator}} = \underbrace{\widehat{\nabla}_N J(\widetilde{\theta})}_{\substack{\text{Large N} \\ \text{to approximate FG}}} + \underbrace{\widehat{\nabla}_B J(\theta)}_{B \ll N} - \underbrace{\omega(\theta, \widetilde{\theta}) \widehat{\nabla}_B J(\widetilde{\theta})}_{\substack{\text{Importance weighting} \\ \text{for non-stationarity}}}$$

For $s = 1, \ldots$

Sample N trajectories using $\widetilde{\theta}$

Compute FG $= \widehat{\nabla}_N J(\widetilde{\theta})$

For $t = 1, \ldots, m$

Sample B trajectories using θ

Compute SG $= \widehat{\nabla}_B J(\theta)$

Compute correction $= \omega(\theta, \widetilde{\theta}) \widehat{\nabla}_B J(\widetilde{\theta})$

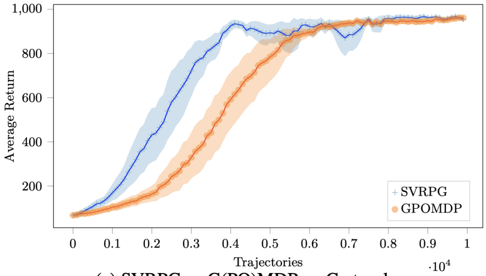
Update $\theta \leftarrow \theta + \alpha \blacktriangledown J(\theta)$

iteration

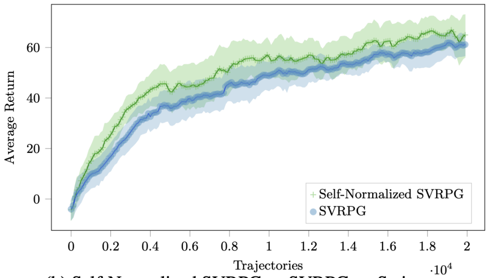
Update $\widetilde{\theta} \leftarrow \theta$

epoch

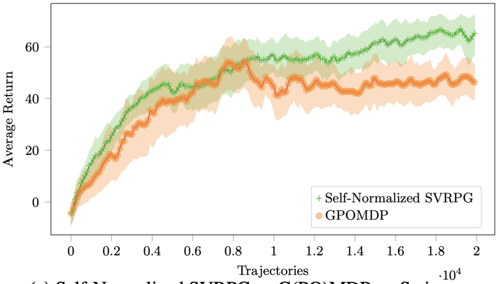
! *Importance sampling may reintroduce variance (use all the tricks)*



(a) SVRPG vs G(PO)MDP on Cart-pole.



(b) Self-Normalized SVRPG vs SVRPG on Swimmer.



(c) Self-Normalized SVRPG vs G(PO)MDP on Swimmer.

Conservative Approaches

Relative Performance

Issues:

- We would like to exploit past samples
- We do not know how much to trust them
- Depends on the distribution over trajectories induced by different policies

Relative Performance

Issues:

- We would like to exploit past samples
- We do not know how much to trust them
- Depends on the distribution over trajectories induced by different policies

Performance-Difference Lemma

[Burnetas and Katehakis, 1997, Prop. 1], [Kakade and Langford, 2002, Lem. 6.1], [Cao, 2007]

For any policies $\pi, \pi' \in \Pi^{\mathsf{SR}}$

$$J(\pi') - J(\pi) = \sum_{s,a} d^{\pi'}(s,a) A^{\pi}(s,a)$$
$$= \sum_{s} d^{\pi'}(s) \sum_{a} \pi'(s,a) A^{\pi}(s,a)$$

Proof

$$
\begin{aligned}
\mathbb{E}_{(s,a)\sim d^{\pi'}}[A^\pi(s,a)] &= \mathbb{E}_{(s,a)\sim d^{\pi'}}[q^\pi(s,a) - v^\pi(s)] \\
&= \mathbb{E}_{(s,a)\sim d^{\pi'}}[r(s,a)] + \mathbb{E}_{(s,a)\sim d^{\pi'}}\left[\gamma \sum_y p(y|s,a) v^\pi(y) - v^\pi(s)\right] \\
&= J(\pi') + \ \mathbb{E}_{(s,a)\sim d^{\pi'}}\left[\gamma \sum_y p(y|s,a) v^\pi(y)\right] \ - \mathbb{E}_{s\sim d^{\pi'}}[v^\pi(s)]
\end{aligned}
$$

Proof

$$\begin{aligned}
\mathbb{E}_{(s,a)\sim d^{\pi'}}[A^\pi(s,a)] &= \mathbb{E}_{(s,a)\sim d^{\pi'}}[q^\pi(s,a) - v^\pi(s)] \\
&= \mathbb{E}_{(s,a)\sim d^{\pi'}}[r(s,a)] + \mathbb{E}_{(s,a)\sim d^{\pi'}}\left[\gamma \sum_y p(y|s,a) v^\pi(y) - v^\pi(s)\right] \\
&= J(\pi') + \mathbb{E}_{(s,a)\sim d^{\pi'}}\left[\gamma \sum_y p(y|s,a) v^\pi(y)\right] - \mathbb{E}_{s\sim d^{\pi'}}[v^\pi(s)]
\end{aligned}$$

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

Proof

$$\begin{aligned}
\mathbb{E}_{(s,a)\sim d^{\pi'}}[A^\pi(s,a)] &= \mathbb{E}_{(s,a)\sim d^{\pi'}}[q^\pi(s,a) - v^\pi(s)] \\
&= \mathbb{E}_{(s,a)\sim d^{\pi'}}[r(s,a)] + \mathbb{E}_{(s,a)\sim d^{\pi'}}\left[\gamma \sum_y p(y|s,a) v^\pi(y) - v^\pi(s)\right] \\
&= J(\pi') + \mathbb{E}_{(s,a)\sim d^{\pi'}}\left[\gamma \sum_y p(y|s,a) v^\pi(y)\right] - \mathbb{E}_{s\sim d^{\pi'}}[v^\pi(s)]
\end{aligned}$$

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

$$= J(\pi') + \sum_y d^{\pi'}(y) v^\pi(y) - \sum_y \rho(y) v^\pi(y) - \mathbb{E}_{s\sim d^{\pi'}}[v^\pi(s)]$$

Optimization step

$$\max_{\pi'} J(\pi')$$

Optimization step

$$\max_{\pi'} J(\pi') = \max_{\pi'} J(\pi') - J(\pi)$$

Issue: as before, cannot be directly estimated using information from π

Optimization step

$$
\begin{aligned}
\max_{\pi'} J(\pi') &= \max_{\pi'} J(\pi') - J(\pi) \\
&= \max_{\pi'} \mathbb{E}_{(s,a)\sim d^{\pi'}} \left[A^{\pi}(s,a)\right]
\end{aligned}
$$

Issue: as before, cannot be directly estimated using information from π

Optimization step

$$J(\pi') - J(\pi) = \mathbb{E}_{s \sim d^\pi}\left[\sum_a \pi'(s,a) A^\pi(s,a)\right] + \sum_s (d^{\pi'}(s) - d^\pi(s)) \sum_a \pi'(s,a) A^\pi(s,a)$$

Optimization step

$$J(\pi') - J(\pi) = \mathbb{E}_{s\sim d^\pi}\left[\sum_a \pi'(s,a)A^\pi(s,a)\right] + \sum_s \underbrace{(d^{\pi'}(s) - d^\pi(s))}_{?} \sum_a \pi'(s,a)A^\pi(s,a)$$

$$\geq \mathbb{E}_{s\sim d^\pi}\left[\sum_a \pi'(s,a)A^\pi(s,a) - \frac{\gamma\varepsilon}{(1-\gamma)^2} D_{TV}(\pi'\|\pi)[s]\right]$$

where $\varepsilon = \max_s \left|\mathbb{E}_{a\sim\pi'}[A^\pi(s,a)]\right|$ and

$$D_{TV}(\pi'\|\pi)[s] = \sum_a |\pi'(s,a) - \pi(s,a)|$$

Surrogate Loss

$$L_\pi(\pi') = J(\pi) + \sum_s d^{\pi}(s) \sum_a \pi'(s,a) A^\pi(s,a)$$

- $L_\pi(\pi) = J(\pi)$
- If parametric policies $\pi = \pi_\theta$, $\nabla_\theta L_{\pi_\theta}(\pi_\theta) = \nabla_\theta J(\pi_\theta)$

! in an interval close to π, L_π is a good surrogate for J

$\implies$ *Conservative Policy Iteration* [Kakade and Langford, 2002]

(fig)

