Computer Engineering Department

Reinforcement Learning:
Advanced Policy Gradients

Mohammad Hossein Rohban, Ph.D.

Hosein Hasani

Spring 2023

Courtesy: Most of slides are adopted from CS 285 Berkeley.

Off-Policy Policy Gradients

$$\theta^\star = \arg\max_\theta J(\theta)$$

$$J(\theta) = E_{\tau\sim p_\theta(\tau)}[r(\tau)]$$

$$\nabla_\theta J(\theta) = E_{\tau\sim p_\theta(\tau)}[\nabla_\theta \log p_\theta(\tau) r(\tau)]$$

This is the problem!

- Neural networks change only a little bit with each gradient step
- On-policy learning can be extremely inefficient!

REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run it on the robot)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right)\left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i)\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Can't just skip this!

Policy Gradients Objective with Importance Sampling

$$\theta^\star = \arg\max_\theta J(\theta)$$

$$J(\theta) = E_{\tau\sim p_\theta(\tau)}[r(\tau)]$$

what if we don't have samples from $p_\theta(\tau)$?

(we have samples from some $\bar{p}(\tau)$ instead)

$$J(\theta) = E_{\tau\sim\bar{p}(\tau)}\left[\frac{p_\theta(\tau)}{\bar{p}(\tau)} r(\tau)\right]$$

importance weight

importance sampling

$$\begin{aligned} E_{x\sim p(x)}[f(x)] &= \int p(x)f(x)dx \\ &= \int \frac{q(x)}{q(x)} p(x) f(x) dx \\ &= \int q(x) \frac{p(x)}{q(x)} f(x) dx \\ &= E_{x\sim q(x)}\left[\frac{p(x)}{q(x)} f(x)\right] \end{aligned}$$

$$p_\theta(\tau) = p(\mathbf{s}_1)\prod_{t=1}^{T} \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

$$\frac{p_\theta(\tau)}{\bar{p}(\tau)} = \frac{\cancel{p(\mathbf{s}_1)}\prod_{t=1}^{T}\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\cancel{p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)}}{\cancel{p(\mathbf{s}_1)}\prod_{t=1}^{T}\bar{\pi}(\mathbf{a}_t|\mathbf{s}_t)\cancel{p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)}} = \frac{\prod_{t=1}^{T}\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}{\prod_{t=1}^{T}\bar{\pi}(\mathbf{a}_t|\mathbf{s}_t)}$$

Deriving the Policy Gradient with Importance Sampling

$$\theta^\star = \arg\max_\theta J(\theta)$$

$$J(\theta) = E_{\tau \sim p_\theta(\tau)}[r(\tau)]$$

a convenient identity

$$p_\theta(\tau)\nabla_\theta \log p_\theta(\tau) = \nabla_\theta p_\theta(\tau)$$

can we estimate the value of some *new* parameters θ'?

$$J(\theta') = E_{\tau \sim p_\theta(\tau)}\left[\frac{p_{\theta'}(\tau)}{p_\theta(\tau)} r(\tau)\right]$$

$$\nabla_{\theta'} J(\theta') = E_{\tau \sim p_\theta(\tau)}\left[\frac{\nabla_{\theta'} p_{\theta'}(\tau)}{p_\theta(\tau)} r(\tau)\right] = E_{\tau \sim p_\theta(\tau)}\left[\frac{p_{\theta'}(\tau)}{p_\theta(\tau)} \nabla_{\theta'} \log p_{\theta'}(\tau) r(\tau)\right]$$

now estimate locally, at $\theta = \theta'$: $\nabla_\theta J(\theta) = E_{\tau \sim p_\theta(\tau)}[\nabla_\theta \log p_\theta(\tau) r(\tau)]$

Off-Policy Policy Gradient

$$\theta^\star = \arg\max_\theta J(\theta) \qquad J(\theta) = E_{\tau\sim p_\theta(\tau)}[r(\tau)]$$

$$\nabla_{\theta'} J(\theta') = E_{\tau\sim p_\theta(\tau)}\left[\frac{p_{\theta'}(\tau)}{p_\theta(\tau)}\nabla_{\theta'}\log\pi_{\theta'}(\tau)r(\tau)\right] \quad \text{when } \theta \neq \theta'$$

$$\frac{p_{\theta'}(\tau)}{p_\theta(\tau)} = \frac{\prod_{t=1}^T \pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\prod_{t=1}^T \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}$$

$$= E_{\tau\sim p_\theta(\tau)}\left[\left(\prod_{t=1}^T \frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}\right)\left(\sum_{t=1}^T \nabla_{\theta'}\log\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)\right)\left(\sum_{t=1}^T r(\mathbf{s}_t,\mathbf{a}_t)\right)\right] \quad \text{what about causality?}$$

