Advanced policy space methods

Managing the learning rate

Cathy Wu

6.7920: Reinforcement Learning: Foundations and Methods

Readings

1. Alekh Agarwal Nan Jiang Sham M. Kakade Wen Sun. Reinforcement Learning: Theory and Algorithms, 2021. (*Ch 11.1-11.2, Ch 12*)

Outline

1. Recap: policy gradient and actor-critic
2. Conservative policy iteration (CPI)
   a. Performance difference lemma
3. Natural policy gradient (NPG)
4. Trust region policy optimization (TRPO)
5. Proximal policy optimization (PPO)

Outline

1. **Recap: policy gradient and actor-critic**
2. Conservative policy iteration (CPI)
    a. Performance difference lemma
3. Natural policy gradient (NPG)
4. Trust region policy optimization (TRPO)
5. Proximal policy optimization (PPO)

Policy Gradient as Policy Update

Approximate Policy Iteration

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

Unstable (fast)

Policy Gradient

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

Smooth, fine control (slow)

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

A variety of approaches help to reduce variance:
temporal structure, baselines, actor-critic methods.

Policy-based vs value-based methods

update: Increase probability of selecting actions w/ higher returns, i.e. $\sum_{i=1}^{H} r_i$

Policy $\pi(\cdot\,|s)$

sample

max w.p. $1-\epsilon$

Action-value function $Q(s,\cdot)$

update

random w.p. ϵ

Action

Bootstrap target

e.g. $r + \gamma \max_{a'} Q(s', a')$

state, reward

state, reward

Environment

Policy-based methods

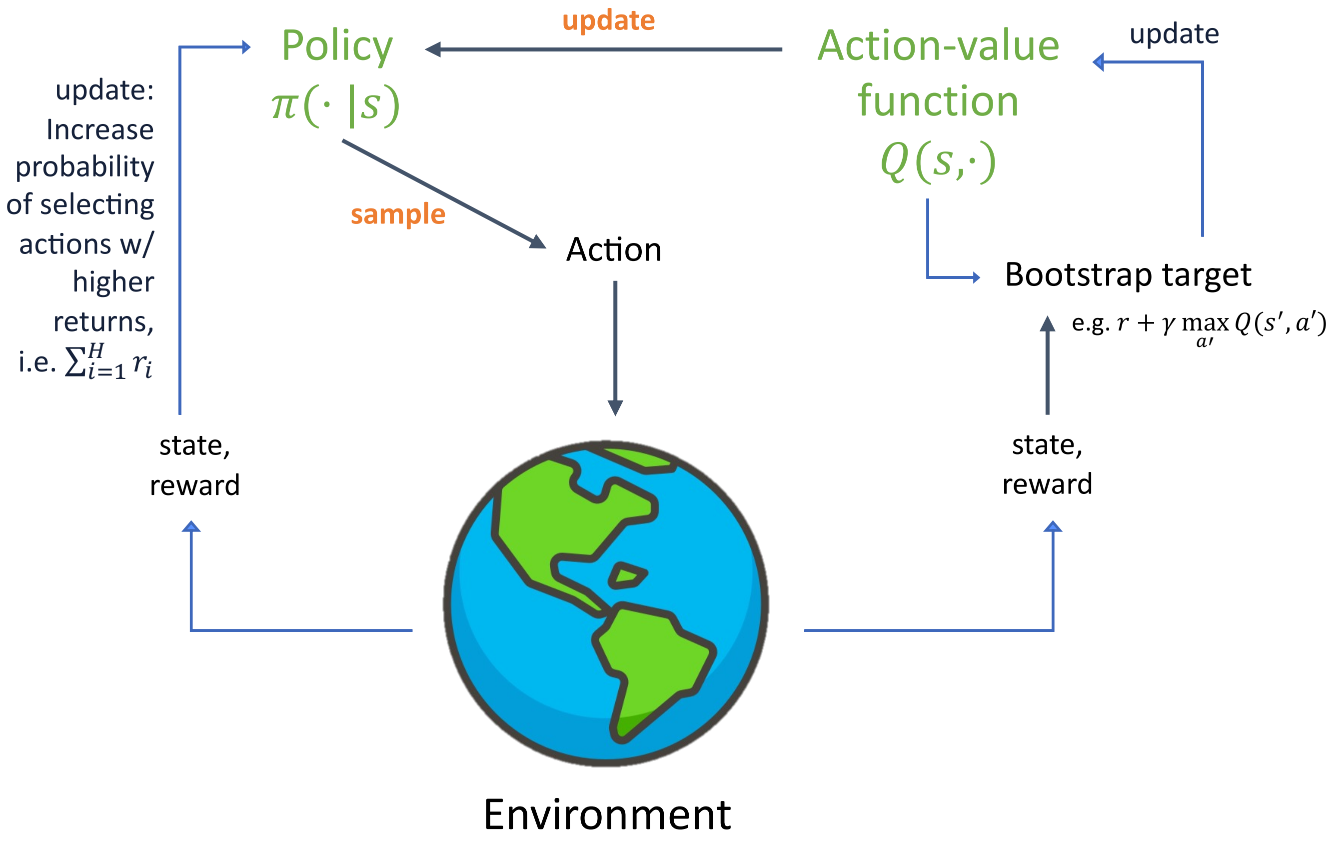
- Sensible & simple ☺
- Local convergence guarantees only
- Also “works” for partial observation, nonstationary settings
- Doesn’t work that well (on its own) ☹

Value-based methods

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

Wu

Actor-critic methods



Policy Gradient as Policy Update

Approximate Policy Iteration

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

Unstable (fast)

