

The Provable Effectiveness of Policy Gradient Methods in Reinforcement Learning

Sham Kakade

University of Washington & Microsoft Research

(with **Alekh Agarwal, Jason Lee, and Gaurav Mahajan**)

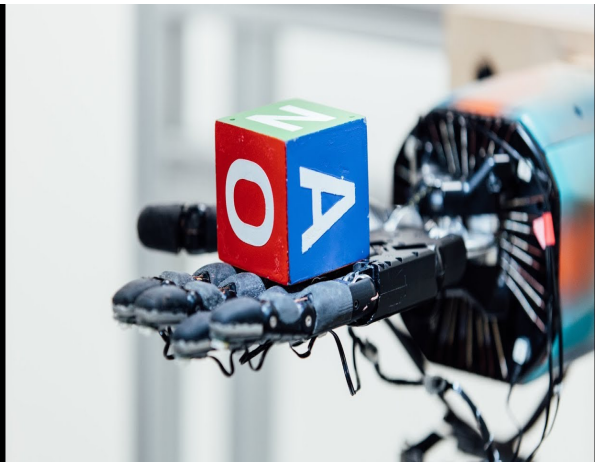
Policy Optimization in RL



[AlphaZero, Silver et.al, 17]



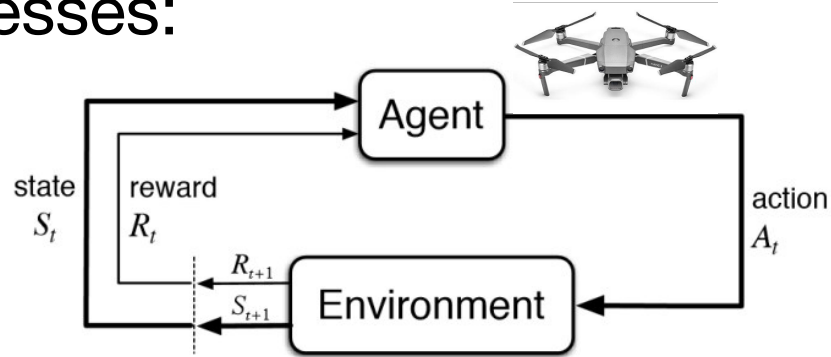
[OpenAI Five, 18]



[OpenAI,19]

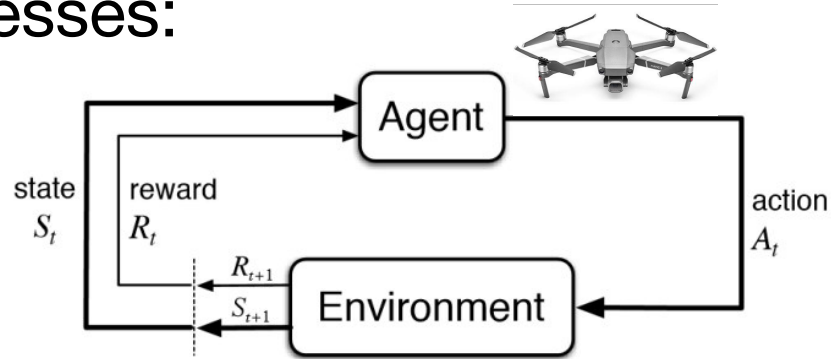
Markov Decision Processes:

a framework for RL



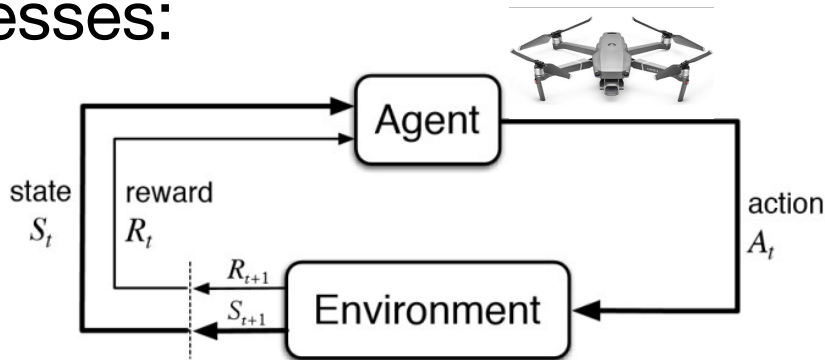
Markov Decision Processes: a framework for RL

- A **policy**:
 $\pi : \text{States} \rightarrow \text{Actions}$



Markov Decision Processes: a framework for RL

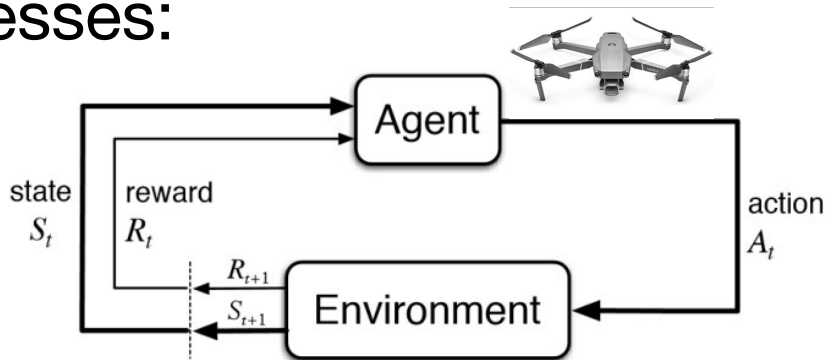
- A **policy**:
 $\pi : \text{States} \rightarrow \text{Actions}$
- We execute π to obtain a trajectory:
 $s_0, a_0, r_0, s_1, a_1, r_1 \dots$



Markov Decision Processes: a framework for RL

- A **policy**:
 $\pi : \text{States} \rightarrow \text{Actions}$
- We execute π to obtain a trajectory:
 $s_0, a_0, r_0, s_1, a_1, r_1 \dots$
- Total **γ -discounted reward**:

$$V^\pi(s_0) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$



Markov Decision Processes: a framework for RL

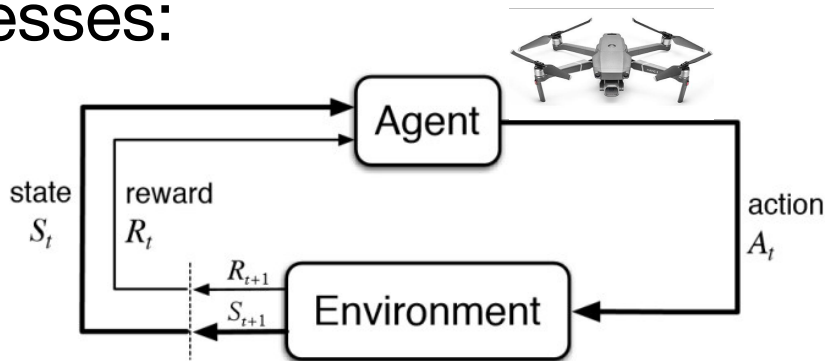
- A **policy**:
 $\pi : \text{States} \rightarrow \text{Actions}$
- We execute π to obtain a trajectory:


$s_0, a_0, r_0, s_1, a_1, r_1 \dots$

- Total **γ -discounted reward**:


$$V^\pi(s_0) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$

- **Goal**: Find a policy π that maximizes our value $V^\pi(s_0)$




	0	1	2	3	4	5
0	Start		Wall			+1
1		Wall				-1
2		Wall			Wall	
3						

Challenges in RL

	0	1	2	3	4	5
0	Start		Wall			+1
1		Wall				-1
2		Wall			Wall	
3						

Challenges in RL

1. **Exploration**
(the environment may be unknown)