$$= E_{\tau\sim p_\theta(\tau)}\left[\sum_{t=1}^T \nabla_{\theta'}\log\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)\underbrace{\left(\prod_{t'=1}^t \frac{\pi_{\theta'}(\mathbf{a}_{t'}|\mathbf{s}_{t'})}{\pi_\theta(\mathbf{a}_{t'}|\mathbf{s}_{t'})}\right)}\left(\sum_{t'=t}^T r(\mathbf{s}_{t'},\mathbf{a}_{t'})\underbrace{\left(\prod_{t''=t}^{t'} \frac{\pi_{\theta'}(\mathbf{a}_{t''}|\mathbf{s}_{t''})}{\pi_\theta(\mathbf{a}_{t''}|\mathbf{s}_{t''})}\right)}\right)\right]$$

future actions don’t affect current weight

ignore this part!

Off-Policy Policy Gradient: First Order Approximation

$$\nabla_{\theta'} J(\theta') = E_{\tau \sim p_\theta(\tau)}\left[\sum_{t=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t) \underbrace{\left(\prod_{t'=1}^{t} \frac{\pi_{\theta'}(\mathbf{a}_{t'}|\mathbf{s}_{t'})}{\pi_\theta(\mathbf{a}_{t'}|\mathbf{s}_{t'})}\right)}_{\text{exponential in T}} \left(\sum_{t'=t}^{T} r(\mathbf{s}_{t'}, \mathbf{a}_{t'})\right)\right]$$

Approximation:

$$\nabla_{\theta'} J(\theta') \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \frac{\pi_{\theta'}(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})}{\pi_\theta(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})} \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \hat{Q}_{i,t} \qquad (\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \sim \pi_\theta(\mathbf{s}_t, \mathbf{a}_t)$$

$$= \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \underbrace{\cancel{\frac{\pi_{\theta'}(\mathbf{s}_{i,t})}{\pi_\theta(\mathbf{s}_{i,t})}}}_{\text{ignore this part!}} \frac{\pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}{\pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})} \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \hat{Q}_{i,t}$$

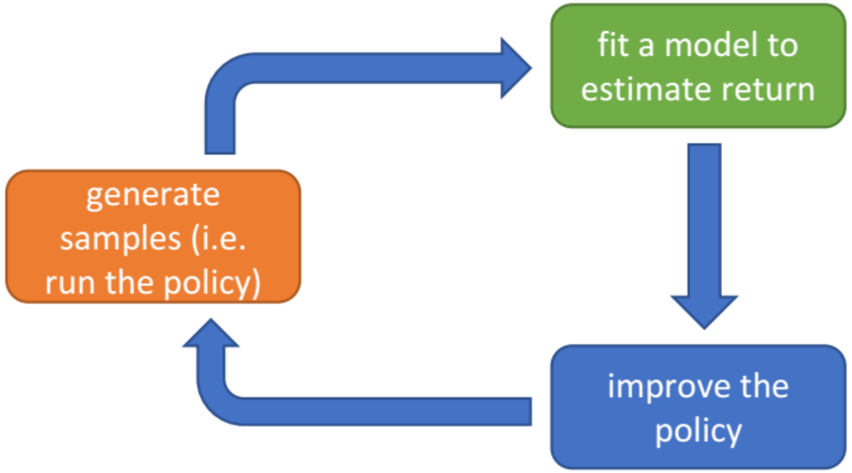
Recap: Policy Gradients

REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run the policy)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i) \left(\sum_{t'=t}^{T} r(\mathbf{s}_{t'}^i, \mathbf{a}_{t'}^i) \right) \right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \underbrace{\hat{Q}_{i,t}^{\pi}}_{\text{“reward to go”}}$$

$$\hat{Q}^\pi(\mathbf{x}_t, \mathbf{u}_t) = \sum_{t'=t}^{T} r(\mathbf{x}_{t'}, \mathbf{u}_{t'})$$