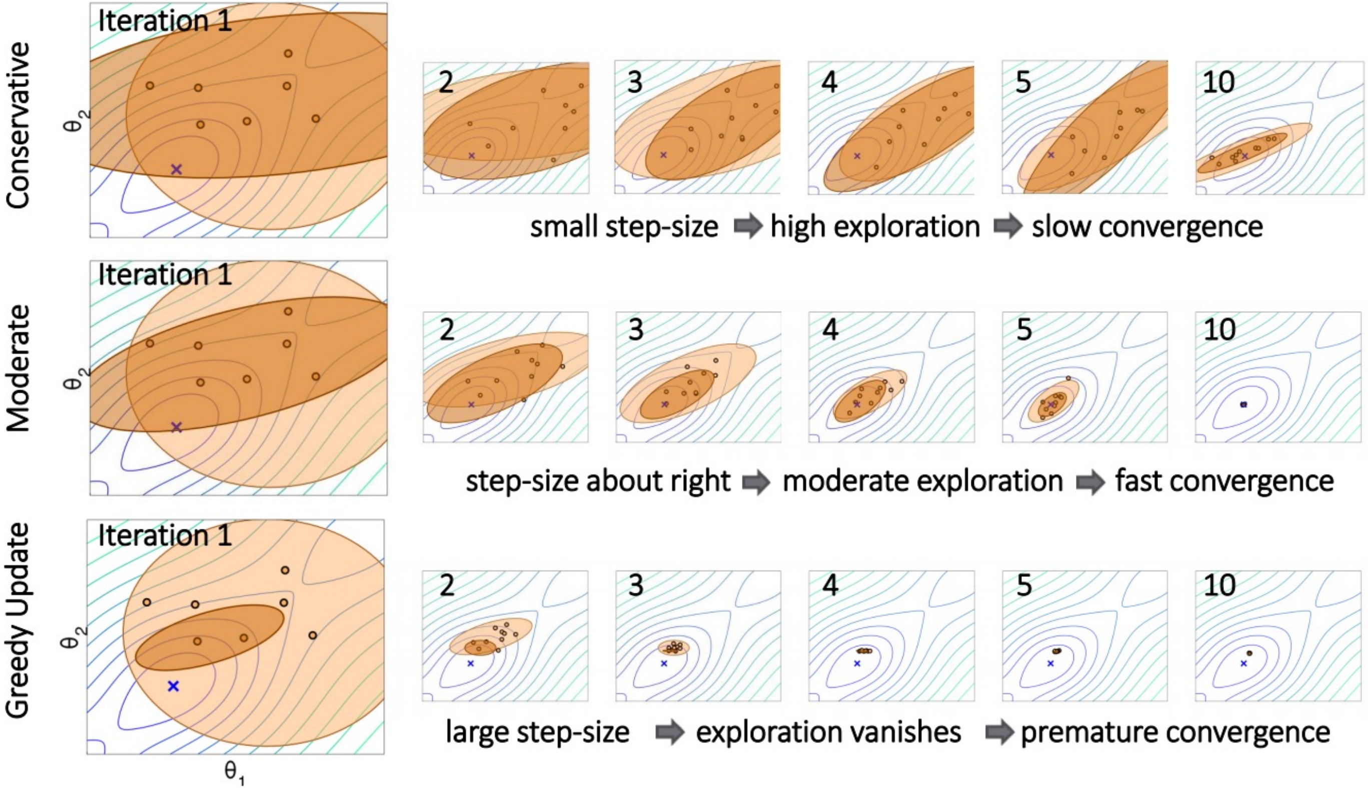
Policy Gradient

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

Smooth, fine control (slow)

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

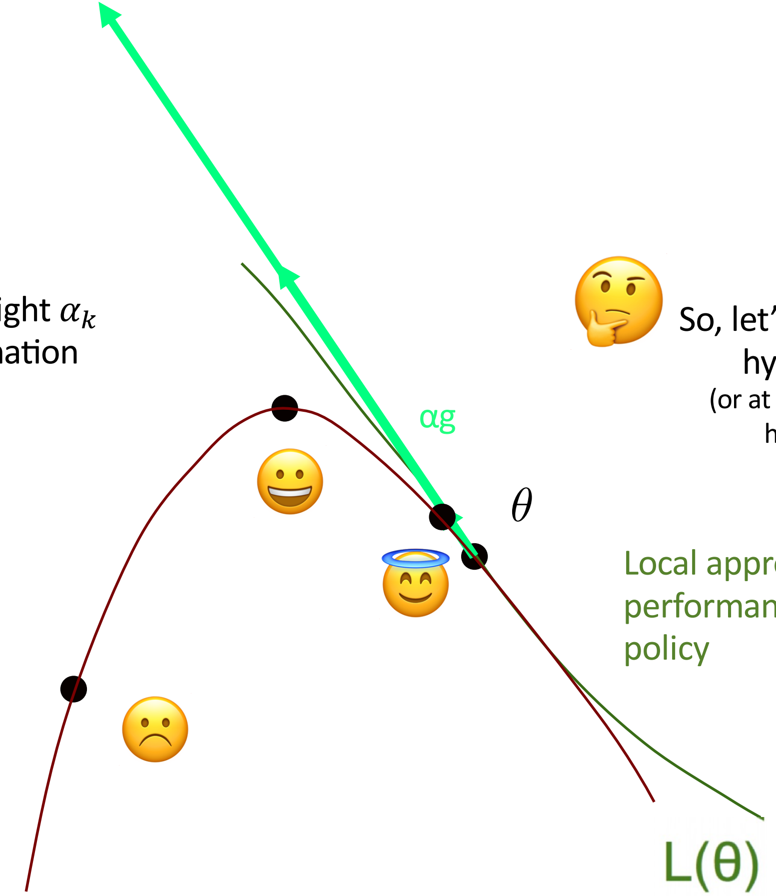
Exploration-exploitation trade-off



Source: Policy Search: Methods and Applications, Peters and Neumann

Policy gradient

Difficult to pick the right α_k without more information



So, let's avoid having this hyperparameter
(or at least find an "easier" hyperparameter)

Local approximation to performance of the policy

True objective (the performance of the policy)

V(θ)