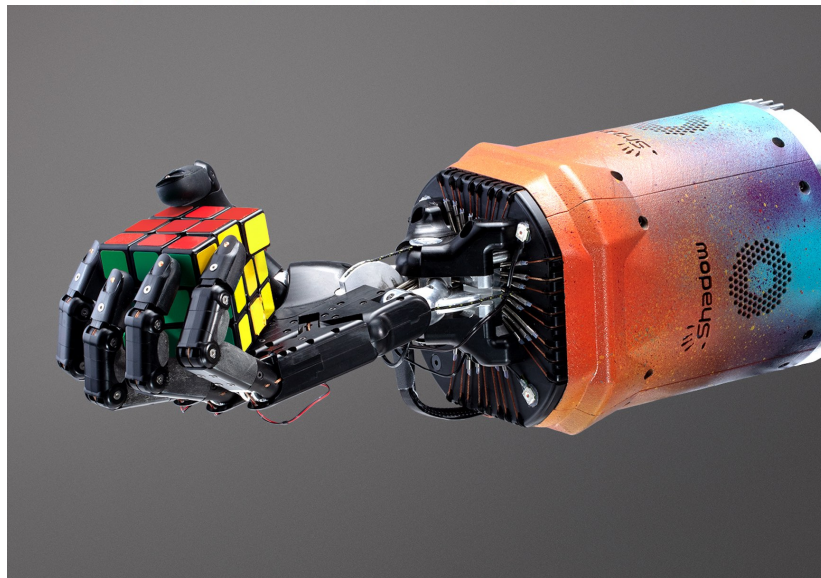
	0	1	2	3	4	5
0	Start		Wall			+1
1		Wall				-1
2		Wall			Wall	
3						

Challenges in RL

1. **Exploration**
(the environment may be unknown)
2. **Credit assignment problem**
(due to delayed rewards)

Dexterous Robotic Hand Manipulation

OpenAI, 2019



Challenges in RL

1. **Exploration**
(the environment may be unknown)
2. **Credit assignment problem**
(due to delayed rewards)
3. **Large state/action spaces:**
hand state: joint angles/velocities
cube state: configuration
actions: forces applied to actuators

Part 0: Background

RL, Deep RL, and Supervised Learning (SL)

The “Tabular” Dynamic Programming approach

State s : (joint angles, ... cube config,...)	Action a : (forces at joints)	$Q^\pi(s, a)$: state-action value “one step look-ahead value” using π
(31°, 12°, ..., 8134, ...)	(1.2 Newton, 0.1 Newton, ...)	8 units of reward
⋮	⋮	⋮

- “Tabular” dynamic programming approach: (with known model)
 1. For **every entry** in the table, compute the state-action value:

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a \right]$$

2. **Update the policy** π to be greedy: $\pi(s) \leftarrow \operatorname{argmax}_a Q^\pi(s, a)$

The “Tabular” Dynamic Programming approach

State s : (joint angles, ... cube <u>config</u> ,...)	Action a : (forces at joints)	$Q^\pi(s, a)$: state-action value “one step look-ahead value” using π
(31°, 12°, ..., 8134, ...)	(1.2 Newton, 0.1 Newton, ...)	8 units of reward
⋮	⋮	⋮

- “Tabular” dynamic programming approach: (with known model)
 1. For **every entry** in the table, compute the state-action value:

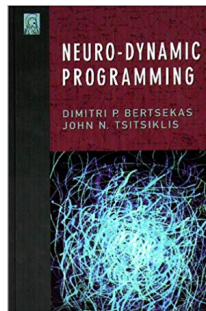
$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a \right]$$

2. **Update the policy** π to be greedy: $\pi(s) \leftarrow \operatorname{argmax}_a Q^\pi(s, a)$
- **Generalization**: how can we deal with this infinite table?
Use sampling/supervised learning + deep learning.

The “Tabular” Dynamic Programming approach

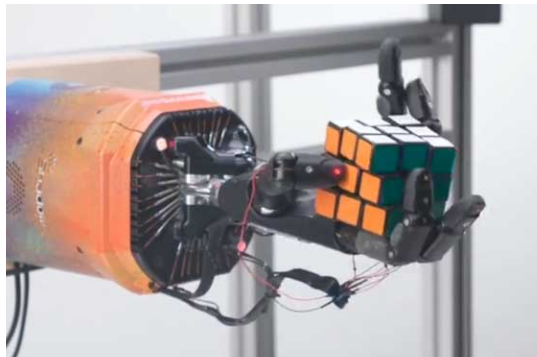
“deep RL”?

[Bertsekas & Tsitsiklis '97] provides first systematic analysis of RL with (worst case) “function approximation”.



- “Tabular” dynamic programming approach: (with known model)
 1. For **every entry** in the table, compute the state-action value:
$$Q^{\pi}(s, a) = \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a \right]$$
 2. **Update the policy** π to be greedy: $\pi(s) \leftarrow \operatorname{argmax}_a Q^{\pi}(s, a)$
- **Generalization**: how can we deal with this infinite table?
Use sampling/supervised learning + deep learning.

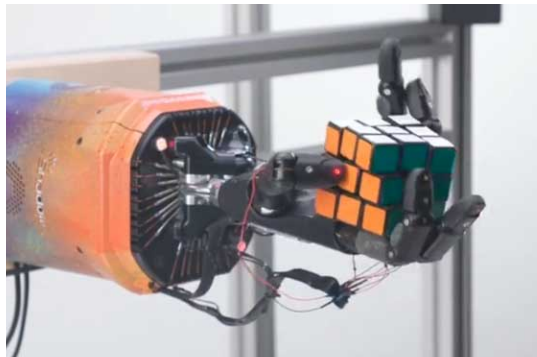
In practice, policy gradient methods rule...



In practice, policy gradient methods rule...

- They are the most effective method for obtaining state of the art.

$$\theta \leftarrow \theta + \eta \nabla_{\theta} V^{\pi_{\theta}}(s_0)$$

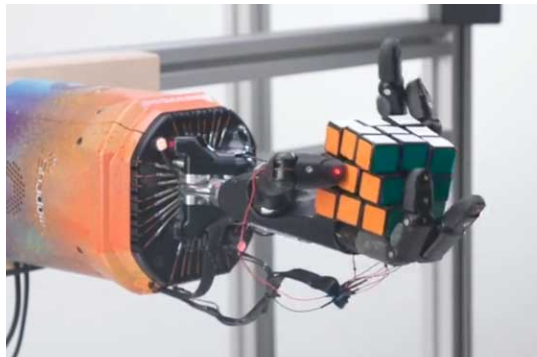


In practice, policy gradient methods rule...

- They are the most effective method for obtaining state of the art.

$$\theta \leftarrow \theta + \eta \nabla_{\theta} V^{\pi_{\theta}}(s_0)$$

- Why do we like them?

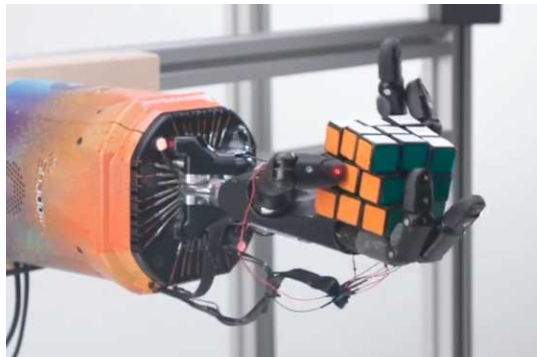


In practice, policy gradient methods rule...