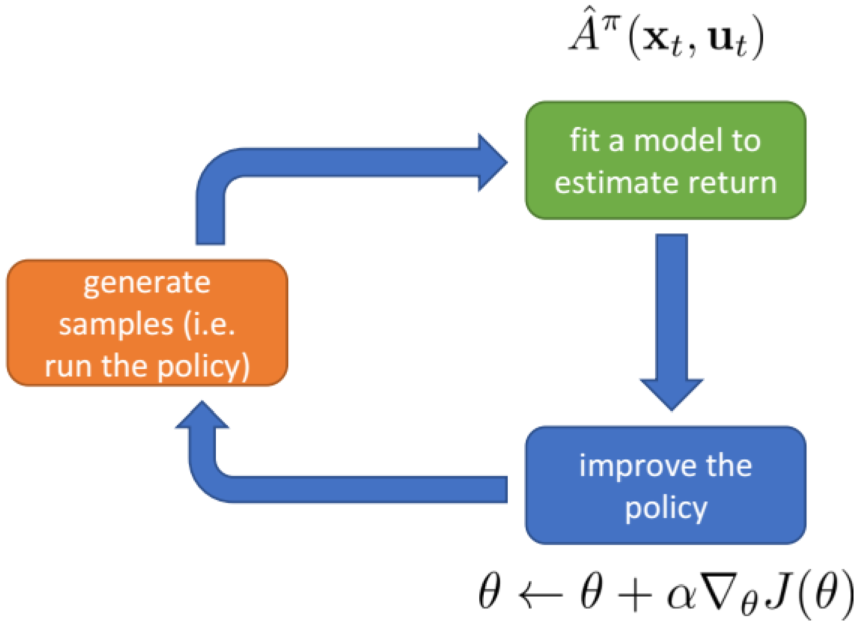
$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

Policy Gradient as Policy Iteration

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \hat{A}^\pi_{i,t}$$

main steps of policy gradient algorithm:

1. Estimate $\hat{A}^\pi(\mathbf{s}_t, \mathbf{a}_t)$ for current policy π
2. Use $\hat{A}^\pi(\mathbf{s}_t, \mathbf{a}_t)$ to get *improved* policy π'



Familiar to policy iteration algorithm:

1. evaluate $A^\pi(\mathbf{s}, \mathbf{a})$
2. set $\pi \leftarrow \pi'$

Policy Gradient as Policy Iteration

$$J(\theta) = E_{\tau \sim p_\theta(\tau)}\left[\sum_t \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)\right]$$

claim: $J(\theta') - J(\theta) = E_{\tau \sim p_{\theta'}(\tau)}\left[\sum_t \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)\right]$

could be interpret as policy improvement!

Policy Gradient as Policy Iteration

claim: $J(\theta') - J(\theta) = E_{\tau \sim p_{\theta'}(\tau)}\left[\sum_t \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)\right]$

proof:
$$J(\theta') - J(\theta) = J(\theta') - E_{\mathbf{s}_0 \sim p(\mathbf{s}_0)}\left[V^{\pi_\theta}(\mathbf{s}_0)\right]$$
$$= J(\theta') - E_{\tau \sim p_{\theta'}(\tau)}\left[V^{\pi_\theta}(\mathbf{s}_0)\right]$$
$$= J(\theta') - E_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t V^{\pi_\theta}(\mathbf{s}_t) - \sum_{t=1}^{\infty} \gamma^t V^{\pi_\theta}(\mathbf{s}_t)\right]$$
$$= J(\theta') + E_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t (\gamma V^{\pi_\theta}(\mathbf{s}_{t+1}) - V^{\pi_\theta}(\mathbf{s}_t))\right]$$
$$= E_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)\right] + E_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t (\gamma V^{\pi_\theta}(\mathbf{s}_{t+1}) - V^{\pi_\theta}(\mathbf{s}_t))\right]$$
$$= E_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t (r(\mathbf{s}_t, \mathbf{a}_t) + \gamma V^{\pi_\theta}(\mathbf{s}_{t+1}) - V^{\pi_\theta}(\mathbf{s}_t))\right]$$
$$= E_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)\right]$$

Policy Gradient as Policy Iteration

$$J(\theta') - J(\theta) = E_{\tau \sim p_{\theta'}(\tau)}\left[\sum_t \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)\right]$$

expectation under $\pi_{\theta'}$

advantage under π_θ

$$E_{\tau \sim p_{\theta'}(\tau)}\left[\sum_t \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)\right] = \sum_t E_{\mathbf{s}_t \sim p_{\theta'}(\mathbf{s}_t)}\left[E_{\mathbf{a}_t \sim \pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}\left[\gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)\right]\right]$$

$$= \sum_t E_{\mathbf{s}_t \sim p_{\theta'}(\mathbf{s}_t)}\left[E_{\mathbf{a}_t \sim \underline{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}}\left[\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}\gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)\right]\right]$$

is it OK to use $p_\theta(\mathbf{s}_t)$ instead?