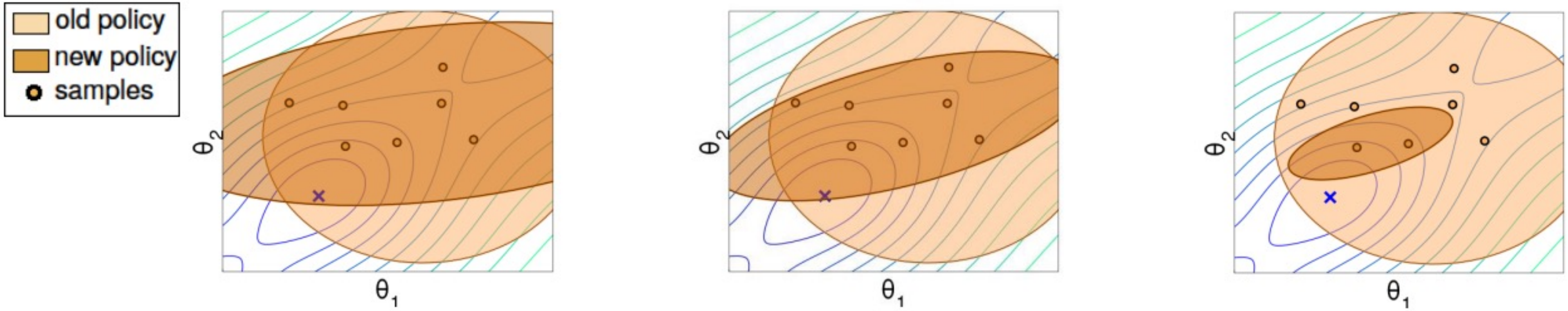
→ A slew of algorithms

Adapted from Matteo Pirotta

Wu

Desired Properties for the Policy Update

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



Conservative Update
Small "step size"

Moderate Update,
Moderate "step size"

Greedy update
Large "step size"

Outline

1. Recap: policy gradient and actor-critic
2. **Conservative policy iteration (CPI)**
   a. Performance difference lemma
3. Natural policy gradient (NPG)
4. Trust region policy optimization (TRPO)
5. Proximal policy optimization (PPO)

Relative Performance

Issues:

- We would like to exploit past samples (collected by π)
- We do not know how much to trust them (since $\pi' \neq \pi$)
- Depends on 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 $\pi, \pi' \in \Pi^{\mathrm{SR}}$

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

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

Proof: See Recitation.

Optimization Step

Let π be current policy, π' be a candidate next policy.

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

👍: Can maintain an estimate of $A^{\pi}(s,a)$.

Issue: Still cannot be directly estimated using data collected from π.

Kakade and Langford. Approximately Optimal Approximate Reinforcement Learning, ICML 2002.

Optimization Step

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

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

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

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

*Kakade and Langford. Approximately Optimal Approximate Reinforcement Learning, ICML 2002.

Surrogate Loss

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

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

also with this

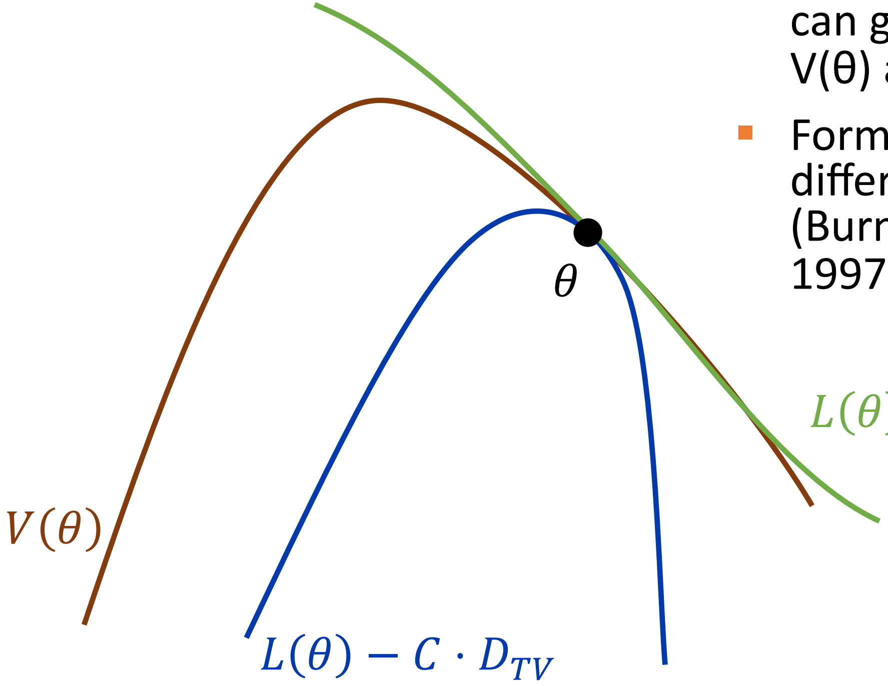
! In an interval close to π, L_π is a good surrogate for V

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

Kakade and Langford. Approximately Optimal Approximate Reinforcement Learning, ICML 2002.

Surrogate Loss (Continued)

- Key idea: if θ and θ’ close, can guarantee how close V(θ) and V(θ’) are
- Formally: performance-difference lemma (Burnetas and Katehakis, 1997)



True Objective (the performance of the policy)

Local approximation to performance of the policy

Pessimistic approximation to true objective (a lower bound we can maximize)

(Total variation (TV) distance)

Conservative Policy Iteration

- New policy improvement schema
  - Given current policy π_k solve:

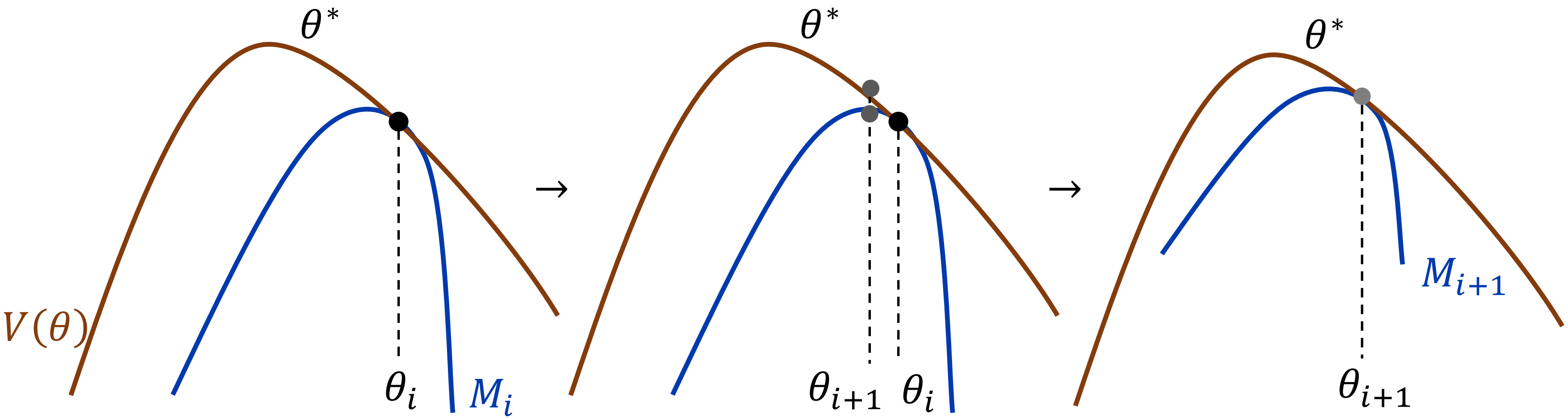
$$\boldsymbol{V(\pi')} - \boldsymbol{V(\pi_k)} \geq \max_{\pi'} \left\{ L_{\pi_k}(\pi') - C\ \mathbb{E}_{s\sim d^{\pi_k}}[D_{TV}(\pi'||\pi_k)[s]] \right\} \geq \boldsymbol{0}$$

$\Longrightarrow$ 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

Idea

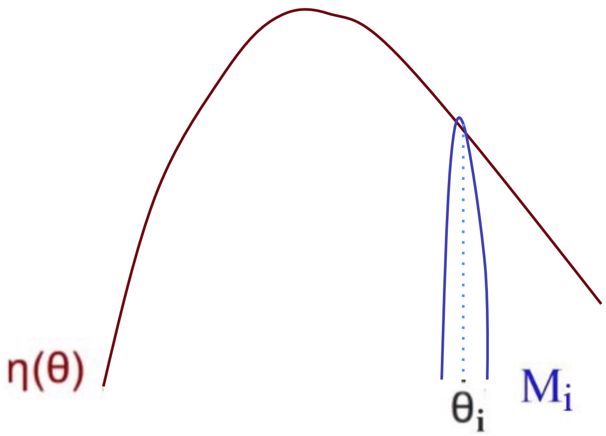


$V(\theta) = \mathbb{E}[\sum_{t=0}^{\infty} r_t | \pi_\theta]$ and M is the lower bound.

Source: Jonathan Hui: RL-The Math behind TRPO & PPO

Idea: Conservative policy iteration

Problem 1: too conservative



Problem 2: total variance (TV) distance is hard to optimize

→ Kullback–Leibler (KL) divergence
a.k.a. relative entropy

Kullback-Leibler divergence (relative entropy)

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

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

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

Toward Practical Algorithm

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

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

Also: not a gradient method anymore?

😕

New hyperparameter

😕

Further Steps Towards Practical Algorithms

- Importance weighting

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

$\Longrightarrow$ Natural Policy Gradient (NPG) [Kakade, 2002]

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

Outline

Parameter space vs distribution space

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

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

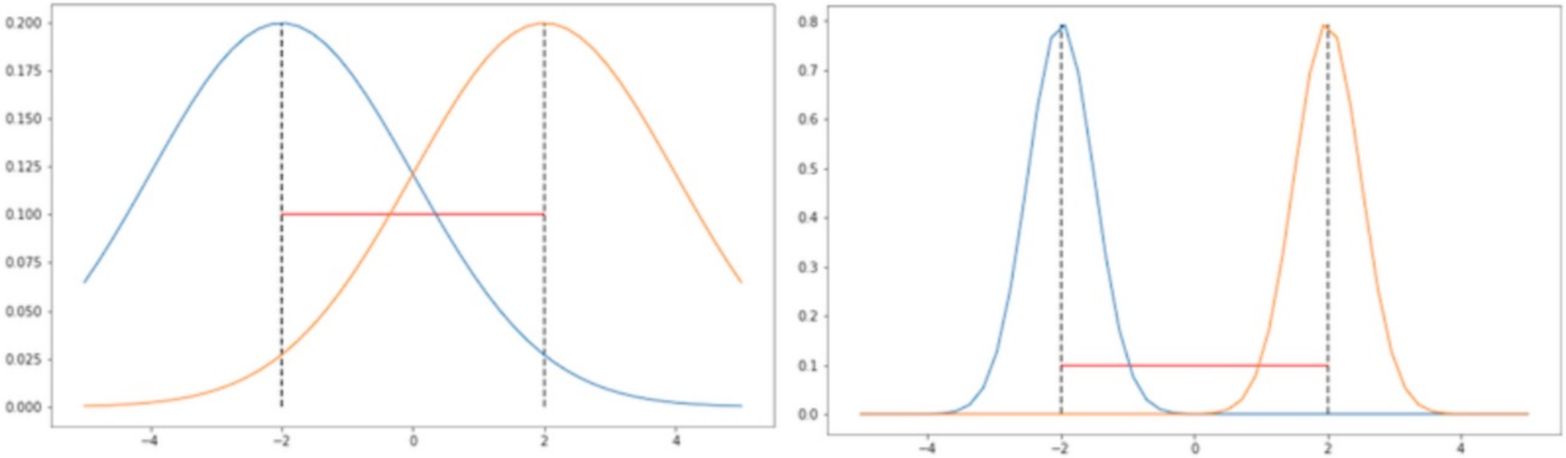
$$\frac{-\nabla h}{\|\nabla h\|} = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \arg \min_{d:\, \|d\| \leq \epsilon} \{h(\theta + d)\}$$

Is the Euclidean norm the best metric?

- Recall: θ induces stochastic policy
- → we are interested in optimizing in distribution space.