- They are the most effective method for obtaining state of the art.

$$\theta \leftarrow \theta + \eta \nabla_{\theta} V^{\pi_{\theta}}(s_0)$$

- Why do we like them?
 - they easily deal with large state/action spaces (through the neural net parameterization)

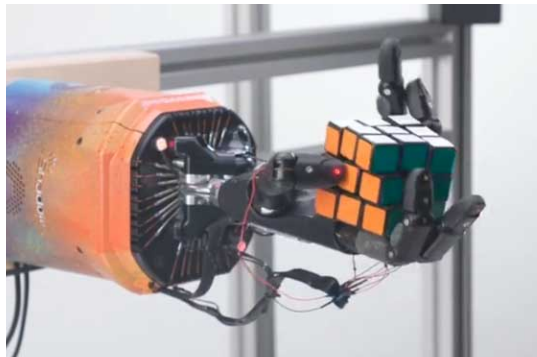


In practice, policy gradient methods rule...

- They are the most effective method for obtaining state of the art.

$$\theta \leftarrow \theta + \eta \nabla_{\theta} V^{\pi_{\theta}}(s_0)$$

- Why do we like them?
 - they easily deal with large state/action spaces (through the neural net parameterization)
 - We can estimate the gradient using only simulation of our current policy π_{θ} (the expectation is under the state actions visited under π_{θ})

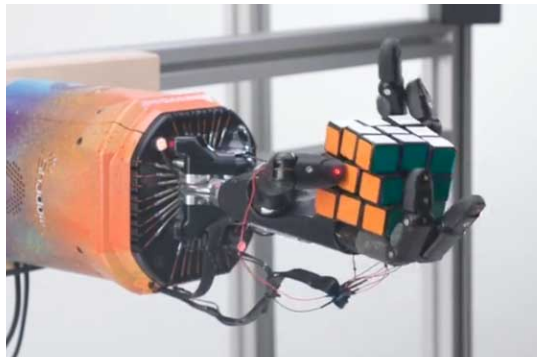


In practice, policy gradient methods rule...

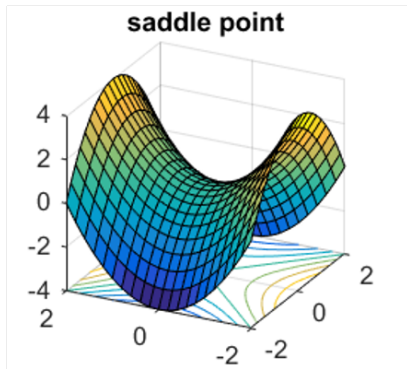
- They are the most effective method for obtaining state of the art.

$$\theta \leftarrow \theta + \eta \nabla_{\theta} V^{\pi_{\theta}}(s_0)$$

- Why do we like them?
 - they easily deal with large state/action spaces (through the neural net parameterization)
 - We can estimate the gradient using only simulation of our current policy π_{θ} (the expectation is under the state actions visited under π_{θ})
 - They directly optimize the cost function of interest!

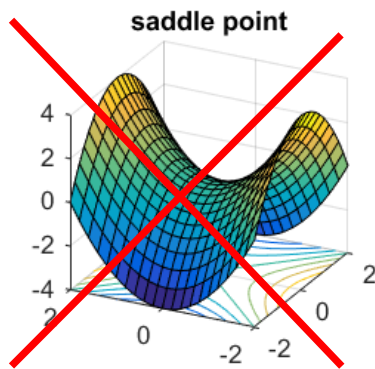


The Optimization Landscape



Supervised Learning:

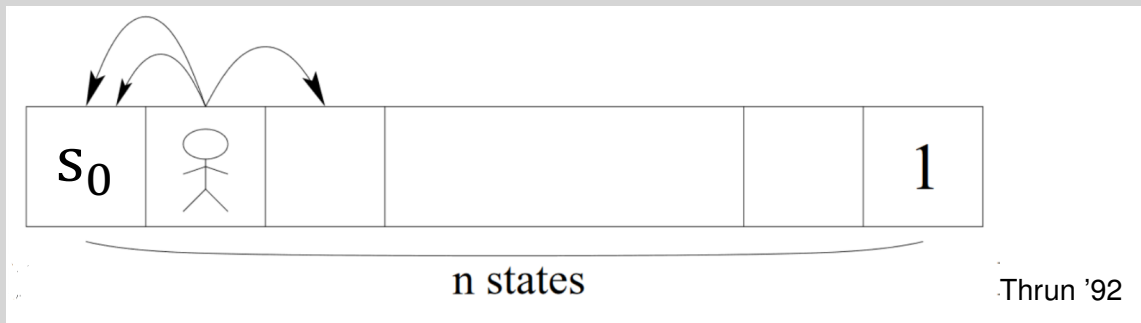
- Gradient descent tends to 'just work' in practice (not sensitive to initialization)
- Saddle points not a problem...



Reinforcement Learning:

- In many real RL problems, we have "very" flat regions.
- Gradients can be exponentially small in the "horizon" due to lack of exploration.

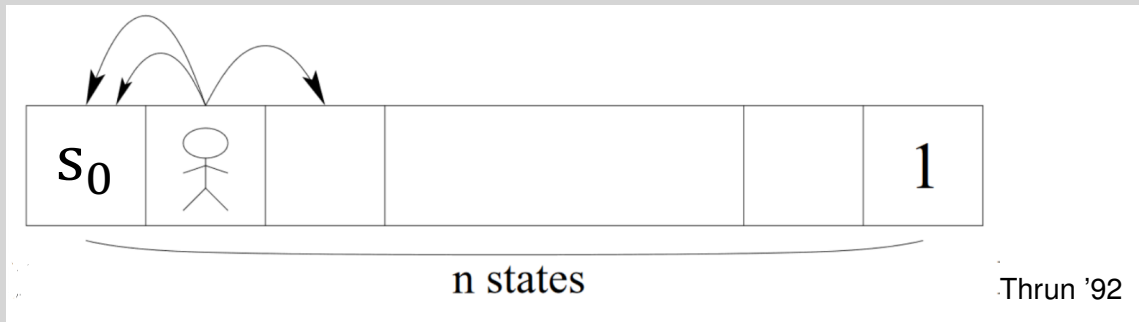
The Optimization Landscape



Lemma: [Higher order vanishing gradients]

Suppose there are $S \leq 1/(1 - \gamma)$ states in the MDP. With random initialization, all k -th higher-order gradients, for $k < S/\log(S)$, the spectral norm of the gradients are bounded by $2^{-S/2}$.

The Optimization Landscape



Lemma: [Higher order vanishing gradients]

Suppose there are $S \leq 1/(1 - \gamma)$ states in the MDP. With random initialization, all k -th higher-order gradients, for $k < S/\log(S)$, the spectral norm of the gradients are bounded by $2^{-S/2}$.

This talk: Can we get any handle on policy gradient methods because they are one of the most widely used practical tools?

This talk

We provide provable **global convergence** and **generalization** guarantees of (nonconvex) policy gradient methods.

This talk

We provide provable **global convergence** and **generalization** guarantees of (nonconvex) policy gradient methods.

- **Part – I:** small state spaces + exact gradients
curvature + non-convexity
 - Vanilla PG
 - PG with regularization
 - Natural Policy Gradient

This talk

We provide provable **global convergence** and **generalization** guarantees of (nonconvex) policy gradient methods.