Surrogate Loss

$$L_\pi(\pi') = J(\pi) + \sum_s d^\pi(s) \sum_a \pi'(s,a) A^\pi(s,a) \quad - \sum_s d^\pi(s) \frac{\gamma \varepsilon}{(1-\gamma)^2} D_{TV}(\pi' \| \pi)[s]$$

also with this

- $L_\pi(\pi) = J(\pi)$
- If parametric policies $\pi = \pi_\theta$, $\nabla_\theta L_{\pi_\theta}(\pi_\theta) = \nabla_\theta J(\pi_\theta)$

! in an interval close to π, L_π is a good surrogate for J

$\Longrightarrow$ *Conservative Policy Iteration* [Kakade and Langford, 2002]

(fig)

Conservative Policy Iteration

- *New policy improvement schema*
  - Give current policy π_k solve

$$\max_{\pi'} \left\{ L_{\pi_k}(\pi') - \boldsymbol{C}\ \mathbb{E}_{s\sim d^{\pi}} \left[D_{TV}(\pi' \| \pi_k)[s] \right] \right\}$$

Conservative Policy Iteration

- *New policy improvement schema*
    - Give current policy π_k solve

$$\max_{\pi'} \left\{ L_{\pi_k}(\pi') - \boldsymbol{C}\ \mathbb{E}_{s\sim d^{\pi}} \left[D_{TV}(\pi' \| \pi_k)[s] \right] \right\} \geq 0$$

Conservative Policy Iteration

- *New policy improvement schema*
  - Give current policy π_k solve

$$\boldsymbol{J(\pi') - J(\pi_k)} \geq \max_{\pi'} \left\{ L_{\pi_k}(\pi') - \boldsymbol{C}\ \mathbb{E}_{s\sim d^{\pi}} \left[D_{TV}(\pi' \| \pi_k)[s] \right] \right\} \geq \mathbf{0}$$

Conservative Policy Iteration

- *New policy improvement schema*
  - Give current policy π_k solve

$$\boldsymbol{J(\pi') - J(\pi_k)} \geq \max_{\pi'} \left\{ L_{\pi_k}(\pi') - \boldsymbol{C}\ \mathbb{E}_{s\sim d^{\pi}} \left[D_{TV}(\pi' \| \pi_k)[s] \right] \right\} \geq \mathbf{0}$$

$\implies$ *Monotonic performance improvement*

Conservative Policy Iteration

- *New policy improvement schema*
  - Give current policy π_k solve

$$\boldsymbol{J(\pi') - J(\pi_k)} \geq \max_{\pi'} \left\{ L_{\pi_k}(\pi') - \boldsymbol{C} \; \mathbb{E}_{s\sim d^{\pi}} \left[D_{TV}(\pi' \| \pi_k)[s] \right] \right\} \geq \boldsymbol{0}$$

$\implies$ *Monotonic performance improvement*

Several approaches have been proposed [e.g., Kakade and Langford, 2002, Perkins and Precup, 2002, Gabillon et al., 2011, Wagner, 2011, 2013, Pirotta et al., 2013b, Scherrer et al., 2015, Schulman et al., 2015]

Approximate Monotone Improvement

- The objective can be estimated using rollouts from the most recent policy
- Updates respect a notion of distance in the policy space!

This is the basis for many algorithms!

How to solve the optimization problem?

$$\max_{\pi'} \left\{ L_{\pi_k}(\pi') - C \; \mathbb{E}_{s \sim d^{\pi}} \left[D_{TV}(\pi' \| \pi_k)[s] \right] \right\}$$

How to solve the optimization problem?

$$\max_{\pi'} \left\{ L_{\pi_k}(\pi') - C \; \mathbb{E}_{s\sim d^{\pi}} \left[D_{TV}(\pi' \| \pi_k)[s] \right] \right\}$$

In *discrete MDP* with *convex policy update*

$$\pi_{k+1} = \alpha \overline{\pi} + (1-\alpha)\pi_k$$

where $\overline{\pi}$ is the greedy policy

$\Longrightarrow$ closed form solution for α
$\Longrightarrow$ guaranteed improvement

Conservative in Continuous MDPs

- Consider parametrized policies $\theta \mapsto \pi_\theta$
- Construct a *lower bound* to $J(\theta + \Delta\theta) - J(\theta)$
  - e.g., [Pirotta et al., 2013, Papini et al., 2017]

Conservative in Continuous MDPs

- Consider parametrized policies $\theta \mapsto \pi_\theta$
- Construct a *lower bound* to $J(\theta + \Delta\theta) - J(\theta)$
  - e.g., [Pirotta et al., 2013, Papini et al., 2017]

If Π_θ is a smoothing policy class [Papini et al., 2019]

(as a consequence of quadratic bound for L-smooth functions)

$$\forall \theta, \theta' \qquad J(\theta') - J(\theta) \geq (\theta' - \theta)^\mathsf{T} \nabla_\theta J(\theta) - \frac{L}{2} \|\theta' - \theta\|_2^2$$

Conservative in Continuous MDPs

- Consider parametrized policies $\theta \mapsto \pi_\theta$
- Construct a *lower bound* to $J(\theta + \Delta\theta) - J(\theta)$
  - e.g., [Pirotta et al., 2013, Papini et al., 2017]

If Π_θ is a smoothing policy class [Papini et al., 2019]
(as a consequence of quadratic bound for L-smooth functions)

$$\forall \theta, \theta' \qquad J(\theta') - J(\theta) \geq (\theta' - \theta)^\mathsf{T} \nabla_\theta J(\theta) - \frac{L}{2} \|\theta' - \theta\|_2^2$$
$$= \alpha \|\nabla_\theta J(\theta)\|_2^2 - \alpha^2 \frac{L}{2} \|\nabla_\theta J(\theta)\|_2^2$$

by using gradient update rule $\theta' = \theta + \alpha \nabla_\theta J(\theta)$

Conservative in Continuous MDPs

- Consider parametrized policies $\theta \mapsto \pi_\theta$
- Construct a *lower bound* to $J(\theta + \Delta\theta) - J(\theta)$
  - e.g., [Pirotta et al., 2013, Papini et al., 2017]

If Π_θ is a smoothing policy class [Papini et al., 2019]
(as a consequence of quadratic bound for L-smooth functions)

$$\forall \theta, \theta' \qquad J(\theta') - J(\theta) \geq (\theta' - \theta)^\mathsf{T} \nabla_\theta J(\theta) - \frac{L}{2} \|\theta' - \theta\|_2^2$$
$$= \alpha \|\nabla_\theta J(\theta)\|_2^2 - \alpha^2 \frac{L}{2} \|\nabla_\theta J(\theta)\|_2^2$$

by using gradient update rule $\theta' = \theta + \alpha \nabla_\theta J(\theta)$

$\implies \alpha^\star = \frac{1}{L} \qquad \implies$ Monotonic policy performance improvement

Conservative Approaches: Approximation

- Can be extended to handle *approximate estimate*

$$\|A(s,a) - \widehat{A}(s,a)\| \leq \epsilon \quad \text{and/or} \quad \|\nabla J(\theta) - \widehat{\nabla} J(\theta)\| \leq \epsilon$$

- Need to change the stopping condition to *account for the finite-sample error*

Conservative Approaches: Approximation

- Can be extended to handle *approximate estimate*

$$\|A(s,a) - \widehat{A}(s,a)\| \leq \epsilon \quad \text{and/or} \quad \|\nabla J(\theta) - \widehat{\nabla} J(\theta)\| \leq \epsilon$$

- Need to change the stopping condition to *account for the finite-sample error*

Example: $\widehat{\nabla}_N J(\theta)$ estimate of the gradient using N trajectories. Then *whp*

$$\|\nabla J(\theta) - \widehat{\nabla}_N J(\theta)\| \leq \frac{\epsilon_\delta}{\sqrt{N}}$$

As a consequence, *whp*

$$J(\theta') - J(\theta) \geq \alpha \left(\|\nabla_\theta J(\theta)\|_2^2 - \frac{\epsilon_\delta^2}{N} \right) - \alpha^2 \frac{L}{2} \|\nabla_\theta J(\theta)\|_2^2$$

Conservative Approaches: Approximation

- Can be extended to handle *approximate estimate*

$$\|A(s,a) - \widehat{A}(s,a)\| \leq \epsilon \quad \text{and/or} \quad \|\nabla J(\theta) - \widehat{\nabla} J(\theta)\| \leq \epsilon$$

- Need to change the stopping condition to *account for the finite-sample error*

Example: $\widehat{\nabla}_N J(\theta)$ estimate of the gradient using N trajectories. Then *whp*

$$\|\nabla J(\theta) - \widehat{\nabla}_N J(\theta)\| \leq \frac{\epsilon_\delta}{\sqrt{N}}$$

As a consequence, *whp*

$$J(\theta') - J(\theta) \geq \alpha \left(\|\nabla_\theta J(\theta)\|_2^2 - \frac{\epsilon_\delta^2}{N} \right) - \alpha^2 \frac{L}{2} \|\nabla_\theta J(\theta)\|_2^2$$