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)

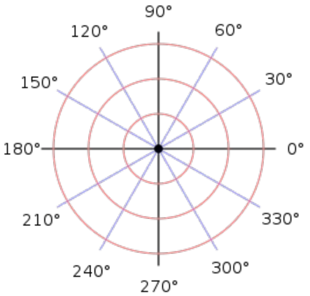
https://wiseodd.github.io/techblog/2018/03/14/natural-gradient/

Natural gradient

- Can we use an alternative definition of (local) distance?
- 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! Distribution space!
- Choose: KL divergence as the "metric"

Natural gradient

- A Riemannian space generalizes Euclidean space to curved spaces. In Riemannian space, the distance is defined as
$$d^2(v, v+\delta v) = \delta v^T G(v) \delta v$$
where G is the metric tensor



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

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

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

$$(\delta x \quad \delta y) \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{G} \begin{pmatrix} \delta x \\ \delta y \end{pmatrix} \qquad (\delta r \quad \delta\theta) \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & r \end{pmatrix}}_{G} \begin{pmatrix} \delta r \\ \delta\theta \end{pmatrix}$$

Natural Gradient

- In Riemannian space, the distance is defined as
$$d^2(v, v + \delta v) = \delta v^T G(v) \delta v^T$$
where G is the metric tensor

Natural Gradient [Amari, 1998]

The steepest descent in a Riemannian space is given by
$$\widetilde{\nabla} h(\theta) = G(\theta)^{-1} \nabla h(\theta)$$

- What is the metric tensor? Known for many objectives!
- Example: KL divergence (metric) → Fisher information (metric tensor)

KL Divergences and the Fisher Information Matrix

- The Kullback Leibler divergence can be approximated by the Fisher information matrix (2nd order Taylor approximation)

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

where $F(\theta)$ is the Fisher Information Matrix (FIM)

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

- Captures information on how a parameter influences the distribution

Natural Policy Gradient

$$\pi_{k+1} = \arg\max_{\pi'} \underbrace{\mathbb{E}_{s\sim d^{\pi}}\mathbb{E}_{a\sim\pi}\left[\frac{\pi'(s,a)}{\pi(s,a)}Q^{\pi}(s,a)\right]}_{:=\mathcal{L}_{\pi_k}(\pi')}$$

$$\text{s.t. } \underbrace{\mathbb{E}_{s\sim d^{\pi}}[D_{KL}(\pi'\|\pi)]}_{:=\bar{D}_{KL}(\pi'\|\pi)} \leq \delta$$

How to solve it? Do it approximately:

$$\mathcal{L}_{\theta_k}(\theta) \approx \mathcal{L}_{\theta_k}(\theta_k) + g^T(\theta-\theta_k)$$

$$\bar{D}_{KL}(\theta\|\theta_k) \approx \frac{1}{2}(\theta-\theta_k)^T F(\theta)(\theta-\theta_k)$$

where $g = \nabla_\theta \mathcal{L}_{\theta_k}(\theta)$ and $F(\theta) := \nabla_\theta^2 \bar{D}_{KL}(\theta\|\theta_k)$ is the FIM.

Natural Policy Gradient

The approximate problem is thus:

$$\theta_{k+1} = \arg\max_{\theta} g^T(\theta - \theta_k)$$
$$\text{s.t. } \frac{1}{2}(\theta - \theta_k)^T F(\theta - \theta_k) \leq \delta$$

whose solution is given by:

$$\theta_{k+1} = \theta_k + \underbrace{\sqrt{\frac{2\delta}{g^T F^{-1} g}}}_{\text{Step size}} \quad \underbrace{F^{-1} g}_{\text{Natural gradient}}$$

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

Natural policy gradient

$$\max_{\theta'} V_\theta(\theta')$$

$$\text{s.t.} \quad \boxed{\mathbb{E}_{s \sim d^\pi}\left[D_{KL}(\theta' || \theta)\right] \leq \delta}$$

Still hard to optimize

Also: not a gradient method anymore?

😕

Can approximate KL with Fisher information matrix

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

Captures how much a parameter influences the distribution

$$\theta_{k+1} = \theta_k + \underbrace{\sqrt{\frac{2\delta}{g^\top F^{-1} g}}}_{\text{step size}} \underbrace{F^{-1} g}_{\text{natural gradient}}$$

vs vanilla policy gradient:
Ignores how much parameters influence the distribution

How much policy changes

How much parameters change

Kakade, 2002; Peters and Schaal, 2008

Adapted from Matteo Pirotta

Wu

Natural Policy Gradient

Initialize policy parameter θ_0

for $\mathrm{k} = 1,2, \dots$ **do**

Collect trajectories $\mathcal{D}_k$ using policy $\pi_k = \pi(\theta_k)$

Estimate advantage function using any algorithm

Compute

- Policy gradient $\hat{g}_k$ (using advantage estimate)
- KL-divergence Hessian / Fisher information matrix $\hat{F}_k$

Compute new policy using natural gradient

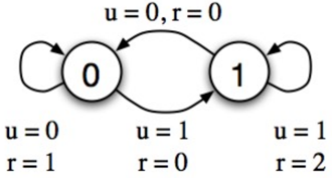
$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{\hat{g}_k^T \hat{F}_k^{-1} \hat{g}_k}} \hat{F}_k^{-1} \hat{g}_k$$