Policy Gradient as Policy Iteration

Can we ignore distribution mismatch?

$$\sum_t E_{\mathbf{s}_t \sim p_{\theta'}(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \left[\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \right] \stackrel{?}{\approx} \underbrace{\sum_t E_{\mathbf{s}_t \sim \underline{p_\theta(\mathbf{s}_t)}} \left[E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \left[\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \right]}_{\bar{A}(\theta')}$$

why do we want this to be true?

$$J(\theta') - J(\theta) \approx \bar{A}(\theta') \quad \Rightarrow \quad \theta' \leftarrow \arg\max_{\theta'} \bar{A}(\theta)$$

2. Use $\hat{A}^\pi(\mathbf{s}_t, \mathbf{a}_t)$ to get *improved* policy π'

is it true? and when?

$p_\theta(\mathbf{s}_t)$ is *close* to $p_{\theta'}(\mathbf{s}_t)$ when π_θ is *close* to $\pi_{\theta'}$

Bounding the distribution change

Claim: $p_\theta(\mathbf{s}_t)$ is *close* to $p_{\theta'}(\mathbf{s}_t)$ when π_θ is *close* to $\pi_{\theta'}$

Simple case: assume π_θ is a *deterministic* policy $\mathbf{a}_t = \pi_\theta(\mathbf{s}_t)$

$\pi_{\theta'}$ is *close* to π_θ if $\pi_{\theta'}(\mathbf{a}_t \neq \pi_\theta(\mathbf{s}_t)|\mathbf{s}_t) \leq \epsilon$

$$p_{\theta'}(\mathbf{s}_t) = \underbrace{(1-\epsilon)^t}_{\text{probability we made no mistakes}} p_\theta(\mathbf{s}_t) + (1-(1-\epsilon)^t))\underbrace{p_{\text{mistake}}(\mathbf{s}_t)}_{\text{some \textit{other} distribution}}$$

$$|p_{\theta'}(\mathbf{s}_t) - p_\theta(\mathbf{s}_t)| = (1-(1-\epsilon)^t)|p_{\text{mistake}}(\mathbf{s}_t) - p_\theta(\mathbf{s}_t)| \leq 2(1-(1-\epsilon)^t)$$

$$\leq 2\epsilon t$$

useful identity: $(1-\epsilon)^t \geq 1 - \epsilon t$ for $\epsilon \in [0,1]$

not a great bound, but a bound!

Bounding the distribution change (cont.)

Claim: $p_\theta(\mathbf{s}_t)$ is *close* to $p_{\theta'}(\mathbf{s}_t)$ when π_θ is *close* to $\pi_{\theta'}$

General case: assume π_θ is an arbitrary distribution

$\pi_{\theta'}$ is *close* to π_θ if $|\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t) - \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)| \leq \epsilon$ for all $\mathbf{s}_t$

Useful lemma: if $|p_X(x) - p_Y(x)| = \epsilon$, exists $p(x, y)$ such that $p(x) = p_X(x)$ and $p(y) = p_Y(y)$ and $p(x = y) = 1 - \epsilon$

$\Rightarrow p_X(x)$ "agrees" with $p_Y(y)$ with probability ϵ

$\Rightarrow \pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)$ takes a different action than $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ with probability at most ϵ

$$|p_{\theta'}(\mathbf{s}_t) - p_\theta(\mathbf{s}_t)| = (1 - (1-\epsilon)^t)|p_{\text{mistake}}(\mathbf{s}_t) - p_\theta(\mathbf{s}_t)| \leq 2(1 - (1-\epsilon)^t)$$

$$\leq 2\epsilon t$$

Bounding the distribution change (cont.)

$\pi_{\theta'}$ is *close* to π_θ if $|\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t) - \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)| \leq \epsilon$ for all $\mathbf{s}_t$

$$|p_{\theta'}(\mathbf{s}_t) - p_\theta(\mathbf{s}_t)| \leq 2\epsilon t$$

$$E_{p_{\theta'}(\mathbf{s}_t)}[f(\mathbf{s}_t)] = \sum_{\mathbf{s}_t} p_{\theta'}(\mathbf{s}_t) f(\mathbf{s}_t) \geq \sum_{\mathbf{s}_t} p_\theta(\mathbf{s}_t) f(\mathbf{s}_t) - |p_\theta(\mathbf{s}_t) - p_{\theta'}(\mathbf{s}_t)| \max_{\mathbf{s}_t} f(\mathbf{s}_t)$$

$$\geq E_{p_\theta(\mathbf{s}_t)}[f(\mathbf{s}_t)] - 2\epsilon t \max_{\mathbf{s}_t} f(\mathbf{s}_t)$$

$$\sum_t E_{\mathbf{s}_t \sim p_{\theta'}(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \left[\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \right] \geq$$

$O(Tr_{\max})$ or $O\left(\frac{r_{\max}}{1-\gamma}\right)$

$$\sum_t E_{\mathbf{s}_t \sim p_\theta(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \left[\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \right] - \sum_t 2\epsilon t C$$

maximizing this maximizes a bound on the thing we want!