- **Part – I:** small state spaces + exact gradients
curvature + non-convexity
 - Vanilla PG
 - PG with regularization
 - Natural Policy Gradient
- **Part – II:** large state spaces
generalization and distribution shift
 - Function approximation/deep nets? Why use PG?

Part I: Small State Spaces

(and the softmax policy class)

Policy Optimization over the "softmax" policy class (let's start simple!)

- Simplest way to parameterize the simplex, without constraints.

Policy Optimization over the "softmax" policy class (let's start simple!)

- Simplest way to parameterize the simplex, without constraints.
- $\pi_\theta(a | s)$ is the probability of action a given state s

$$\pi_\theta(a | s) = \frac{\exp(\theta_{s,a})}{\sum_{a'} \exp(\theta_{s,a'})}$$

Policy Optimization over the "softmax" policy class (let's start simple!)

- Simplest way to parameterize the simplex, without constraints.
- $\pi_\theta(a | s)$ is the probability of action a given state s

$$\pi_\theta(a | s) = \frac{\exp(\theta_{s,a})}{\sum_{a'} \exp(\theta_{s,a'})}$$

- Complete class: contains every stationary policy

Policy Optimization over the "softmax" policy class (let's start simple!)

- Simplest way to parameterize the simplex, without constraints.

- $\pi_\theta(a | s)$ is the probability of action a given state s

$$\pi_\theta(a | s) = \frac{\exp(\theta_{s,a})}{\sum_{a'} \exp(\theta_{s,a'})}$$

- Complete class: contains every stationary policy

assume
 $\nabla V^{\pi_\theta}(s_0)$
exactly

The policy optimization problem $\max_{\theta} V^{\pi_\theta}(s_0)$ is non-convex.

Do we have global convergence?

Global Convergence of PG for Softmax

$$V^\theta(\mu) = E_{s \sim \mu}[V^\theta(s)]$$

$$\theta \leftarrow \theta + \eta \nabla_\theta V^\theta(\mu)$$

μ - starting
state
dist.

Theorem [Vanilla PG for Softmax Policy class]

Suppose μ has full support over the state space. Then, for all states s ,

$$V^\theta(s) \rightarrow V^*(s)$$

Global Convergence of PG for Softmax

$$V^\theta(\mu) = E_{s \sim \mu}[V^\theta(s)]$$

$$\theta \leftarrow \theta + \eta \nabla_\theta V^\theta(\mu)$$

Theorem [Vanilla PG for Softmax Policy class]

Suppose μ has full support over the state space. Then, for all states s ,

$$V^\theta(s) \rightarrow V^*(s)$$

- Even though problem is non-convex, we have global convergence.
 - proof is detailed/asymptotic

Global Convergence of PG for Softmax

$$V^\theta(\mu) = E_{s \sim \mu}[V^\theta(s)]$$

$$\theta \leftarrow \theta + \eta \nabla_\theta V^\theta(\mu)$$

Theorem [Vanilla PG for Softmax Policy class]

Suppose μ has full support over the state space. Then, for all states s ,

$$V^\theta(s) \rightarrow V^*(s)$$

- Even though problem is non-convex, we have global convergence.
 - proof is detailed/asymptotic
- Rate could be exponentially slow in terms of #states
 - Issue: the softmax can have very flat gradients

Global Convergence: Softmax + Log Barrier regularization

$$L_\lambda(\theta) := V^\theta(\mu) + \frac{\lambda}{SA} \sum_{s,a} \log \pi_\theta(a | s)$$

$$\theta \leftarrow \theta + \eta \nabla L_\lambda(\theta)$$

Theorem [PG: Softmax+Log Barrier]

S : #states, A : #actions, H : Horizon = $1/(1 - \gamma)$

Suppose $\mu = \text{uniform}_S$ and with appropriate settings of λ and η

After $\frac{S^4 A^2 H^6}{\epsilon^2}$ iterations, we have for all s ,

$$V^\theta(s) \geq V^\star(s) - \epsilon$$

Global Convergence: Softmax + Log Barrier regularization

$$L_\lambda(\theta) := V^\theta(\mu) + \frac{\lambda}{SA} \sum_{s,a} \log \pi_\theta(a | s)$$

$$\theta \leftarrow \theta + \eta \nabla L_\lambda(\theta)$$

Theorem [PG: Softmax+Log Barrier]

S : #states, A : #actions, H : Horizon = $1/(1 - \gamma)$

Suppose $\mu = \text{uniform}_S$ and with appropriate settings of λ and η

After $\frac{S^4 A^2 H^6}{\epsilon^2}$ iterations, we have for all s ,

$$V^\theta(s) \geq V^\star(s) - \epsilon$$

- Even though problem is non-convex, we have poly iteration complexity.

Global Convergence: Softmax + Log Barrier regularization

$$L_\lambda(\theta) := V^\theta(\mu) + \frac{\lambda}{SA} \sum_{s,a} \log \pi_\theta(a | s)$$

$$\theta \leftarrow \theta + \eta \nabla L_\lambda(\theta)$$

Theorem [PG: Softmax+Log Barrier]

S : #states, A : #actions, H : Horizon = $1/(1 - \gamma)$

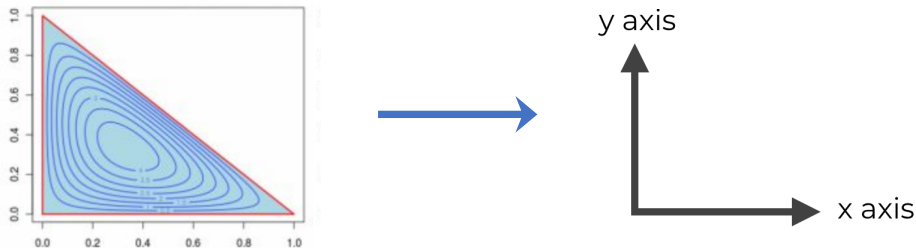
Suppose $\mu = \text{uniform}_S$ and with appropriate settings of λ and η

After $\frac{S^4 A^2 H^6}{\epsilon^2}$ iterations, we have for all s ,

$$V^\theta(s) \geq V^\star(s) - \epsilon$$

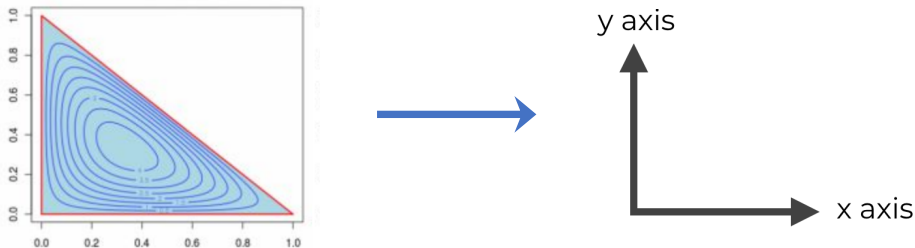
- Even though problem is non-convex, we have poly iteration complexity.
- Log barrier and uniform μ helps with conditioning problems.
 - proof is succinct/ requires showing $\pi_\theta(a | s)$ doesn't become too small.
 - log barrier reg = KL-regularization \neq entropy regularization

Preconditioning: The Natural Policy Gradient (NPG)



- **Practice:** most methods are gradient based, usually variants of:
NPG [K. '01]; TRPO [Schulman '15]; PPO [Schulman '17]

Preconditioning: The Natural Policy Gradient (NPG)