$+$ possibility to adapt also N

Toward Practical Algorithm

- Optimizing the total variation $D_{TV}(\pi' \| \pi)$ may be *difficult*
- Relax the problem using *Pinsker's inequality* [Csiszar and Körner, 2011]

$$D_{TV}(\pi' \| \pi) \leq \sqrt{2 D_{KL}(\pi' \| \pi)}$$

* implicitly done in the analysis of conservative gradient

Kullback–Leibler divergence

Given two probability distributions P and Q

$$D_{KL}(P\|Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$$

Properties:

- $D_{KL}(P\|Q) \geq 0$
- $D_{KL}(Q\|Q) = 0$
- $D_{KL}(P\|Q) \neq D_{KL}(Q\|P)$ (non-symmetric)
- No triangle inequality

Note: Réni divergences provide generalizations of the KL divergence

Further Steps toward Practical Algorithms

- C provided by theory is quite high (*too conservartive*)
- Replace regularization with constraint (*trust region*) (e.g., REPS [Peters et al., 2010])

$$\pi_{k+1} = \arg\max_{\pi'} L_\pi(\pi')$$
$$\text{s.t. } \mathbb{E}_{s\sim d^\pi}[D_{KL}(\pi' \| \pi)] \leq \delta$$

Further Steps toward Practical Algorithms

- C provided by theory is quite high (*too conservartive*)
- Replace regularization with constraint (*trust region*) (e.g., REPS [Peters et al., 2010])

$$\pi_{k+1} = \arg\max_{\pi'} L_\pi(\pi')$$
$$\text{s.t. } \mathbb{E}_{s\sim d^\pi}[D_{KL}(\pi'\|\pi)] \leq \delta$$

- Importance weighting

$$\mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim\pi'}[A^\pi(s,a)] = \mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim z}\left[\frac{\pi'(s,a)}{z(s,a)}A^\pi(s,a)\right]$$

Further Steps toward Practical Algorithms

- C provided by theory is quite high (*too conservartive*)
- Replace regularization with constraint (*trust region*) (e.g., REPS [Peters et al., 2010])

$$\pi_{k+1} = \arg\max_{\pi'} L_\pi(\pi')$$
$$\text{s.t. } \mathbb{E}_{s\sim d^\pi}[D_{KL}(\pi'\|\pi)] \leq \delta$$

- Importance weighting

$$\mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim\pi'}[A^\pi(s,a)] = \mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim z}\left[\frac{\pi'(s,a)}{z(s,a)}A^\pi(s,a)\right]$$

- Replace A^π with q^π and remove $J(\pi)$

$$\pi_{k+1} = \arg\max_{\pi'} \mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim z}\left[\frac{\pi'(s,a)}{z(s,a)}q^\pi(s,a)\right]$$
$$\text{s.t. } \mathbb{E}_{s\sim d^\pi}[D_{KL}(\pi'\|\pi)] \leq \delta$$

$\Longrightarrow$ Trust-Region Policy Optimization (TRPO) [Schulman et al., 2015]

Beyond Simple Gradient Descent

Gradient Descent

Steepest descent direction of a function $h(\theta) \to -\nabla h(\theta)$

- It yields the *most reduction* in h per unit of change in θ
- Change is measured using the standard *Euclidean norm* $\|\cdot\|$

$$\frac{-\nabla h}{\|\nabla h\|} = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \arg\min_{d:\|d\| \leq \epsilon} \{h(\theta + d)\}$$

Gradient Descent

Steepest descent direction of a function $h(\theta) \to -\nabla h(\theta)$

- It yields the *most reduction* in h per unit of change in θ
- Change is measured using the standard *Euclidean norm* $\|\cdot\|$

$$\frac{-\nabla h}{\|\nabla h\|} = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \arg \min_{d:\|d\| \leq \epsilon} \{h(\theta + d)\}$$

Is the Euclidean norm the best metric?
Can we use an alternative definition of (*local*) distance?

Gradient Descent

Steepest descent direction of a function $h(\theta) \to -\nabla h(\theta)$

- It yields the *most reduction* in h per unit of change in θ
- Change is measured using the standard *Euclidean norm* $\|\cdot\|$

$$\frac{-\nabla h}{\|\nabla h\|} = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \arg\min_{d:\|d\| \leq \epsilon} \{h(\theta + d)\}$$

Is the Euclidean norm the best metric?
Can we use an alternative definition of (*local*) distance?

$\Longrightarrow$ as suggested by [Amari, 1998] it is better to *define a metric* based not on the choice of the coordinates but rather *on the manifold these coordinates parametrize*!

(Example: gradient descent is not affine invariant)

Natural Gradient

- In Riemannian space, the distance is defined as

$$d^2(v, v + \delta v) = \delta v^{\mathsf{T}} G(v) \delta v^{\mathsf{T}}$$

where G is the *metric tensor*

Natural Gradient

- In Riemannian space, the distance is defined as

$$d^2(v, v+\delta v) = \delta v^{\mathsf{T}} G(v) \delta v^{\mathsf{T}}$$

  where G is the *metric tensor*

Example: consider the Euclidean space ($\mathbb{R}^2$)

- Cartesian coordinate, the metric tensor is the identity
- Polar coordinate

$$\begin{aligned}
x = r\cos\theta &\implies \delta x = \delta r \cos\theta - r\delta\theta \sin\theta \\
y = r\sin\theta &\implies \delta y = \delta r \sin\theta + r\delta\theta \cos\theta \\
d^2(v, v+\delta v) &= \delta x^2 + \delta y^2 \\
&= \delta r^2 + r^2 \delta\theta^2 \\
&= (\delta r, \delta\theta)^{\mathsf{T}} diag(1, r^2)(\delta r, \delta\theta)
\end{aligned}$$

Natural Gradient

Natural Gradient [Amari, 1998]

The steepest descent in a Riemannian is given by

$$\widetilde{\nabla} h(\theta) = G(\theta)^{-1} \nabla h(\theta)$$

Natural Gradient

Natural Gradient [Amari, 1998]

The steepest descent in a Riemannian is given by

$$\widetilde{\nabla} h(\theta) = G(\theta)^{-1} \nabla h(\theta)$$

Natural gradient can be applied to any objective function

Issue: what is the metric tensor?

Natural Gradient

Natural Gradient [Amari, 1998]

The steepest descent in a Riemannian is given by

$$\widetilde{\nabla} h(\theta) = G(\theta)^{-1} \nabla h(\theta)$$

Natural gradient can be applied to any objective function

Issue: what is the metric tensor?

known for many objectives!

Natural Gradient

Natural Gradient [Amari, 1998]

The steepest descent in a Riemannian is given by

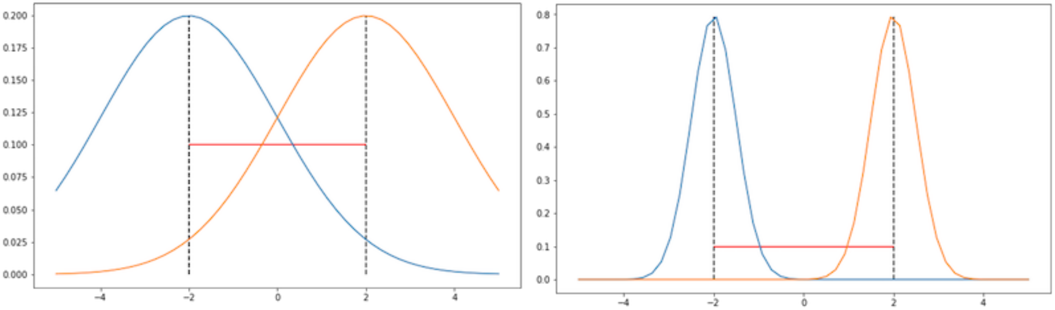
$$\widetilde{\nabla} h(\theta) = G(\theta)^{-1} \nabla h(\theta)$$

Natural gradient can be applied to any objective function
Issue: what is the metric tensor?
known for many objectives!