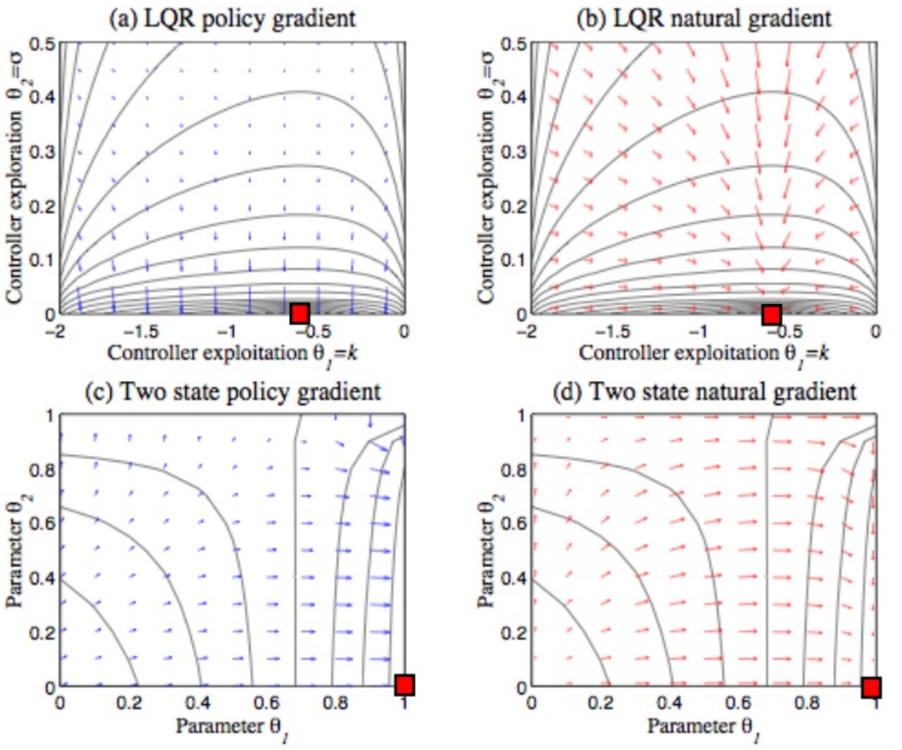
end

Linear Quadratic Regulation

$$x_{t+1} = Ax_t + Bu_t$$
$$u_t \sim \pi(u|x_t) = \mathcal{N}(u|kx_t, \sigma)$$
$$r_t = -x_t^T Q x_t - u_t^T R u_t$$

Two-State Problem





[Peters et al. 2003, 2005]

The standard gradient reduces the exploration too quickly!

Source: Policy Search: Methods and Applications, Peters and Neumann

Natural actor-critic (2008)

- NPG + refinements + compatible function approximation + imitation learning initialization

[Real-time online learning for robot control]

Outline

1. Recap: policy gradient and actor-critic
2. Conservative policy iteration (CPI)
   a. Performance difference lemma
3. Natural policy gradient (NPG)
4. **Trust region policy optimization (TRPO)**
5. Proximal policy optimization (PPO)

What's the problem now?

d could be in the millions

$$\theta_{k+1} = \theta_k + \underbrace{\sqrt{\frac{2\delta}{g^\intercal F^{-1} g}}}_{\text{step size}} \underbrace{F^{-1} g}_{\text{natural gradient}}$$

Problem:
Matrix inversion is $\mathcal{O}(d^3)$

Use iterative solution approach
(conjugate gradient method)

Called truncated natural policy gradient (TNPG)

Pascanu, Razvan, and Yoshua Bengio. "Revisiting natural gradient for deep networks." ICLR, 2014.

Problem: Might not improve V(θ) due to KL approximation (of TV distance)

Problem: Due to approximation, KL-constraint might be violated

Solution: enforce KL constraint using an *adaptive step size*

i.e. Try several step sizes and pick one that gives improvement & not too far

- Can always select a smaller step size to get improvement
- Don't want to update too slowly

Called trust region policy optimization (TRPO)

Schulman, John, et al. "Trust region policy optimization." ICML, 2015.

How much to change?
How to control how much of a change?

Trust Region Policy Optimization

How?

Backtracking line search with exponential decay (decay coeff. $\alpha \in (0,1)$, budget L)

Compute NPG step Δ_k

for j $= 0, \dots, L$ **do**

  Compute update $\theta = \theta_k + \alpha^j \Delta_k$

  if $\mathcal{L}_{\theta_k}(\theta) > 0$ and $\bar{D}_{KL}(\theta \| \theta_k) \leq \delta$ **then**