- **Practice:** most methods are gradient based, usually variants of: NPG [K. '01]; TRPO [Schulman '15]; PPO [Schulman '17]
- **NPG warps the distance metric** to stretch the corners out (using the Fisher information metric) to move 'more' near the boundaries. The update is:

$$F(\theta) = E_{s,a \sim \pi_\theta} \left[\nabla \log \pi_\theta(a | s) \nabla \log \pi_\theta(a | s)^\top \right]$$

$$\theta \leftarrow \theta + \eta F(\theta)^{-1} \nabla V^\theta(s_0)$$

NPG and “soft” policy iteration

- The softmax policy class: $\pi_{\theta}(a | s) \propto \exp(\theta_{s,a})$

NPG and “soft” policy iteration

- The softmax policy class: $\pi_{\theta}(a | s) \propto \exp(\theta_{s,a})$
- At iteration t , the NPG update rule:

$$\theta \leftarrow \theta + \eta F(\theta)^{-1} \nabla V^{\theta}(s_0)$$

is equivalent to a “soft” policy iteration update rule:

$$\pi(a | s) \leftarrow \pi(a | s) \frac{\exp(\eta Q^{\pi}(s, a))}{Z}$$

NPG and “soft” policy iteration


- The softmax policy class: $\pi_{\theta}(a | s) \propto \exp(\theta_{s,a})$
- At iteration t , the NPG update rule:

$$\theta \leftarrow \theta + \eta F(\theta)^{-1} \nabla V^{\theta}(s_0)$$

is equivalent to a “soft” policy iteration update rule:

$$\pi(a | s) \leftarrow \pi(a | s) \frac{\exp(\eta Q^{\pi}(s, a))}{Z}$$

$A^{\pi}(s, a)$



What happens for this non-convex update rule?

Global Convergence of NPG

Theorem [NPG]

Set $\eta = (1 - \gamma)^2 \log A$.

For the softmax policy class, we have after T iterations,

$$V^{(T)}(\varphi) \geq V^*(\varphi) - \frac{2}{(1 - \gamma)^2 T}$$

$\forall \varphi$

Global Convergence of NPG

Theorem [NPG]

Set $\eta = (1 - \gamma)^2 \log A$.

For the softmax policy class, we have after T iterations,

$$V^{(T)}(\rho) \geq V^*(\rho) - \frac{2}{(1 - \gamma)^2 T}$$

- Dimension free iteration complexity: (No dependence on S, A, μ)

Global Convergence of NPG

Theorem [NPG]

Set $\eta = (1 - \gamma)^2 \log A$.

For the softmax policy class, we have after T iterations,

$$V^{(T)}(\rho) \geq V^*(\rho) - \frac{2}{(1 - \gamma)^2 T}$$

- Dimension free iteration complexity: (No dependence on S, A, μ)
- Also a “fast rate”.

Global Convergence of NPG

Theorem [NPG]

Set $\eta = (1 - \gamma)^2 \log A$.

For the softmax policy class, we have after T iterations,

$$V^{(T)}(\rho) \geq V^*(\rho) - \frac{2}{(1 - \gamma)^2 T}$$

- Dimension free iteration complexity: (No dependence on S, A, μ)
- Also a “fast rate”.
- Even though problem is non-convex, a mirror descent analysis applies. Analysis idea from [Even-Dar, K., Mansour 2009]

Global Convergence of NPG

Theorem [NPG]

Set $\eta = (1 - \gamma)^2 \log A$.

For the softmax policy class, we have after T iterations,

$$V^{(T)}(\rho) \geq V^*(\rho) - \frac{2}{(1 - \gamma)^2 T}$$

- Dimension free iteration complexity: (No dependence on S, A, μ)
- Also a “fast rate”.
- Even though problem is non-convex, a mirror descent analysis applies. Analysis idea from [Even-Dar, K., Mansour 2009]

What about approximate/sampled gradients and large state space?

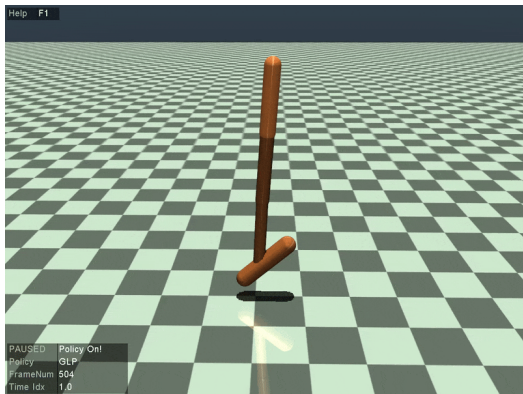
Taking stock: “measures” and related work

what is the role of the “coverage measure” μ ?

Brittle policies if we train only from one configuration!

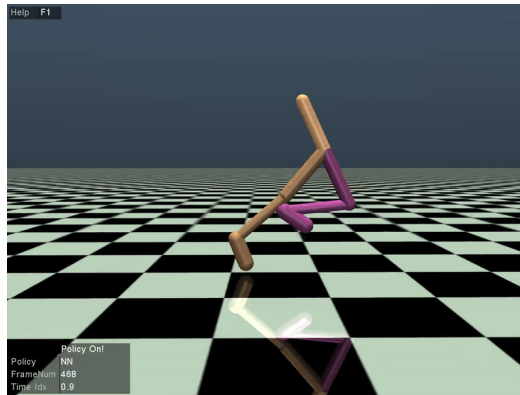
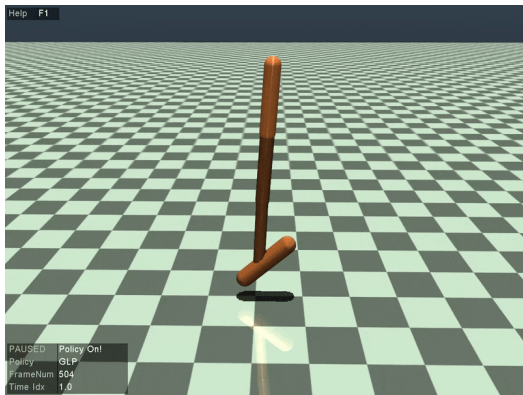
- [Rajeswaran, Lowrey, Todorov, K. 2017]: showed policies optimized for a single starting configuration s_0 are not robust!

Brittle policies if we train only from one configuration!



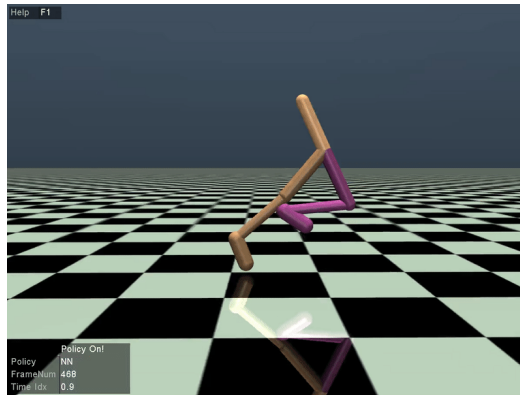
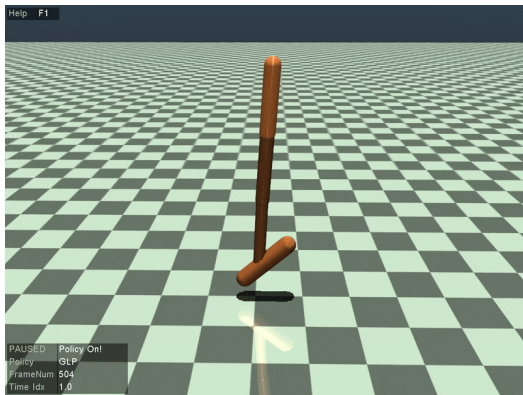
- [Rajeswaran, Lowrey, Todorov, K. 2017]: showed policies optimized for a single starting configuration s_0 are not robust!