Maximum Likelihood: we have a probabilistic model represented by its likelihood $p(x|\theta)$
We want to maximize this likelihood function to find the most likely parameter

Example

Consider a Gaussian parameterized by only its mean and keep the variance fixed to 2 and 0.5 for the first and second image respectively



The distance of those Gaussians are the same, i.e. 4, according to Euclidean metric (red line)

`https://wiseodd.github.io/techblog/2018/03/14/natural-gradient/`

Fisher Information Matrix

$$F = \mathop{\mathbb{E}}_{x \sim p(\cdot|\theta)} \left[\nabla \log p(x|\theta) \, \nabla \log p(x|\theta)^{\mathsf{T}} \right]$$

Property 1: Fisher Information Matrix *is the Hessian of KL-divergence* between two distributions $p(x|\theta)$ and $p(x|\theta')$, with respect to θ', evaluated at $\theta = \theta'$

$$H_{D_{KL}}(p(x|\theta) \| p(x|\theta')) = F$$

Fisher Information Matrix

$$F = \mathop{\mathbb{E}}_{x \sim p(\cdot|\theta)} \left[\nabla \log p(x|\theta) \, \nabla \log p(x|\theta)^{\mathsf{T}} \right]$$

Property 1: Fisher Information Matrix *is the Hessian of KL-divergence* between two distributions $p(x|\theta)$ and $p(x|\theta')$, with respect to θ', evaluated at $\theta = \theta'$

$$H_{D_{KL}}(p(x|\theta) \| p(x|\theta')) = F$$

Property 2: Second-order Taylor series expansion

$$D_{KL}(p(x|\theta) \| p(x|\theta + d)) = d^{\mathsf{T}} F d + O(d^3)$$

(proofs)

Natural Gradient in ML

[Martens, 2014]

For a positive definite matrix A, we have [Ollivier et al., 2017] (def. $\|x\|_B = \sqrt{x^\top B x}$)

$$\frac{-A^{-1}\nabla h}{\|\nabla h\|_{A^{-1}}} = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \operatorname*{arg\,min}_{d : \|d\|_{A^{-1}} \leq \epsilon} \{h(\theta + d)\}$$

Natural Gradient in ML

[Martens, 2014]

For a positive definite matrix A, we have [Ollivier et al., 2017] (def. $\|x\|_B = \sqrt{x^\mathsf{T} B x}$)

$$\frac{-A^{-1}\nabla h}{\|\nabla h\|_{A^{-1}}} = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \operatorname*{arg\,min}_{d : \|d\|_{A^{-1}} \leq \epsilon} \{h(\theta + d)\}$$

$$A = \frac{1}{2} F \quad \Longrightarrow \quad -\sqrt{2} \frac{\widetilde{\nabla} h}{\|\nabla h\|_{F^{-1}}} = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \operatorname*{arg\,min}_{d : D_{KL}(p(x|\theta) \| p(x|\theta + d)) \leq \epsilon^2} \{h(\theta + d)\}$$

Negative natural gradient

- steepest descent direction *in the space of distributions*
- where distance is (*approximately*) measured in local neighborhoods by the KL divergence

Natural Gradient in ML

[Martens, 2014]

For a positive definite matrix A, we have [Ollivier et al., 2017] (def. $\|x\|_B = \sqrt{x^\top B x}$)

$$\frac{-A^{-1}\nabla h}{\|\nabla h\|_{A^{-1}}} = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \underset{d: \|d\|_{A^{-1}} \leq \epsilon}{\arg\min} \{h(\theta + d)\}$$

$$A = \frac{1}{2}F \quad \Longrightarrow \quad -\sqrt{2}\frac{\widetilde{\nabla} h}{\|\nabla h\|_{F^{-1}}} = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \underset{d: D_{KL}(p(x|\theta)\|p(x|\theta+d)) \leq \epsilon^2}{\arg\min} \{h(\theta + d)\}$$

Negative natural gradient

- steepest descent direction *in the space of distributions*
- where distance is (*approximately*) measured in local neighborhoods by the KL divergence

! $D_{KL}(p(x|\theta)\|p(x|\theta + d))$ is locally/asymptotically *symmetric* as $d \to 0$,
and so will be (approximately) symmetric in a local neighborhood [Martens, 2014]

! $\widetilde{\nabla} h$ is be *invariant* to the choice of parameterization

Natural Policy Gradient

Trust-region objective

$$\pi_{k+1} = \arg\max_{\pi'} L_{\pi_k}(\pi')$$
$$\text{s.t. } \overline{D}_{KL}(\pi' \| \pi_k) \leq \delta$$

Approximate objective and KL

$$L_{\theta_k}(\theta) \approx L_{\theta_k}(\theta_k) + g^\mathsf{T}(\theta - \theta_k)$$
$$\overline{D}_{KL}(\theta \| \theta_k) \approx \frac{1}{2}(\theta - \theta_k)^\mathsf{T} F(\theta - \theta_k)$$

$\Longrightarrow$

$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{g^\mathsf{T} F^{-1} g}} \underbrace{F^{-1} g}_{:=\widetilde{\nabla} J}$$

Algorithms [Kakade, 2002, Peters and Schaal, 2008a]

Truncated Natural Policy Gradient

Issues:

- $\theta \in \mathbb{R}^d$, d can be very large (e.g., thousands or millions)
- H or F have dimension d^2
- matrix inversion is $\mathcal{O}(d^3)$

Truncated Natural Policy Gradient

Issues:

- $\theta \in \mathbb{R}^d$, d can be very large (e.g., thousands or millions)
- H or F have dimension d^2
- matrix inversion is $\mathcal{O}(d^3)$

Solution:

- Use conjugate gradient to compute $F^{-1}g$ without inverting F [Pascanu and Bengio, 2013]
- With j iterations, CG solves systems of equations $Hx = g$ for x by finding projection onto Krylov subspace (i.e., $span(g, Hg, \dots H^{j-1}g)$)

$\implies$ *Truncated Natural Policy Gradient*

Truncated Natural Policy Gradient

Issues:

- $\theta \in \mathbb{R}^d$, d can be very large (e.g., thousands or millions)
- H or F have dimension d^2
- matrix inversion is $\mathcal{O}(d^3)$

Solution:

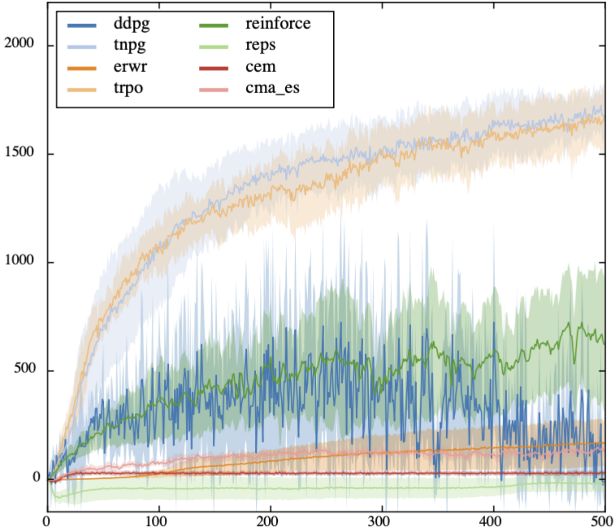
- Use conjugate gradient to compute $F^{-1}g$ without inverting F [Pascanu and Bengio, 2013]
- With j iterations, CG solves systems of equations $Hx = g$ for x by finding projection onto Krylov subspace (i.e., $span(g, Hg, \dots H^{j-1}g)$)

$\Longrightarrow$ *Truncated Natural Policy Gradient*

Other solutions are possible: see ACKTR [Wu et al., 2017], [Ollivier, 2017]

Example: Walker-2d

[Duan et al., 2016]



Discussion

- Natural gradient contains second order informations
- Newton method?

Discussion

- Natural gradient contains second order informations
- Newton method?

The Hessian [Furmston and Barber, 2012, Shen et al., 2019]

$$\nabla^2 J(\theta) = \mathbb{E}_\tau \left[\nabla g(\theta, \tau) \nabla \log \mathbb{P}(\tau|\theta)^\mathsf{T} + \nabla^2 g(\theta, \tau) \right]$$

with

$$g(\theta, \tau) = \sum_{h=1}^{H} \sum_{i=h}^{H} \gamma^i r(s_i, a_i) \log \pi_\theta(s_h, a_h)$$

Discussion

- [Furmston and Barber, 2012] noticed a connection between $\mathbb{E}[\nabla^2 g(\theta, \tau)]$ and the FIM!
- This hessian can be estimated using first-order information (leading to *quasi Newton approaches*) or *finite difference*
  - see [Shen et al., 2019] also for sample complexity

REINFORCE find an ϵ-approximate first-order stationary point in $O(1/\epsilon^4)$
Hessian aided policy gradient method [Shen et al., 2019] sample complexity of $O(1/\epsilon^3)$

Proximal Policy Optimization

[Schulman et al., 2017b]

- Avoid to compute the natural gradient
- Approximate the KL constraint

Proximal Policy Optimization

[Schulman et al., 2017b]

- Avoid to compute the natural gradient
- Approximate the KL constraint

1. *Adaptive KL Penalty*
    - Consider regularized optimization problem

$$\theta_{k+1} = \arg\max_{\theta} L_{\theta_k}(\theta) - \lambda_k \mathbb{E}[D_{KL}(\theta \| \theta_k)]$$

    - Adapt λ_k to enforce KL constraint

$$\lambda_{k+1} = \begin{cases} 2\lambda_k & \text{if } \mathbb{E}[D_{KL}(\theta \| \theta_k)] \geq 1.5\delta \\ \lambda_k/2 & \text{if } \mathbb{E}[D_{KL}(\theta \| \theta_k)] \leq \delta/1.5 \\ \lambda_k & \text{otherwise} \end{cases}$$