Where are we at so far?

$$\theta' \leftarrow \arg\max_{\theta'} \sum_t E_{\mathbf{s}_t \sim p_\theta(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \left[\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \right]$$

$$\text{such that } |\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t) - \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)| \leq \epsilon$$

for small enough ϵ, this is guaranteed to improve $J(\theta') - J(\theta)$

a more convenient bound: $|\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t) - \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)| \leq \sqrt{\frac{1}{2} D_{\mathrm{KL}}(\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t) \| \pi_\theta(\mathbf{a}_t|\mathbf{s}_t))}$

Some Useful Preliminaries: Taylor Series

Approximation of a differentiable function around a given point with sum of terms of the function's derivatives:

$$f(x) \approx f(x_0) + (x - x_0)^T \nabla f(x_0) + \frac{1}{2}(x - x_0)^T H (x - x_0) + \cdots$$

Some Useful Preliminaries: Constrained Optimization

- equality constraints: method of Lagrange multipliers

$$\begin{aligned} &optimize\ f(x) \\ &subject\ to: g(x) = 0 \end{aligned} \longrightarrow \mathcal{L}(x,\lambda) = f(x) + \lambda g(x)$$

- Inequality constraints: KKT

$$\begin{aligned} &optimize\ f(x) \\ &subject\ to: \\ &\quad g_i(x) \leq 0, \\ &\quad h_j(x) = 0 \end{aligned} \longrightarrow \begin{aligned} &\mathcal{L}(x,\mu,\lambda) = f(x) + \mu^T g(x) + \lambda^T h(x) \\ &subject\ to: \\ &\qquad \mu_i \geq 0 \\ &\qquad \mu^T g(x) = 0 \end{aligned}$$

Some Useful Preliminaries: KL-Divergence

A Common distance measure for distributions:

$$D_{KL}(p||q) = \int_x p(x) log \frac{p(x)}{q(x)} dx$$

Other useful distance measures:

- Total variation distance
- Wasserstein distance
- Jensen–Shannon divergence
- …

Some Useful Preliminaries: Fisher Information

likelihood function: $p_\theta(x)$

score function: $\nabla_\theta log p_\theta(x)$

Fisher information: measuring the amount of information that a random variable (x) carries about likelihood parameters (θ):

$$[I(\theta)]_{i,j} = E_{x \sim p_\theta(x)}[\,(\frac{\partial}{\partial \theta_i} log p_\theta(x))(\frac{\partial}{\partial \theta_j} log p_\theta(x))]$$

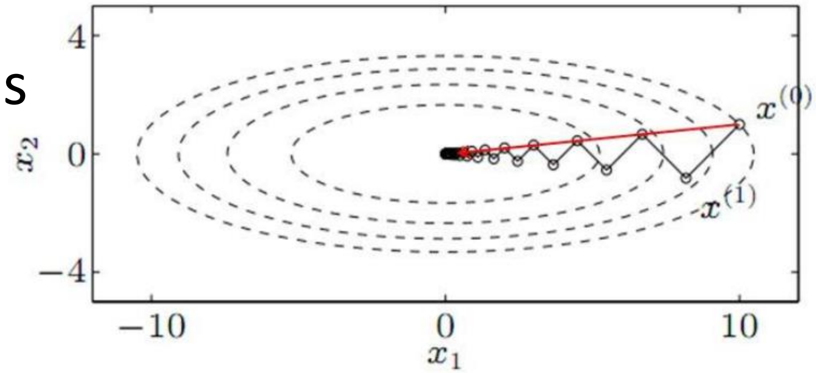
$$= -E_{x \sim p_\theta(x)}[\frac{\partial^2}{\partial \theta_i \partial \theta_j} log p_\theta(x)]$$

variance (covariance) of score function

curvature of score function

Importance of step size in RL

- Supervised learning: Step too far → next updates will fix it
- Reinforcement learning: Policy is determining data collection!
  - Step too far → bad policy
  - Next batch: collected under bad policy
  - May not be able to recover from a bad choice, collapse in performance!
- Learning rate tuning is hard
  - Poor conditioning could be more dangerous in RL settings
  - More sophisticated optimizers can reduce numerical issues
  - Need for advanced learning rate adjustment methods!