Brittle policies if we train only from one configuration!



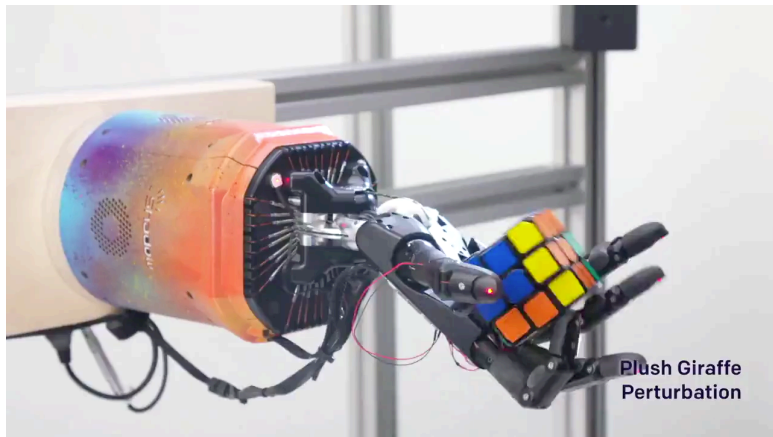
- [Rajeswaran, Lowrey, Todorov, K. 2017]: showed policies optimized for a single starting configuration s_0 are not robust!

Brittle policies if we train only from one configuration!



- [Rajeswaran, Lowrey, Todorov, K. 2017]: showed policies optimized for a single starting configuration s_0 are not robust!
- How to fix this?
Training from different starting configurations sampled from $s_0 \sim \mu$ fixes this.

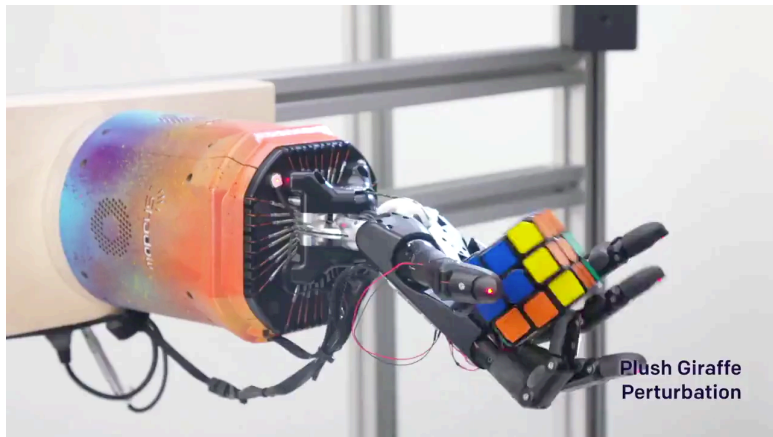
OpenAI: progress on dexterous hand manipulation



Trained with “domain randomization”

Basically, the measure $s_0 \sim \mu$ was diverse.

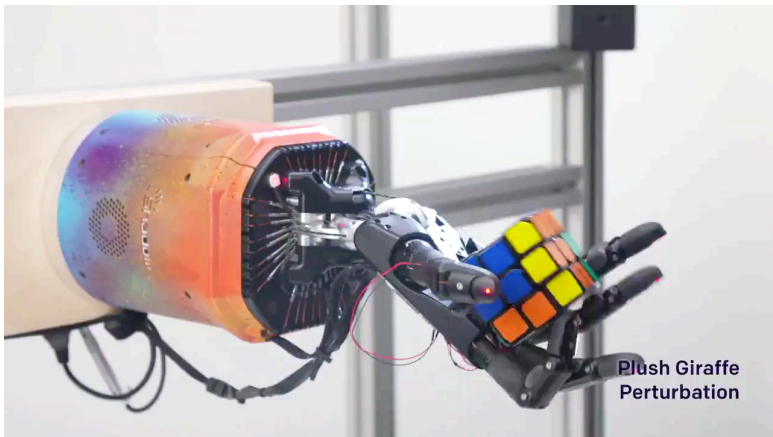
OpenAI: progress on dexterous hand manipulation



Trained with “domain randomization”

Basically, the measure $s_0 \sim \mu$ was diverse.

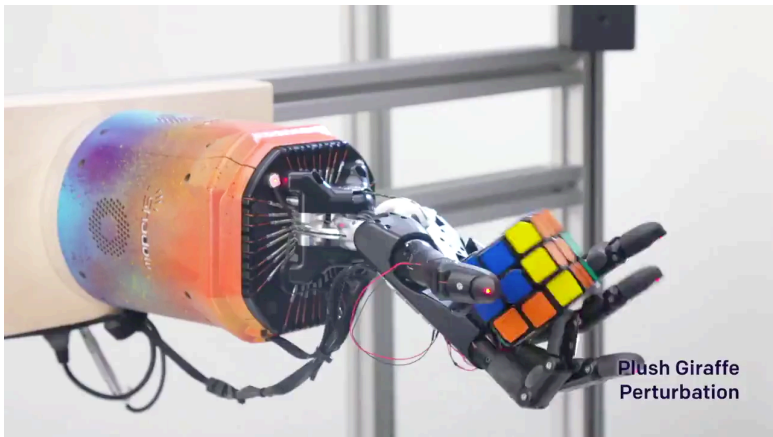
OpenAI: progress on dexterous hand manipulation



Trained with “domain randomization”

Basically, the measure $s_0 \sim \mu$ was diverse.

OpenAI: progress on dexterous hand manipulation



Trained with “domain randomization”

Basically, the measure $s_0 \sim \mu$ was diverse.

- How should we think about **approximation/generalization**?
(this is **not an issue in supervised learning**)
- How should we think about the measure μ in the infinite state space case?
(μ lets us **sidestep exploration...**)

Related Work: optimization and generalization

Generalization:

Related Work:

optimization and generalization

Generalization:

- approx. dynamic programming requires worst-case ℓ_∞ guarantees on errors.
some relaxations possible: [Munos, 2005, Antos et al., 2008]

Related Work:

optimization and generalization

Generalization:

- approx. dynamic programming requires worst-case ℓ_∞ guarantees on errors.
some relaxations possible: [Munos, 2005, Antos et al., 2008]
- [K. & Langford; '02] conservative policy iteration (CPI)
provable guarantees in terms of 'supervised learning' error + μ
 - Related: PSDP [Bagnell et al, '04], [Scherer & Geist, '14], MD-MPI [Geist et al., '19]...
 - imitation learning

Related Work:

optimization and generalization

Generalization:

- approx. dynamic programming requires worst-case ℓ_∞ guarantees on errors.
some relaxations possible: [Munos, 2005, Antos et al., 2008]
- [K. & Langford; '02] conservative policy iteration (CPI)
provable guarantees in terms of 'supervised learning' error + μ
 - Related: PSDP [Bagnell et al, '04], [Scherer & Geist, '14], MD-MPI [Geist et al., '19]...
 - imitation learning

Optimization and global convergence:

Related Work:

optimization and generalization

Generalization:

- approx. dynamic programming requires worst-case ℓ_∞ guarantees on errors.
some relaxations possible: [Munos, 2005, Antos et al., 2008]
- [K. & Langford; '02] conservative policy iteration (CPI)
provable guarantees in terms of 'supervised learning' error + μ
 - Related: PSDP [Bagnell et al, '04], [Scherer & Geist, '14], MD-MPI [Geist et al., '19]...
 - imitation learning

Optimization and global convergence:

- roots in [Even-Dar, K., Mansour 2009]

Related Work:

optimization and generalization

Generalization:

- approx. dynamic programming requires worst-case ℓ_∞ guarantees on errors.
some relaxations possible: [Munos, 2005, Antos et al., 2008]
- [K. & Langford; '02] conservative policy iteration (CPI)
provable guarantees in terms of 'supervised learning' error + μ
 - Related: PSDP [Bagnell et al, '04], [Scherer & Geist, '14], MD-MPI [Geist et al., '19]...
 - imitation learning

Optimization and global convergence:

- roots in [Even-Dar, K., Mansour 2009]
- CPI also gives a subgradient condition for policy search
 - [Scherer & Geist, '14], [Bhandari & Russo]

Related Work:

optimization and generalization

Generalization:

- approx. dynamic programming requires worst-case ℓ_∞ guarantees on errors.
some relaxations possible: [Munos, 2005, Antos et al., 2008]
- [K. & Langford; '02] conservative policy iteration (CPI)
provable guarantees in terms of 'supervised learning' error + μ
 - Related: PSDP [Bagnell et al, '04], [Scherer & Geist, '14], MD-MPI [Geist et al., '19]...
 - imitation learning

Optimization and global convergence:

- roots in [Even-Dar, K., Mansour 2009]
- CPI also gives a subgradient condition for policy search
 - [Scherer & Geist, '14], [Bhandari & Russo]
- [Fazel et. al. 18]: global convergence for LQRs

Part-II: Large State Spaces

Approximation and Generalization

Policy Classes

$\pi_\theta(a \mid s)$ is the probability of action a given s , parameterized by

$$\pi_\theta(a \mid s) \propto \exp(f_\theta(s, a))$$

Policy Classes

$\pi_\theta(a \mid s)$ is the probability of action a given s , parameterized by

$$\pi_\theta(a \mid s) \propto \exp(f_\theta(s, a))$$

- Softmax policy class: $f_\theta(s, a) = \theta_{s,a}$

Policy Classes

$\pi_\theta(a \mid s)$ is the probability of action a given s , parameterized by

$$\pi_\theta(a \mid s) \propto \exp(f_\theta(s, a))$$

- Softmax policy class: $f_\theta(s, a) = \theta_{s,a}$
- Log-linear policy class: $f_\theta(s, a) = \vec{\theta} \cdot \vec{\phi}(s, a)$ where $\vec{\phi}(s, a) \in R^d$

Policy Classes

$\pi_\theta(a | s)$ is the probability of action a given s , parameterized by

$$\pi_\theta(a | s) \propto \exp(f_\theta(s, a))$$

- Softmax policy class: $f_\theta(s, a) = \theta_{s,a}$
- Log-linear policy class: $f_\theta(s, a) = \vec{\theta} \cdot \vec{\phi}(s, a)$ where $\vec{\phi}(s, a) \in R^d$
- Neural policy class: $f_\theta(s, a)$ is a neural network

NPG & Log Linear Policy Classes

- Feature vector $\phi(s, a) \in \mathbb{R}^d$, $\pi_\theta(a | s) \propto \exp(\theta \cdot \phi_{s,a})$

NPG & Log Linear Policy Classes

- Feature vector $\phi(s, a) \in \mathbb{R}^d$, $\pi_\theta(a | s) \propto \exp(\theta \cdot \phi_{s,a})$
- At iteration t , the NPG update rule:

$$\theta \leftarrow \theta + \eta F(\theta)^{-1} \nabla V^\theta(s_0)$$

is equivalent to the “soft”+approximate policy iteration update:

NPG & Log Linear Policy Classes

- Feature vector $\phi(s, a) \in \mathbb{R}^d$, $\pi_\theta(a | s) \propto \exp(\theta \cdot \phi_{s,a})$
- At iteration t , the NPG update rule:

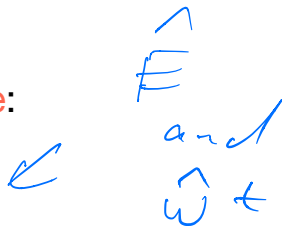
$$\theta \leftarrow \theta + \eta F(\theta)^{-1} \nabla V^\theta(s_0)$$

is equivalent to the “soft”+approximate policy iteration update:

1. approximate the Q^θ with the the features:

$$w_\star \in \operatorname{argmin}_w E_{s,a \sim d(\cdot | \pi, \mu)} \left[\left(Q^\theta(s, a) - w \cdot \phi_{s,a} \right)^2 \right].$$

where $d(\cdot | \pi, \mu)$ is “on-policy” distribution starting from $s_0, a_0 \sim \mu$



NPG & Log Linear Policy Classes

- Feature vector $\phi(s, a) \in \mathbb{R}^d$, $\pi_\theta(a | s) \propto \exp(\theta \cdot \phi_{s,a})$
- At iteration t , the NPG update rule:

$$\theta \leftarrow \theta + \eta \widehat{F}(\theta)^{-1} \nabla \widehat{V}^\theta(s_0)$$

is equivalent to the “soft”+approximate policy iteration update:

1. approximate the Q^θ with the the features:

$$w_\star \in \operatorname{argmin}_w E_{s,a \sim d(\cdot | \pi, \mu)} [(Q^\theta(s, a) - w \cdot \phi_{s,a})^2].$$

where $d(\cdot | \pi, \mu)$ is “on-policy” distribution starting from $s_0, a_0 \sim \mu$

2. policy update

$$\pi(a | s) \leftarrow \frac{\pi(a | s) \exp(w_\star \cdot \phi_{s,a})}{Z_s}$$

(Z_s is the normalizing constant)

\hat{E}, \hat{w}^t

use samples

\hat{Q}

Realizable case: NPG + log linear policy classes

Realizable case: NPG + log linear policy classes

- **Realizability:** Suppose that $Q^\theta(s, a)$ is a linear function in $\phi(s, a)$

Realizable case: NPG + log linear policy classes

- **Realizability:** Suppose that $Q^\theta(s, a)$ is a linear function in $\phi(s, a)$
- **Supervised learning error:** our estimate \widehat{w}^t has bounded regression error (say due to sampling)

$$E_{s,a \sim d(\cdot|\pi,\mu)} \left[\left(Q^\theta(s, a) - \widehat{w}^t \cdot \phi_{s,a} \right)^2 \right] \leq \epsilon_{\text{stat}} \approx \frac{1}{\sqrt{n}}$$

Realizable case: NPG + log linear policy classes

- **Realizability:** Suppose that $Q^\theta(s, a)$ is a linear function in $\phi(s, a)$
- **Supervised learning error:** our estimate \widehat{w}^t has bounded regression error (say due to sampling)

$$E_{s,a \sim d(\cdot|\pi,\mu)} \left[\left(Q^\theta(s, a) - \widehat{w}^t \cdot \phi_{s,a} \right)^2 \right] \leq \epsilon_{\text{stat}}$$

- **Conditioning (i.e. “feature coverage”):** $\|\phi_{s,a}\| \leq 1$ and define

$$\kappa = 1/\sigma_{\min} \left(E_{s,a \sim \mu} [\phi_{s,a} \phi_{s,a}^\top] \right)$$