Proximal Policy Optimization

[Schulman et al., 2017b]

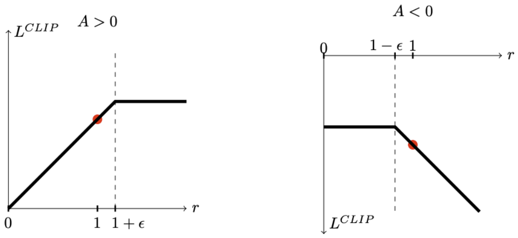
2 *Clipped Objective*

- Recall surrogate objective

$$L_\pi^{\text{IS}}(\pi') = \mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim\pi}\left[\frac{\pi'(s,a)}{\pi(s,a)}A^\pi(s,a)\right] = \mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim\pi}\left[r_{sa}(\pi')A^\pi(s,a)\right]$$

- Form a lower bound via clipped importance ratios

$$L_\pi^{\text{CLIP}}(\pi') = \mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim\pi}\left[\min\left\{r_{sa}(\pi')A^\pi(s,a), \text{clip}(r_{sa}(\pi'), 1-\epsilon, 1+\epsilon)A^\pi(s,a)\right\}\right]$$



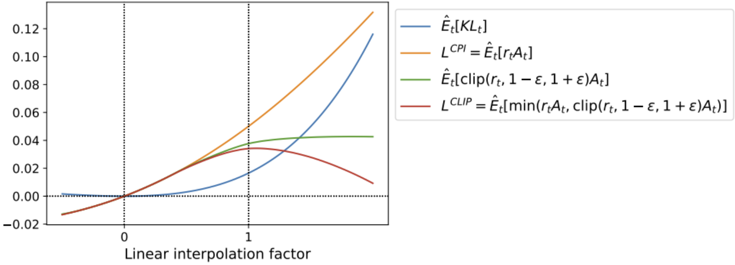
- $\pi_{k+1} = \arg\max_\pi L_{\pi_k}^{\text{CLIP}}(\pi)$

Proximal Policy Optimization

[Schulman et al., 2017b]

- Clipping prevents policy from moving too much away from θ_k
- Seems to work as well as PPO with KL penalty
- Much simpler to implement

How does it work?



Various objectives as a function of function of α between θ_k and θ_{k+1}

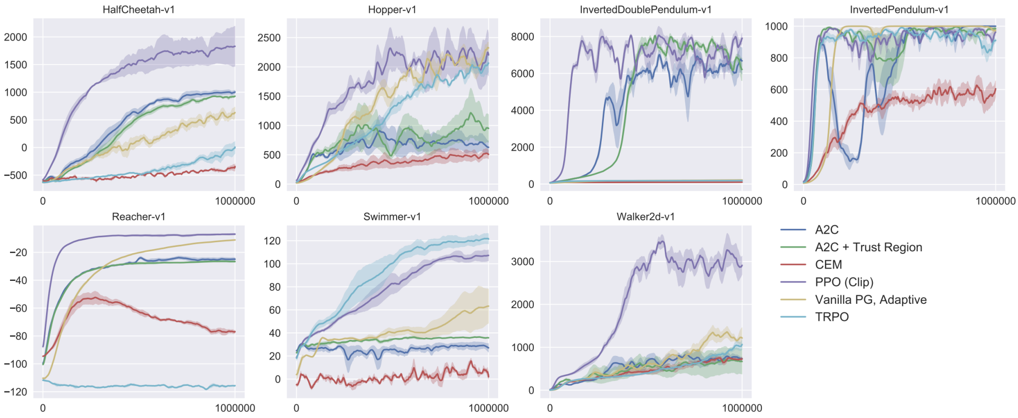


Figure 3: Comparison of several algorithms on several MuJoCo environments, training for one million timesteps.

Non-Parametric Policy Update

- Solve a constrained optimization problem in a non-parameterized policy space
- *Fit a parametric policy* on the best non-parametric policy

$\Longrightarrow$ Supervised Policy Update [Vuong et al., 2019]

Non-Parametric Policy Update

- Solve a constrained optimization problem in a non-parameterized policy space
- *Fit a parametric policy* on the best non-parametric policy

$\Longrightarrow$ Supervised Policy Update [Vuong et al., 2019]

1. Sample N trajectories using policy π_{θ_k}
   - construct dataset (s_i, a_i, A_i) where $A_i \approx A^{\pi_k}(s_i, a_i)$
2. For each s_i solve the constrained optimization problem
   - obtain a non-parametric policy $\widetilde{\pi}$ defined in each sample s_i
3. Fit a parametric policy $\pi_{\theta_{k+1}}$ on π

$$\min_{\theta} \left\{ \mathcal{L}(\theta) = \frac{1}{m} \sum_{i=1}^{m} D_{KL}(\pi_\theta \| \widetilde{\pi})[s_i] \right\}$$

Non-Parametric Policy Update

Example: TRPO optimization problem
Almost closed form solution (up to parameters $\lambda = f(\delta, \epsilon)$)

$$\widetilde{\pi}(s,a) \propto \pi_{\theta_k}(s,a) \exp\left[\frac{A^{\pi_{\theta_k}}(s,a)}{\lambda}\right]$$

Non-Parametric Policy Update

Example: TRPO optimization problem

Almost closed form solution (up to parameters $\lambda = f(\delta, \epsilon)$)

$$\widetilde{\pi}(s,a) \propto \pi_{\theta_k}(s,a) \exp\left[\frac{A^{\pi_{\theta_k}}(s,a)}{\lambda}\right]$$

Then (*approximately*)

$$\mathcal{L}(\theta) \approx \frac{1}{m}\sum_{i=1}^{m}\left(\nabla_\theta \underbrace{D_{KL}(\pi_\theta \| \pi_{\theta_k})[s_i]}_{\textit{policy deviation}} - \underbrace{\frac{1}{\lambda}\frac{\nabla_\theta \pi_\theta(s_i,a_i)}{\pi_{\theta_k}(s_i,a_i)}A_i}_{\textit{approximate performance}}\right)\mathbb{1}\left(D_{KL}(\pi_\theta \| \pi_{\theta_k})[s_i] \leq \epsilon\right)$$

Non-Parametric Policy Update

Example: TRPO optimization problem
Almost closed form solution (up to parameters $\lambda = f(\delta, \epsilon)$)

$$\widetilde{\pi}(s,a) \propto \pi_{\theta_k}(s,a) \exp\left[\frac{A^{\pi_{\theta_k}}(s,a)}{\lambda}\right]$$

Then (*approximately*)

$$\mathcal{L}(\theta) \approx \frac{1}{m}\sum_{i=1}^{m}\left(\nabla_\theta \underbrace{D_{KL}(\pi_\theta \| \pi_{\theta_k})[s_i]}_{\text{policy deviation}} - \underbrace{\frac{1}{\lambda}\frac{\nabla_\theta \pi_\theta(s_i,a_i)}{\pi_{\theta_k}(s_i,a_i)} A_i}_{\text{approximate performance}}\right) \mathbb{1}\left(D_{KL}(\pi_\theta \| \pi_{\theta_k})[s_i] \leq \epsilon\right)$$

! *minimize by gradient descent and consider λ to be a parameter!*
still an actor-critic approach!

Non-Parametric Policy Update

Example: TRPO optimization problem

Almost closed form solution (up to parameters $\lambda = f(\delta, \epsilon)$)

$$\widetilde{\pi}(s,a) \propto \pi_{\theta_k}(s,a) \exp\left[\frac{A^{\pi_{\theta_k}}(s,a)}{\lambda}\right]$$

Then (*approximately*)

$$\mathcal{L}(\theta) \approx \frac{1}{m}\sum_{i=1}^{m}\left(\nabla_\theta \underbrace{D_{KL}(\pi_\theta \| \pi_{\theta_k})[s_i]}_{\textit{policy deviation}} - \underbrace{\frac{1}{\lambda}\frac{\nabla_\theta \pi_\theta(s_i,a_i)}{\pi_{\theta_k}(s_i,a_i)}A_i}_{\textit{approximate performance}}\right)\mathbb{1}\left(D_{KL}(\pi_\theta \| \pi_{\theta_k})[s_i] \le \epsilon\right)$$

! *minimize by gradient descent and consider λ to be a parameter!*

still an actor-critic approach!