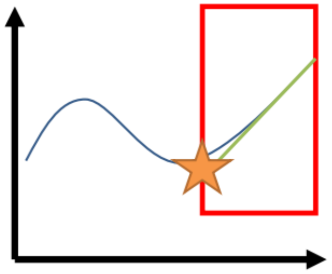


Natural Policy Gradient

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

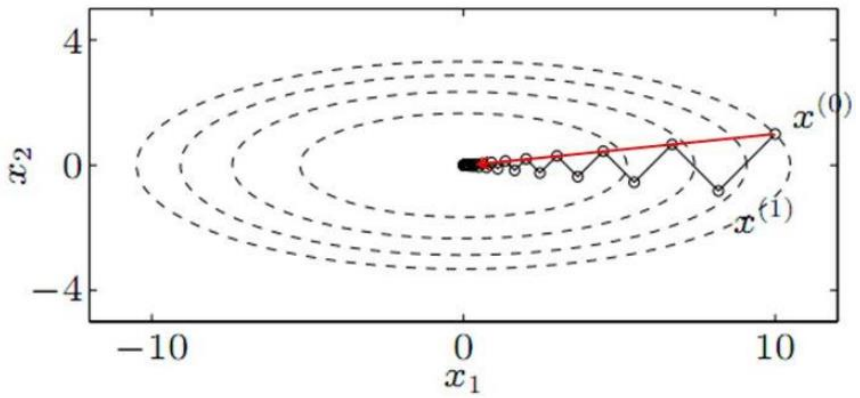
Could be shown as a constraint problem:

$$\theta' \leftarrow \arg\max_{\theta'} (\theta' - \theta)^T \nabla_\theta J(\theta) \text{ s.t. } \underbrace{\|\theta' - \theta\|^2 \leq \epsilon}_{\text{controls how far we go}}$$



Can we rescale the gradients to reduce the problem with poor conditioning?

$$\theta' \leftarrow \arg\max_{\theta'} (\theta' - \theta)^T \nabla_\theta J(\theta) \text{ s.t. } \underbrace{D(\pi_{\theta'}, \pi_\theta) \leq \epsilon}_{\text{parameterization-independent divergence measure}}$$



Natural Policy Gradient

$$\theta' \leftarrow \arg\max_{\theta'} (\theta' - \theta)^T \nabla_\theta J(\theta) \text{ s.t. } \underline{D(\pi_{\theta'}, \pi_\theta)} \le \epsilon$$

parameterization-independent divergence measure

usually KL-divergence: $D_{\text{KL}}(\pi_{\theta'} \| \pi_\theta) = E_{\pi_{\theta'}}[\log \pi_\theta - \log \pi_{\theta'}]$

Taylor expansion:

$$D_{\text{KL}}(\pi_{\theta'} \| \pi_\theta) \approx (\theta' - \theta)^T \underline{\mathbf{F}} (\theta' - \theta)$$

Fisher-information matrix

$$\mathbf{F} = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s}) \nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s})^T]$$

can estimate with samples

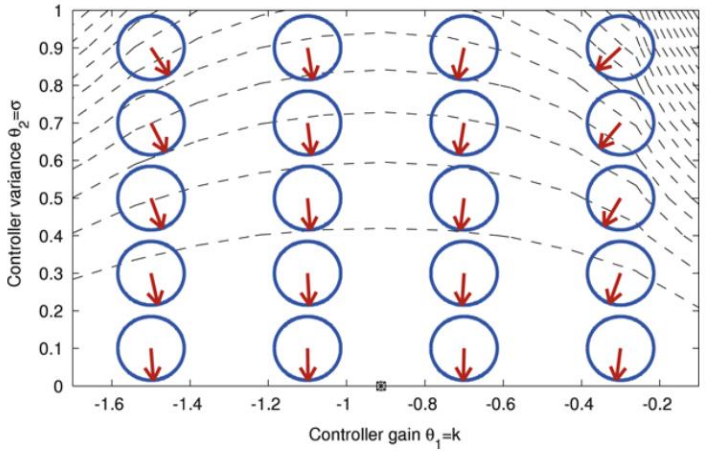
Natural Policy Gradient

$$D_{\mathrm{KL}}(\pi_{\theta'} \| \theta_\pi) \approx (\theta' - \theta)^T \mathbf{F} (\theta' - \theta)$$