    Accept update and $\theta_{k+1} = \theta_k + \alpha^j \Delta_k$

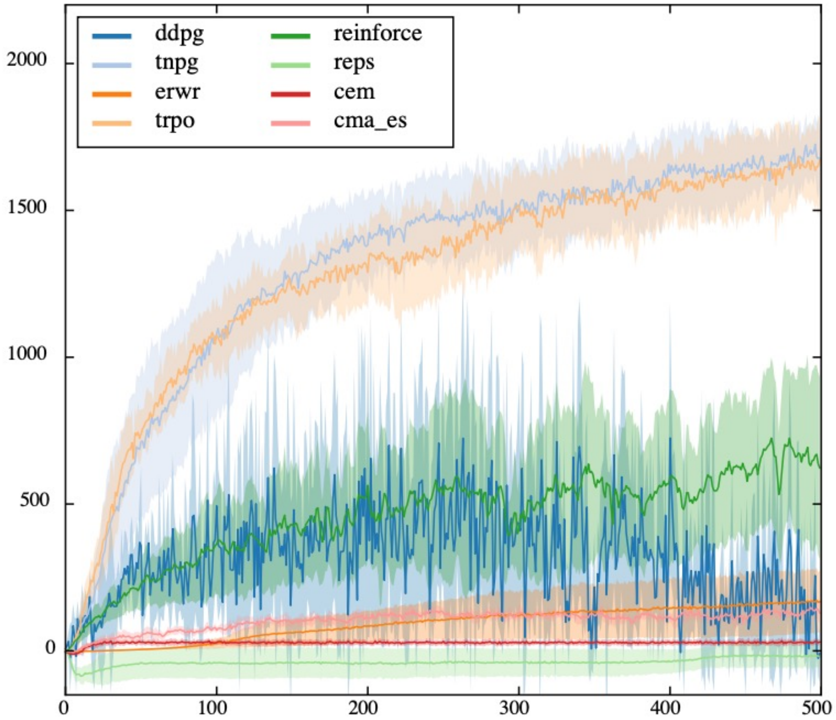
    break

  end

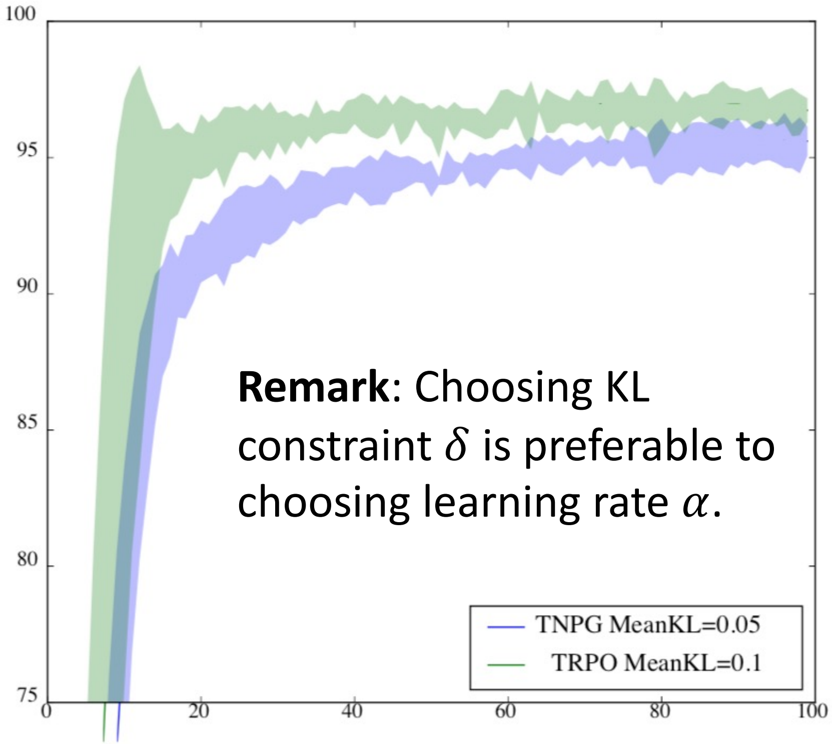
endfor

In practice, TRPO is implemented as (T)NPG plus line search.

Example: Continuous control [Duan et al., 2016]



Walker-2D



Remark: Choosing KL constraint δ is preferable to choosing learning rate α.

Swimmer

Outline

1. Recap: policy gradient and actor-critic
2. Conservative policy iteration (CPI)
   a. Performance difference lemma
3. Natural policy gradient (NPG)
4. Trust region policy optimization (TRPO)
5. **Proximal policy optimization (PPO)**

One more method:
proximal policy optimization [Schulman et al., 2017]

$$\theta_{k+1} = \theta_k + \underbrace{\sqrt{\frac{2\delta}{g^\top F^{-1} g}}}_{\text{step size}} \underbrace{F^{-1} g}_{\text{natural gradient}}$$

+ iterative approach to computing $F^{-1}g$

+ adaptive step sizes

Still pretty expensive to compute
Involves computing a Hessian $O(n^2)$

Adaptive step sizes worked well. Let's lean into it. Avoid natural gradient.

Regularize on KL term
Use adaptive step sizes to stay within δ

proximal policy optimization [Schulman et al., 2017]

Problem: Too many approximations going on. Still have high variance issues.

Key idea: Extra safety measure. Modify objective to ignore big changes.

What's a big change?

$$\pi'(s,a) \neq \pi(s,a)$$

One way: look at

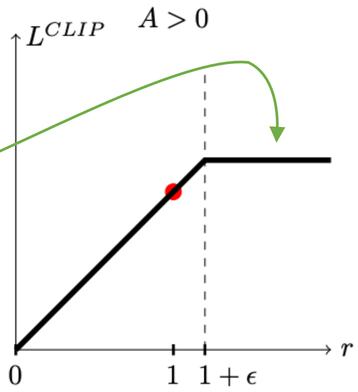
$$\frac{\pi'(s,a)}{\pi(s,a)}$$

Surrogate objective

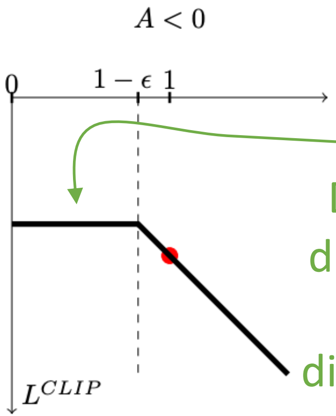
Importance weighting

Note: r is ratio, not reward

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



Data is noisy → Don't over-index on advantageous actions when the new policy is v. different from old policy



Data is noisy → Don't under-index on disadvantageous when the new policy is v. different from old policy

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

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

PPO with Clipping

Input: policy θ_0, clipping ϵ

for $\mathrm{k} = 1, \ldots$ **do**

Collect trajectories $\mathcal{D}_k$ using policy $\pi_k = \pi(\theta_k)$

Estimate advantage or Q-function using any algorithm

Compute:

$$\theta_{k+1} = \arg\max_{\theta} L_{\theta_k}^{CLIP}(\theta)$$

where:

$$L_{\pi}^{CLIP}(\pi') = \mathbb{E}_{\tau \sim \pi_k}\left[\sum_{t=1}^{T} \min\left\{r_t(\theta)\hat{A}_t^{\pi_k}, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t^{\pi_k}\right\}\right]$$

end

PPO with Adaptive KL Penalty

Input: policy θ_0, KL penalty β_0, KL-divergence δ

for $\mathrm{k} = 1, \ldots$ **do**

Collect trajectories $\mathcal{D}_k$ using policy $\pi_k = \pi(\theta_k)$

Estimate advantage or q-function using any algorithm

Compute by gradient descent:

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}(\theta) - \beta_k \bar{D}_{KL}(\theta \| \theta_k)$$

if $\bar{D}_{KL}(\theta_{k+1} \| \theta_k) \geq 1.5\delta$ **then**

$\beta_{k+1} = 2\beta_k$

end

if $\bar{D}_{KL}(\theta_{k+1} \| \theta_k) \leq \frac{\delta}{1.5}$ **then**

$\beta_{k+1} = \frac{\beta_k}{2}$

end

end

Initial β_0 not important. Some iteration may violate KL constraint, mostly not!