Not really a novel idea $\Longrightarrow$ *Classification-based PI*

Classification-based Policy Iteration (RCPI)

- replaces the policy evaluation step with computing rollout estimates of q^π

$$\mathcal{D} = \{x_i\}_{i=1}^N \mapsto \widehat{q}^\pi$$

- casts the policy improvement step as a classification problem
  - find a policy in a given hypothesis space that best predicts the greedy action at every (observed) state

$$\min_{\pi \in \Pi} \frac{1}{N} \sum_{i=1}^N \Big(\max_a \widehat{q}^{\pi_k}(s_i, a) - \widehat{q}^{\pi_k}(s_i, \pi(s_i)) \Big)$$

Classification-based approaches: [Lagoudakis and Parr, 2003b, Fern et al., 2003, Dimitrakakis and Lagoudakis, 2008, Lazaric et al., 2012, Gabillon et al., 2011]

Classification-based Policy Iteration with Critic

[Gabillon et al., 2011]

Estimate the return of a state-action pair as

$$R_j^{\pi_k}(s_i, a) = \underbrace{R_j^{\pi_k, H}(s_i, a)}_{H-\text{horizon rollout}} + \underbrace{\gamma^H \widehat{v}^{\pi_k}(s_{ij}^H)}_{\text{bostrapping}}$$

with

$$R_j^{\pi_k, H}(s_i, a) = r(s_i, a) + \sum_{t=1}^{H-1} \gamma^t r(x_{ij}^t, \pi_k(x_{ij}^t))$$

Then

$$\widehat{q}^{\pi_k}(s_i, a) = \frac{1}{m} \sum_{j=1}^{m} R_j^{\pi_k}(s_i, a)$$

Discussion

Key components:

1. Stochastic policies
2. Regularized or constrained optimization

What are the motivations

- Exploration
- Controlling the deviation
- Differentiability of Bellman operator

So far regularization was coming from lower bound to the performance
Can we analyse it independently?

Stochastic vs. Deterministic Policies

$$J_D(\pi) = \mathbb{E}_{s\sim d^\pi}[r(s, \pi(s))]$$

Deterministic Policy Gradient

$$\begin{aligned}\nabla_\theta J_D(\theta) &= \sum_s d^\pi(s) \nabla_\theta \pi_\theta(s) \nabla_a q^\pi(s,a)|_{a=\pi_\theta(s)} \\ &= \mathbb{E}_{s\sim d^\pi}[\nabla_\theta \pi_\theta(s) \nabla_a q^\pi(s,a)|_{a=\pi_\theta(s)}]\end{aligned}$$

Issues:

- We need to be able to differentiate the model
- Explicitly force exploration at the level of actions

Stochastic vs. Deterministic Policies

Plug it into an actor-critic framework
$\implies$ Use TD(0) to update a parametric representation of q^π

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

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

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

Softmax Operator

$$v^\star(s) = \max_a \left\{ r(s,a) + \gamma \sum_y p(y|s,a) v^\star(y) \right\}$$

replace $\max$ with "*softmax*" operator

$$v^\star(s) = \frac{1}{\eta} \log \left(\sum_a \exp \left[\eta \left(r(s,a) + \gamma \sum_y p(y|s,a) v^\star(y) \right) \right] \right)$$

[Marcus et al., 1997, Ruszczyński, 2010, Ziebart et al., 2010, Ziebart, 2010, Braun et al., 2011, Azar et al., 2012, Rawlik et al., 2012, Fox et al., 2016, Asadi and Littman, 2017, Haarnoja et al., 2017, Schulman et al., 2017, Nachum et al., 2017]

Entropy Regularization

$$\max_{\pi} \left\{ J(\pi) = \mathbb{E}\left[\sum_{t=1}^{+\infty} \gamma^{t-1} r_t + \alpha \Omega(\pi(s_t, \cdot)) \right] \right\}$$

The two approaches are connected by Lagrangian duality when

$$\Omega(\pi(s, \cdot)) = \sum_a \pi(s, a) \log \pi(s, a)$$ *negative entropy*

Entropy Regularization

$$\max_{\pi} \left\{ J(\pi) = \mathbb{E}\left[\sum_{t=1}^{+\infty} \gamma^{t-1} r_t + \alpha \Omega(\pi(s_t, \cdot)) \right] \right\}$$

The two approaches are connected by Lagrangian duality when

$$\Omega(\pi(s, \cdot)) = \sum_{a} \pi(s, a) \log \pi(s, a) \qquad \textit{negative entropy}$$

Results: [Neu et al., 2017]

- Existence and uniqueness
- Well-defined contractive DP operator
- Policy Gradient Theorem

Entropy Regularization

Optimal policy:

$$\pi^\star(s,a) \propto \exp\left[\eta\left(r(s,a) + \gamma \mathbb{E}_s'[v^\star(s')]\right)\right]$$

Note:

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

$$v^\pi(s) = \mathbb{E}_{a\sim\pi}[q^\pi(s,a)] - \Omega(\pi(s,\cdot))$$

Soft-Actor Critic

1. Train the value function v

$$\arg\min_{\psi} \in \mathbb{E}_{s_t \sim H}\left[\frac{1}{2}\left(v_\psi(s_t) + \mathbb{E}_{a_t \sim \pi_\phi}[q_\theta(s_t, a_t) - \log \pi_\phi(s_t, a_t)]\right)^2\right]$$

2. Train the action-value function q^π

$$\arg\min_{\theta} \mathbb{E}_{(s,a) \in H}\left[\frac{1}{2}\left(q_\theta(s_t, a_t) - (r(s_t, a_t) + \gamma \mathbb{E}[v_{\overline{\psi}}(s')])\right)^2\right]$$

! fix the target network (e.g., DQN) $\rightarrow$ increase stability / break dependences

3. Fit the new policy

$$\arg\min_{\phi} \mathbb{E}_{s \in H}\left[D_{KL}(\pi_\psi \| \exp[\eta q_\psi]/Z)[s]\right]$$

Path-Consistency Learning

[Nachum et al., 2017]

Suppose the MDP is deterministic (otherwise take a conditional expectation w.r.t. to history)

For any $v^\star, \pi^\star$ optimizing the regularized objective

$$v^\star(s) - \gamma v^\star(s') = r(s,a) - \eta \log \pi^\star(s,a)$$

$$v^\star(s_1) - \gamma^{t-1} v^\star(s_t) = \sum_{t=1}^{t-1} \gamma^{i-1} \left(r(s_i, a_i) - \eta \log \pi^\star(s_i, a_i) \right)$$

! if (π, v) satisfies the *path consistency* for every (s,a), then $\pi = \pi^\star$ and $v = v^\star$

Path-Consistency Learning

- Maintain two sets of parameters (ϕ, θ): $\theta \mapsto \pi_\theta$, $\phi \mapsto v_\phi$
- Minimize the consistency error

$$\min_{\phi,\theta} O_{PCL}(\phi, \theta, H) = \sum_{s_{i:i+d} \in E_H} \frac{1}{2} C(s_{i:i+d}, \phi, \theta)^2$$

where E_H is the set of (sub)trajectories and

$$C(s_{i:i+d}, \phi, \theta) = -v_\phi(s_i) + \gamma^d v_\phi(s_{i+d}) + \sum_{j=0}^{d-1} \gamma^j \left(r(s_{i+j}, a_{i+j}) - \eta \log \pi_\theta(s_{a+j}, a_{i+j})\right)$$

Path-Consistency Learning

- Maintain two sets of parameters (ϕ, θ): $\theta \mapsto \pi_\theta$, $\phi \mapsto v_\phi$
- Minimize the consistency error

$$\min_{\phi,\theta} O_{PCL}(\phi, \theta, H) = \sum_{s_{i:i+d} \in E_H} \frac{1}{2} C(s_{i:i+d}, \phi, \theta)^2$$

where E_H is the set of (sub)trajectories and

$$C(s_{i:i+d}, \phi, \theta) = -v_\phi(s_i) + \gamma^d v_\phi(s_{i+d}) + \sum_{j=0}^{d-1} \gamma^j \left(r(s_{i+j}, a_{i+j}) - \eta \log \pi_\theta(s_{a+j}, a_{i+j})\right)$$

In practice:

- Use replay buffer
- Update incrementally $\implies$ semi-batch

Path-Consistency Learning

- Maintain two sets of parameters (ϕ, θ): $\theta \mapsto \pi_\theta$, $\phi \mapsto v_\phi$
- Minimize the consistency error

$$\min_{\phi,\theta} O_{PCL}(\phi, \theta, H) = \sum_{s_{i:i+d} \in E_H} \frac{1}{2} C(s_{i:i+d}, \phi, \theta)^2$$

where E_H is the set of (sub)trajectories and

$$C(s_{i:i+d}, \phi, \theta) = -v_\phi(s_i) + \gamma^d v_\phi(s_{i+d}) + \sum_{j=0}^{d-1} \gamma^j \left(r(s_{i+j}, a_{i+j}) - \eta \log \pi_\theta(s_{a+j}, a_{i+j})\right)$$

In practice:

- Use replay buffer
- Update incrementally $\implies$ semi-batch

Can be extended to different regularizers (e.g., Shannon entropy, Tsallis entropy [Chow et al., 2018])

Regularized Markov Decision Processes

[Geist et al., 2019]