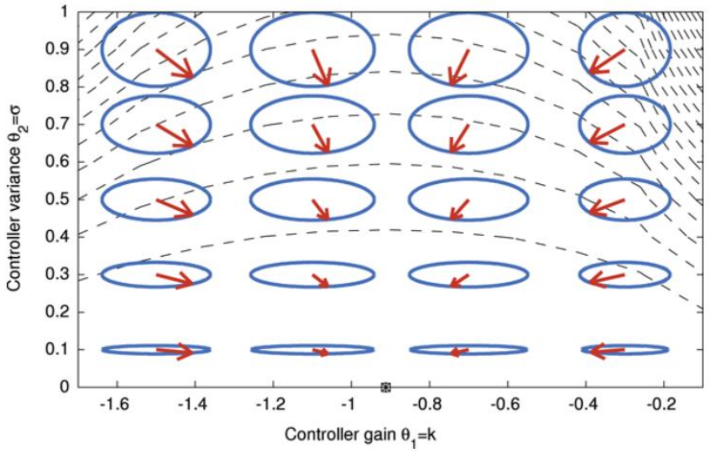
$$\theta' \leftarrow \arg\max_{\theta'} (\theta' - \theta)^T \nabla_\theta J(\theta) \text{ s.t. } D(\pi_{\theta'}, \pi_\theta) \leq \epsilon$$

$$\boxed{\theta \leftarrow \theta + \alpha \mathbf{F}^{-1} \nabla_\theta J(\theta)}$$

natural gradient: pick α



(a) Vanilla policy gradient.



(b) Natural policy gradient.

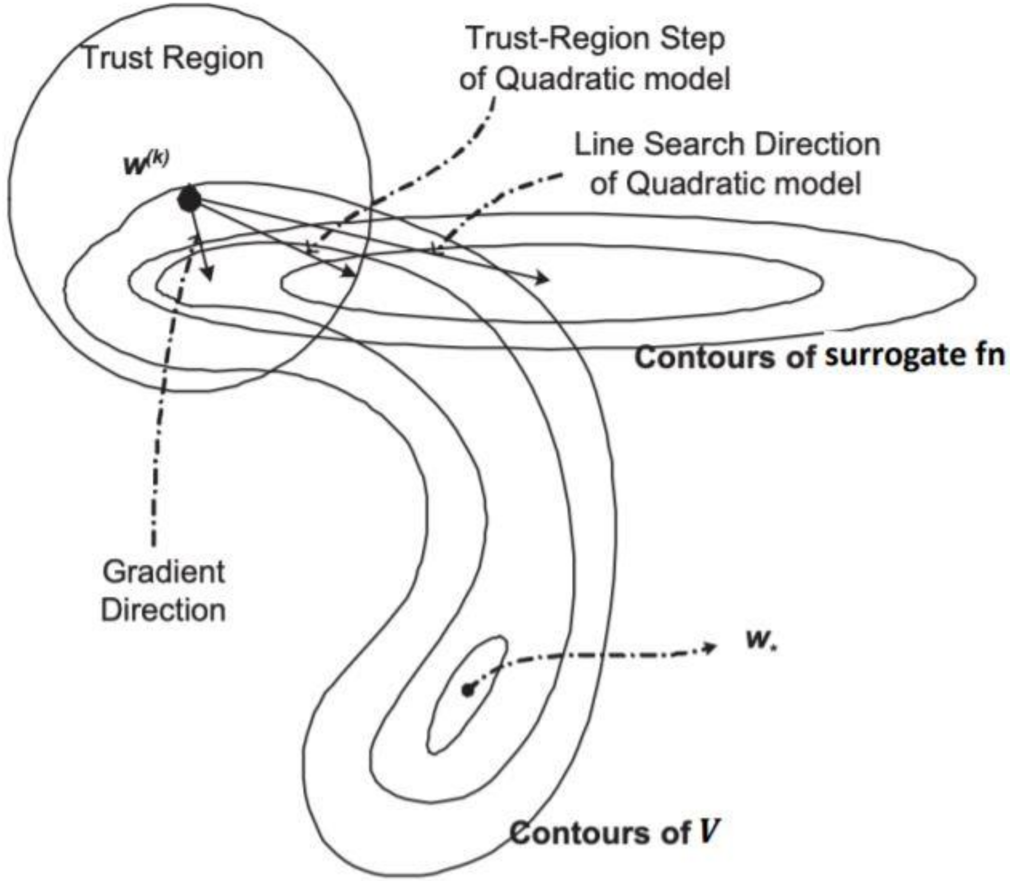
trust region policy optimization: pick ϵ

can solve for optimal α while solving $\mathbf{F}^{-1} \nabla_\theta J(\theta)$

(figure from Peters & Schaal 2008)

Trust Region Method

- We often optimize a surrogate objective
- Surrogate objective may be trustable only in a small region
- Limit search to small trust region



Cs885, waterloo 2022

Trust Region Policy Optimization

Recall from "Policy Gradient as Policy Iteration":

$$\theta' \leftarrow \arg\max_{\theta'} \overbrace{\sum_t E_{\mathbf{s}_t \sim p_\theta(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \left[\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \right]}^{\bar{A}(\theta')}$$

$$\text{such that } D_{\mathrm{KL}}(\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t) \| \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)) \leq \epsilon$$

surrogate loss

trust region

for small enough ϵ, this is guaranteed to improve $J(\theta') - J(\theta)$

Policy Gradient with Constraints

How do we enforce the constraint?