Realizable case: NPG + log linear policy classes

- **Realizability:** Suppose that $Q^\theta(s, a)$ is a linear function in $\phi(s, a)$
- **Supervised learning error:** our estimate \widehat{w}^t has bounded regression error (say due to sampling)

$$E_{s,a \sim d(\cdot|\pi, \mu)} \left[(Q^\theta(s, a) - \widehat{w}^t \cdot \phi_{s,a})^2 \right] \leq \epsilon_{\text{stat}} \approx \frac{1}{\sqrt{N}}$$

- **Conditioning (i.e. “feature coverage”):** $\|\phi_{s,a}\| \leq 1$ and define

$$\kappa = 1/\sigma_{\min} \left(E_{s,a \sim \mu} [\phi_{s,a} \phi_{s,a}^\top] \right)$$

Theorem [NPG]

A : #actions, H : Horizon = $1/(1 - \gamma)$, Norm bound: $\|\widehat{w}^t\| \leq W$

After T iterations, the NPG algorithm returns a π s.t.

$$V^{(T)}(\rho) \geq V^*(\rho) - \underbrace{HW \sqrt{\frac{2 \log A}{T}}}_{\text{opt. error.}} + \underbrace{\sqrt{4AH^3 \kappa \epsilon_{\text{stat}}}}_{\text{stat. error.}}$$

opt. error. stat. error.

NPG+Log Linear Case

(just notation for sample based approach)

NPG+Log Linear Case

(just notation for sample based approach)

- For a state-action distribution v , define:

$$L(w; \theta, v) := E_{s,a \sim v} \left[(Q^{\pi_\theta}(s, a) - w \cdot \phi_{s,a})^2 \right].$$

NPG+Log Linear Case

(just notation for sample based approach)

- For a state-action distribution v , define:

$$L(w; \theta, v) := E_{s,a \sim v} [(Q^{\pi_\theta}(s, a) - w \cdot \phi_{s,a})^2].$$

- The NPG update is equivalent to:

NPG+Log Linear Case

(just notation for sample based approach)

- For a state-action distribution v , define:

$$L(w; \theta, v) := E_{s,a \sim v} [(Q^{\pi_\theta}(s, a) - w \cdot \phi_{s,a})^2].$$

- The NPG update is equivalent to:

1. approximate the Q^θ with the the features:

$$\widehat{w}^t \approx \operatorname{argmin}_w \widehat{L}(w; \theta, d(\cdot | \pi, \mu)).$$

where $d(\cdot | \pi, \mu)$ is “on-policy” distribution starting from $s_0, a_0 \sim \mu$

NPG+Log Linear Case

(just notation for sample based approach)

- For a state-action distribution v , define:

$$L(w; \theta, v) := E_{s,a \sim v} [(Q^{\pi_\theta}(s, a) - w \cdot \phi_{s,a})^2].$$

- The NPG update is equivalent to:

1. approximate the Q^θ with the the features:

$$\widehat{w}^t \approx \operatorname{argmin}_w L(w; \theta, d(\cdot | \pi, \mu)).$$

where $d(\cdot | \pi, \mu)$ is “on-policy” distribution starting from $s_0, a_0 \sim \mu$

2. policy update: $\pi(a | s) \leftarrow \pi(a | s) \exp(w_\star \cdot \phi_{s,a}) / Z_s$

(Z_s is the normalizing constant)

NPG: Conv. Rate with Approx+Est. Errors

NPG: Conv. Rate with Approx+Est. Errors

- **Supervised learning error:** Suppose the **excess risk** and **approx error** are bounded as:

$$L(w^{(t)}; \theta^{(t)}, d^{(t)}) - L(w_{\star}^{(t)}; \theta^{(t)}, d^{(t)}) \leq \epsilon_{\text{stat}},$$

$$L(w_{\star}^{(t)}; \theta^{(t)}, d^{(t)}) \leq \epsilon_{\text{approx}},$$

NPG: Conv. Rate with Approx+Est. Errors

- Supervised learning error: Suppose the excess risk and approx error are bounded as:

$$L(w^{(t)}; \theta^{(t)}, d^{(t)}) - L(w_{\star}^{(t)}; \theta^{(t)}, d^{(t)}) \leq \epsilon_{\text{stat}},$$

$$L(w_{\star}^{(t)}; \theta^{(t)}, d^{(t)}) \leq \epsilon_{\text{approx}},$$

← under fit. at iteration t.

- Conditioning (i.e. “feature coverage”): $\|\phi_{s,a}\| \leq 1$ and define

$$\kappa = 1/\sigma_{\min}(E_{s,a \sim \mu}[\phi_{s,a} \phi_{s,a}^{\top}])$$

NPG: Conv. Rate with Approx+Est. Errors

- Supervised learning error: Suppose the excess risk and approx error are bounded as:

$$L(w^{(t)}; \theta^{(t)}, d^{(t)}) - L(w_{\star}^{(t)}; \theta^{(t)}, d^{(t)}) \leq \epsilon_{\text{stat}},$$

$$L(w_{\star}^{(t)}; \theta^{(t)}, d^{(t)}) \leq \epsilon_{\text{approx}},$$

- Conditioning (i.e. “feature coverage”): $\|\phi_{s,a}\| \leq 1$ and define

$$\kappa = 1/\sigma_{\min}(E_{s,a \sim \mu}[\phi_{s,a} \phi_{s,a}^{\top}])$$

$$\epsilon_{\text{transfer}} = L(w_{\star}^{\star}; \theta^{\star}, d^{\star})$$

Theorem [NPG]

A : #actions, H : Horizon = $1/(1 - \gamma)$

After T iterations, the NPG algorithm returns a π s.t.

d^{\star} = state-action
dist. of any π^{\star}
comparator

$$V^{(T)}(s_0) \geq V^{\star}(s_0) - HW \sqrt{\frac{2 \log A}{T}} + \sqrt{4AH^3 \left(\kappa \cdot \epsilon_{\text{stat}} + \left\| \frac{d^{\star}}{\mu} \right\|_{\infty} \cdot \epsilon_{\text{approx}} \right)}$$

where $\left\| \frac{a}{b} \right\|_{\infty} = \max_i \left| \frac{a_i}{b_i} \right|$.

opt. error

stat. error

approx. error

Thank you!

- theory foundations of PG methods: optimization and approximation guarantees



Alekh Agarwal



Jason Lee



Gaurav Mahajan

Thank you!

- theory foundations of PG methods: optimization and approximation guarantees
 - PG methods effective due to their approximation power



Alekh Agarwal



Jason Lee



Gaurav Mahajan

Thank you!

- theory foundations of PG methods: optimization and approximation guarantees
 - PG methods effective due to their approximation power
- conceptually (and technically) important issues for progress:



Alekh Agarwal



Jason Lee



Gaurav Mahajan

Thank you!

- theory foundations of PG methods: optimization and approximation guarantees
 - PG methods effective due to their approximation power
- conceptually (and technically) important issues for progress:
 - **exploration**: we assumed a good coverage “ μ ” but this should be learned (see pc-pg paper!)



Alekh Agarwal



Jason Lee



Gaurav Mahajan

Thank you!

- theory foundations of PG methods: optimization and approximation guarantees
 - PG methods effective due to their approximation power
- conceptually (and technically) important issues for progress:
 - **exploration**: we assumed a good coverage “ μ ” but this should be learned (see pc-pg paper!)
 - **representation/transfer learning**: theory of RL is different from SL.



Alekh Agarwal



Jason Lee



Gaurav Mahajan