Bellman operator

$$L^\pi v(s) = \sum_a \pi(s,a)\left(r(s,a) + \gamma \sum_y p(y|s,a) v^\pi(y)\right) = \sum_a \pi(s,a) q^\pi(s,a)$$

Optimal Bellman operator

$$L^\star v(s) = \max_a \left\{ r(s,a) + \gamma \sum_y p(y|s,a) v^\star(y) \right\}$$

Greedy policy

$$L^\star v = L_{\pi'} v \iff \pi' \in \arg\max_\pi L^\pi v$$

Regularized Markov Decision Processes

Regularizer

$$\Omega : \mathcal{P}(\mathcal{A}) \to \mathcal{S} \qquad \textit{strongly convex function}$$

Legendre-Fenchel transform (or convex conjugate)

$$\Omega^\star : \mathbb{R}^A \to \mathbb{R}$$

$$\forall q \in \mathbb{R}^A, \qquad \Omega^\star(q) = \max_{z \in \mathcal{P}(\mathcal{A})} \left\{ \sum_s z(a)q(a) - \Omega(z) \right\}$$

Regularized Markov Decision Processes

Regularizer

$$\Omega : \mathcal{P}(\mathcal{A}) \to \mathcal{S} \qquad \textit{strongly convex function}$$

Legendre-Fenchel transform (or convex conjugate)

$$\Omega^\star : \mathbb{R}^A \to \mathbb{R}$$

$$\forall q \in \mathbb{R}^A, \qquad \Omega^\star(q) = \max_{z \in \mathcal{P}(\mathcal{A})} \left\{ \sum_s z(a) q(a) - \Omega(z) \right\}$$

Property of strongly convex functions: *unique maximizing argument*

$$\nabla\Omega^\star \text{ is Lipschitz and} \qquad \nabla\Omega^\star(q) = \arg\max_{z \in \mathcal{P}(\mathcal{A})} \left\{ \sum_s z(a) q(a) - \Omega(z) \right\}$$

Regularized Markov Decision Processes

Examples:

	$\Omega(\pi(s,\cdot))$	$\Omega^\star(q(s,\cdot))$
Negative entropy	$\sum_a \pi_s(a)\log\pi(s,a)$	$\log\sum_a \exp q(s,a)$
	$\nabla\Omega^\star(q(s,\cdot)) = \frac{\exp q(s,a)}{\sum_b \exp q(s,b)}$	i.e., softmax
KL-divergence between π and uniform	$\sum_a \pi(s,a)\log\pi(s,a) + \log(A)$	$\ln\sum_a \frac{1}{A}\exp[[q(s,a)]$
	$\nabla\Omega^\star$ is Mellowmax [Asadi and Littman, 2017]	
Tsallis entropy $(q=2, k=1/2)$	$\frac{1}{2}(\lVert\pi(s,\cdot)\rVert_2^2 - 1)$	
	$\nabla\Omega^\star$ is the sparsemax [Chow et al., 2018]	

Regularized Markov Decision Processes

Regularized Bellman operators w.r.t. Ω

$$L_\Omega^\pi v(s) = L^\pi v(s) - \Omega(\pi(s,\cdot)) = \sum_a \pi(s,a) q^\pi(s,a) - \Omega(\pi(s,\cdot))$$

Regularized Optimal Bellman operators w.r.t. Ω

$$L_\Omega^\star v(s) = \max_\pi L_\Omega^\pi v[s] = \Omega^\star(q(s,\cdot))$$

Greedy policy

$$\pi' = \mathcal{G}_\Omega(v) = \nabla \Omega^\star(q) \iff L_\Omega^{\pi'} v = L_\Omega^\star v$$

We have the usual properties for L_Ω^π: *affine, monotonicity, distributivity, contraction*

Regularized Markov Decision Processes

Regularized value functions: $v_\Omega^\pi = L_\Omega^\pi v_\Omega^\pi$

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

$$v^\pi(s) = \mathbb{E}_{a\sim\pi}[q^\pi(s,a)] - \Omega(\pi(s,\cdot))$$

Regularized optimal value functions: $v_\Omega^\star = L_\Omega^\star v_\Omega^\star$

$$q_\Omega^\star(s,a) = r(s,a) + \gamma \sum_y p(y|s,a) v_\Omega^\star(y)$$

$$v_\Omega^\star(s) = \Omega^\star(q^\star(s,\cdot))$$

Optimality

$$\pi_\Omega^\star = \mathcal{G}_\Omega(v_\Omega^\star) \text{ is optimal}$$

$$\forall \pi, \qquad v_\Omega^{\pi_\Omega^\star} = v_\Omega^\star \geq v_\Omega^\pi$$

Regularized Markov Decision Processes

- This explains many recent algorithms
- They can be seen as a particular instance of Modified Policy Iteration

$$\pi_{k+1} = \mathcal{G}_\Omega(v_k)$$
$$v_{k+1} = (L_\Omega^{\pi_{k+1}})^m v_k$$

- Up to modifications for make them practical

- Soft Q-learning with negative entropy [Fox et al., 2016, Schulman et al., 2017a] or Tsallis entropy [Lee et al., 2018]

- SAC with entropic regularizer [Haarnoja et al., 2018]

- Algorithms based on path consistency [Nachum et al., 2017, Chow et al., 2018]

Regularized Markov Decision Processes

Issues:

- Regularization as defined above is changing the objective
- We obtain a *different optimal policy*
- Should be an algorithm trick and not a change in the objective
  - i.e., estimate the original optimal policy by solving a series of regularized problems

Regularized Markov Decision Processes

Issues:

- Regularization as defined above is changing the objective
- We obtain a *different optimal policy*
- Should be an algorithm trick and not a change in the objective
  - i.e., estimate the original optimal policy by solving a series of regularized problems

Solution:

- Consider a time varying regularized
- Penalize the difference between policy π and the one at previous iteration (*already seen*)

Regularized Markov Decision Processes

Bregman divergence

$$\Omega_{\pi'_s}(\pi_s) = D_\Omega(\pi_s \| \pi'_s) = \Omega(\pi_s) - \Omega(\pi'_s) - \nabla\Omega(\pi')^\mathsf{T}(\pi_s - \pi'_s)$$

Example:

negative entropy $\implies$ $\qquad \Omega_{\pi'_s}(\pi_s) = D_{KL}(\pi \| \pi')[s]$

Regularized Markov Decision Processes

Bregman divergence

$$\Omega_{\pi'_s}(\pi_s) = D_\Omega(\pi_s \| \pi'_s) = \Omega(\pi_s) - \Omega(\pi'_s) - \nabla\Omega(\pi')^\mathsf{T}(\pi_s - \pi'_s)$$

Example:

negative entropy $\implies$ $\qquad \Omega_{\pi'_s}(\pi_s) = D_{KL}(\pi\|\pi')[s]$

Policy Iteration improvement

$$\begin{aligned} \pi_{k+1} &= \mathcal{G}_{\Omega_{\pi_k}}(v_k) \\ &= \arg\max_\pi \sum_a \pi(s,a) q_k(s,a) - D_\Omega(\pi\|\pi_k) \end{aligned}$$

Regularized Markov Decision Processes

Bregman divergence

$$\Omega_{\pi'_s}(\pi_s) = D_\Omega(\pi_s \| \pi'_s) = \Omega(\pi_s) - \Omega(\pi'_s) - \nabla\Omega(\pi')^\mathsf{T}(\pi_s - \pi'_s)$$

Example:

negative entropy $\implies$ $\Omega_{\pi'_s}(\pi_s) = D_{KL}(\pi \| \pi')[s]$

Policy Iteration improvement

$$\begin{aligned} \pi_{k+1} &= \mathcal{G}_{\Omega_{\pi_k}}(v_k) \\ &= \arg\max_\pi \sum_a \pi(s,a) q_k(s,a) - D_\Omega(\pi \| \pi_k) \end{aligned}$$

! *similar to Mirror Descent in proximal form with $-q_k$ as gradient!*

$\implies$ *estimates the original optimal policy*

Regularized Markov Decision Processes

- Common framework
- Algorithms are either Mirror Descent or Dual Averaging [Neu et al., 2017]

TRPO can be seen as a mirror descent approach $\Longrightarrow$ guarantees of convergence
Similar interpretation (as dual averaging algorithm) for DPP [Azar et al., 2012] and MPO [Abdolmaleki et al., 2018].