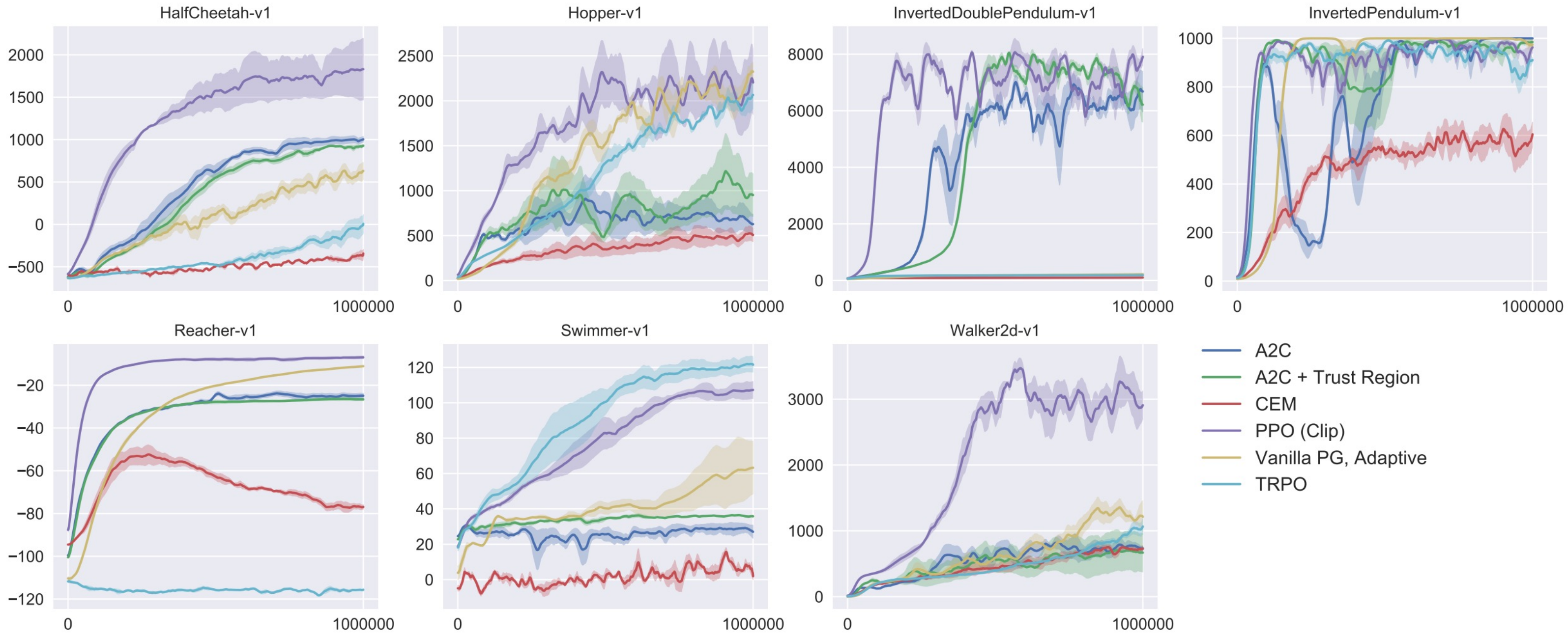


Figure 3: Comparison of several algorithms on several MuJoCo environments, training for one million timesteps.

Solving a Rubik’s cube with robot hand

Policy and value networks: 13 million parameters
Action space: 20 actuated joints (discretized with 11 bins) → $|A| = 11^{20}$

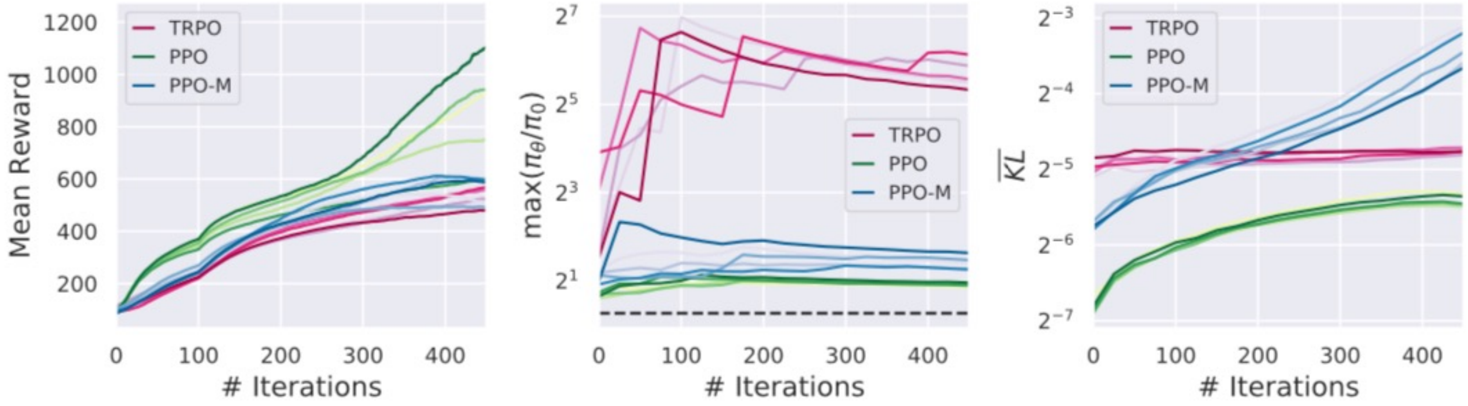
PPO + automatic domain randomization

OpenAI. Solving Rubik's Cube with a Robot Hand. (2019).

Implementation matters. What really makes PPO work?

See careful empirical studies:

- Engstrom, Logan, et al. "Implementation matters in deep policy gradients: A case study on PPO and TRPO." *arXiv preprint arXiv:2005.12729* (2020).
- Andrychowicz, Marcin, et al. "What matters in on-policy reinforcement learning? a large-scale empirical study." *arXiv preprint arXiv:2006.05990* (2020).



Humanoid

PPO vs PPO-M: value function clipping, reward scaling, orthogonal initialization & layer scaling, Adam learning rate annealing

Summary

- Policy gradient methods are an alternative and powerful class of reinforcement learning methods, based on directly optimizing the policy, rather than the value function.
- Due to the on-policy nature, data from π(θ) only says something about π(θ') if θ and θ' are close (performance difference lemma) → Need to be careful about the learning rate.
- A series of techniques to alleviate the burden of learning rate selection: relaxation to KL divergence, trust regions, natural gradients, iterative matrix inversion, adaptive step sizes, advantage clipping.
- Core practical policy gradient methods: REINFORCE, TRPO, PPO, SAC.
- Ultimately, computational efficiency is important when handling millions (or billions) of parameters.

References

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