Regularized Policy Gradient

$$\nabla J_\Omega(\pi) = \sum_s d^\pi(s) \sum_a \pi(s,a) \left(q_\Omega^\pi(s,a) - \frac{\partial \Omega(\pi(s,\cdot))}{\partial \pi(s,a)} \right) \nabla \log \pi(s,a)$$

Possible to replace with Bregman divergence $\implies$ *convergence to original policy*

Resources

Reinforcement Learning

- Books
  - Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1994
  - Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*. MIT press Cambridge, 2 edition, 2018
  - Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. II*. Athena Scientific, 3rd edition, 2007
  - Csaba Szepesvari. *Algorithms for Reinforcement Learning*. Morgan and Claypool Publishers, 2010
- Courses
  - Sergey Levine. Cs 294: Deep reinforcement learning. http://rail.eecs.berkeley.edu/deeprlcourse-fa17/index.html
  - Emma Brunskill. Cs234 reinforcement learning winter 2019. http://web.stanford.edu/class/cs234/index.html
  - Alessandro Lazaric. Mva reinforcement learning. http://chercheurs.lille.inria.fr/~lazaric/Webpage/Teaching.html
  - Alexandre Proutiere. Reinforcement learning: A graduate course. http://www.it.uu.se/research/systems_and_control/education/2017/relearn/

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

Kavosh Asadi and Michael L. Littman. An alternative softmax operator for reinforcement learning. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 243–252. PMLR, 2017.

Mohammad Gheshlaghi Azar, Vicenç Gómez, and Hilbert J Kappen. Dynamic policy programming. *Journal of Machine Learning Research*, 13(Nov):3207–3245, 2012.

Francis Bach. Stochastic optimization: Beyond stochastic gradients and convexity part i. 2016.

Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pages 30–37. Elsevier, 1995.

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

Dimitri P Bertsekas and Sergey Ioffe. Temporal differences-based policy iteration and applications in neuro-dynamic programming. *Lab. for Info. and Decision Systems Report LIDS-P-2349, MIT, Cambridge, MA*, 1996.

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

Shalabh Bhatnagar, Richard S. Sutton, Mohammad Ghavamzadeh, and Mark Lee. Natural actor-critic algorithms. *Automatica*, 45(11):2471–2482, 2009.

Emma Brunskill. Cs234 reinforcement learning winter 2019. http://web.stanford.edu/class/cs234/index.html.

Apostolos N Burnetas and Michael N Katehakis. Optimal adaptive policies for markov decision processes. *Mathematics of Operations Research*, 22(1):222–255, 1997.

X.R. Cao. *Stochastic Learning and Optimization: A Sensitivity-Based Approach*. International Series on Discrete Event Dynamic Systems, v. 17. Springer US, 2007. ISBN 9780387690827.

Yinlam Chow, Ofir Nachum, and Mohammad Ghavamzadeh. Path consistency learning in tsallis entropy regularized mdps. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 978–987. PMLR, 2018.

Imre Csiszar and János Körner. *Information theory: coding theorems for discrete memoryless systems*. Cambridge University Press, 2011.

Thomas Degris, Martha White, and Richard S. Sutton. Off-policy actor-critic. *CoRR*, abs/1205.4839, 2012.

Christos Dimitrakakis and Michail G. Lagoudakis. Rollout sampling approximate policy iteration. *Machine Learning*, 72(3):157–171, 2008.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.

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.

Alan Fern, Sung Wook Yoon, and Robert Givan. Approximate policy iteration with a policy language bias. In *NIPS*, pages 847–854. MIT Press, 2003.

Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'16, pages 202–211, Arlington, Virginia, United States, 2016. AUAI Press. ISBN 978-0-9966431-1-5. URL `http://dl.acm.org/citation.cfm?id=3020948.3020970`.

Thomas Furmston and David Barber. A unifying perspective of parametric policy search methods for markov decision processes. In *NIPS*, pages 2726–2734, 2012.

Victor Gabillon, Alessandro Lazaric, Mohammad Ghavamzadeh, and Bruno Scherrer. Classification-based policy iteration with a critic. In *ICML*, pages 1049–1056. Omnipress, 2011.

Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A theory of regularized markov decision processes. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 2160–2169. PMLR, 2019.

Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *ICLR*. OpenReview.net, 2018.

Shixiang Gu, Timothy P. Lillicrap, Zoubin Ghahramani, Richard E. Turner, and Sergey Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. In *ICLR*. OpenReview.net, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 1856–1865. 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.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013.

Filip Jurcícek. Reinforcement learning for spoken dialogue systems using off-policy natural gradient method. In *SLT*, pages 7–12. IEEE, 2012.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, pages 267–274, 2002.

Sham M Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538, 2002.

Daphne Koller and Ronald Parr. Policy iteration for factored mdps. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 326–334. Morgan Kaufmann Publishers Inc., 2000.

Michail G Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of machine learning research*, 4 (Dec):1107–1149, 2003a.

Michail G Lagoudakis and Ronald Parr. Reinforcement learning as classification: Leveraging modern classifiers. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 424–431, 2003b.

Alessandro Lazaric. Mva reinforcement learning. http://chercheurs.lille.inria.fr/˜lazaric/Webpage/Teaching.html.

Alessandro Lazaric, Mohammad Ghavamzadeh, and Rémi Munos. Finite-sample analysis of least-squares policy iteration. *Journal of Machine Learning Research*, 13:3041–3074, 2012.

Kyungjae Lee, Sungjoon Choi, and Songhwai Oh. Sparse markov decision processes with causal sparse tsallis entropy regularization for reinforcement learning. *IEEE Robotics and Automation Letters*, 3(3):1466–1473, 2018.

Sergey Levine. Cs 294: Deep reinforcement learning. http://rail.eecs.berkeley.edu/deeprlcourse-fa17/index.html.

Hao Liu, Yihao Feng, Yi Mao, Dengyong Zhou, Jian Peng, and Qiang Liu. Action-dependent control variates for policy optimization via stein identity. In *ICLR*. OpenReview.net, 2018.

James Martens. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2014.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. In *NIPS*, pages 2772–2782, 2017.

Gergely Neu, Anders Jonsson, and Vicenç Gómez. A unified view of entropy-regularized markov decision processes. *CoRR*, abs/1705.07798, 2017.

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

Yann Ollivier. True asymptotic natural gradient optimization. *arXiv preprint arXiv:1712.08449*, 2017.

Yann Ollivier, Ludovic Arnold, Anne Auger, and Nikolaus Hansen. Information-geometric optimization algorithms: A unifying picture via invariance principles. *The Journal of Machine Learning Research*, 18(1): 564–628, 2017.

Matteo Papini, Matteo Pirotta, and Marcello Restelli. Adaptive batch size for safe policy gradients. In *Advances in Neural Information Processing Systems*, pages 3591–3600, 2017.

Matteo Papini, Damiano Binaghi, Giuseppe Canonaco, Matteo Pirotta, and Marcello Restelli. Stochastic variance-reduced policy gradient. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 4023–4032. PMLR, 2018.

Matteo Papini, Matteo Pirotta, and Marcello Restelli. Smoothing policies and safe policy gradients. *CoRR*, abs/1905.03231, 2019.

Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.

Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008a.

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

Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

Matteo Pirotta, Marcello Restelli, and Luca Bascetta. Adaptive step-size for policy gradient methods. In *Advances in Neural Information Processing Systems*, pages 1394–1402, 2013.

Alexandre Proutiere. Reinforcement learning: A graduate course. http://www.it.uu.se/research/systems_and_control/education/2017/relearn/.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1994.

Bruno Scherrer, Mohammad Ghavamzadeh, Victor Gabillon, Boris Lesner, and Matthieu Geist. Approximate modified policy iteration and its application to the game of tetris. *Journal of Machine Learning Research*, 16: 1629–1676, 2015.

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

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

John Schulman, Xi Chen, and Pieter Abbeel. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017a.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017b.

Zebang Shen, Alejandro Ribeiro, Hamed Hassani, Hui Qian, and Chao Mi. Hessian aided policy gradient. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 5729–5738. PMLR, 2019.

Richard Sutton. Introduction to reinforcement learning with function approximation. Technical report, Tutorial at the Conference on Neural Information Processing Systems, 2015.

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.

Csaba Szepesvari. *Algorithms for Reinforcement Learning*. Morgan and Claypool Publishers, 2010.

Philip S. Thomas and Emma Brunskill. Policy gradient methods for reinforcement learning with function approximation and action-dependent baselines. *CoRR*, abs/1706.06643, 2017.

George Tucker, Surya Bhupatiraju, Shixiang Gu, Richard E. Turner, Zoubin Ghahramani, and Sergey Levine. The mirage of action-dependent baselines in reinforcement learning. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 5022–5031. PMLR, 2018.

Quan Vuong, Yiming Zhang, and Keith W. Ross. Supervised Policy Update for Deep Reinforcement Learning. In *International Conference on Learning Representations*, 2019.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Rémi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. In *ICLR*. OpenReview.net, 2017.

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

Cathy Wu, Aravind Rajeswaran, Yan Duan, Vikash Kumar, Alexandre M. Bayen, Sham Kakade, Igor Mordatch, and Pieter Abbeel. Variance reduction for policy gradient with action-dependent factorized baselines. In *ICLR*. OpenReview.net, 2018.

Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems*, pages 5279–5288, 2017.

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.