$$\theta' \leftarrow \arg\max_{\theta'} \sum_t E_{\mathbf{s}_t \sim p_\theta(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \left[\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \right]$$

$$\text{such that } D_{\mathrm{KL}}(\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t) \| \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)) \leq \epsilon$$

$$\mathcal{L}(\theta', \lambda) = \sum_t E_{\mathbf{s}_t \sim p_\theta(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \left[\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \right] - \lambda (D_{\mathrm{KL}}(\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t) \| \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)) - \epsilon)$$

1. Maximize $\mathcal{L}(\theta', \lambda)$ with respect to θ'
2. $\lambda \leftarrow \lambda + \alpha (D_{\mathrm{KL}}(\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t) \| \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)) - \epsilon)$

Intuition: raise λ if constraint violated too much, else lower it

an instance of *dual gradient descent*

Natural Gradient Based on Trust Region

Recall:

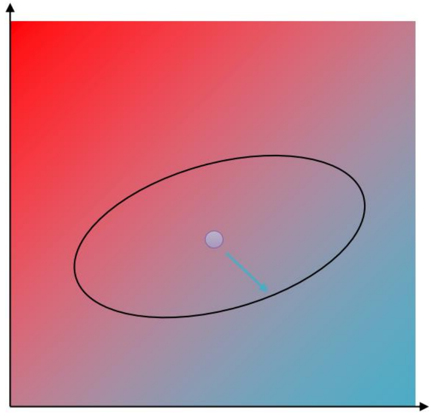
$$\theta' \leftarrow \arg\max_{\theta'} \nabla_\theta J(\theta)^T(\theta' - \theta)$$

$$\text{such that } D_{\mathrm{KL}}(\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)\|\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)) \leq \epsilon$$

$$D_{\mathrm{KL}}(\pi_{\theta'}\|\pi_\theta) \approx \frac{1}{2}(\theta' - \theta)^T\mathbf{F}(\theta' - \theta)$$

$$\theta' = \theta + \alpha\mathbf{F}^{-1}\nabla_\theta J(\theta)$$

natural gradient



$$\boxed{\alpha = \sqrt{\frac{2\epsilon}{\nabla_\theta J(\theta)^T\mathbf{F}^{-1}\nabla_\theta J(\theta)}}}$$

Proximal Policy Optimization

- TRPO is conceptually and computationally challenging in large part because of the constraint in the optimization.

$$D_{KL}(\pi_{\theta'}(.|s)\,||\pi_{\theta}(.|s)) \leq \epsilon$$

- What is the effect of the constraint?
- Recall KL-Divergence:

$$D_{KL}(\pi_{\theta'}(.|s)\,\big|\big|\pi_{\theta}(.|s)\big) = \Sigma_a \pi_{\theta'}(a|s) log \frac{\pi_{\theta'}(a|s)}{\pi_{\theta}(a|s)}$$

We are effectively constraining the ratio $\frac{\pi_{\theta'}(a|s)}{\pi_{\theta}(a|s)}$

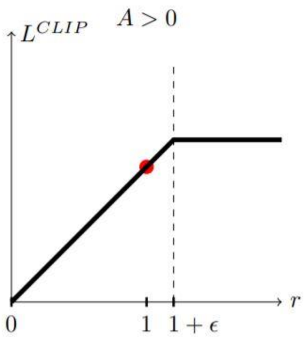
Proximal Policy Optimization

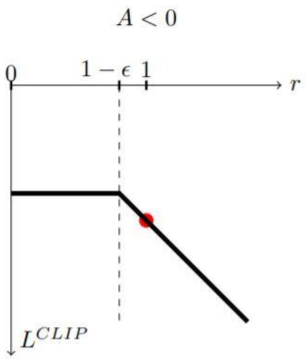
- Let's design a simpler objective that directly constrains $\frac{\pi_{\theta'}(a|s)}{\pi_\theta(a|s)}$

$$\underset{\theta'}{\operatorname{argmax}} E_{\{s\sim\mu_\theta, a\sim\pi_\theta\}}\, min \begin{cases} \dfrac{\pi_{\theta'}(a|s)}{\pi_\theta(a|s)} A^{\pi_\theta}(s,a), \\ clip\left(\dfrac{\pi_{\theta'}(a|s)}{\pi_\theta(a|s)}, 1-\epsilon, 1+\epsilon\right) A^{\pi_\theta}(s,a) \end{cases}$$

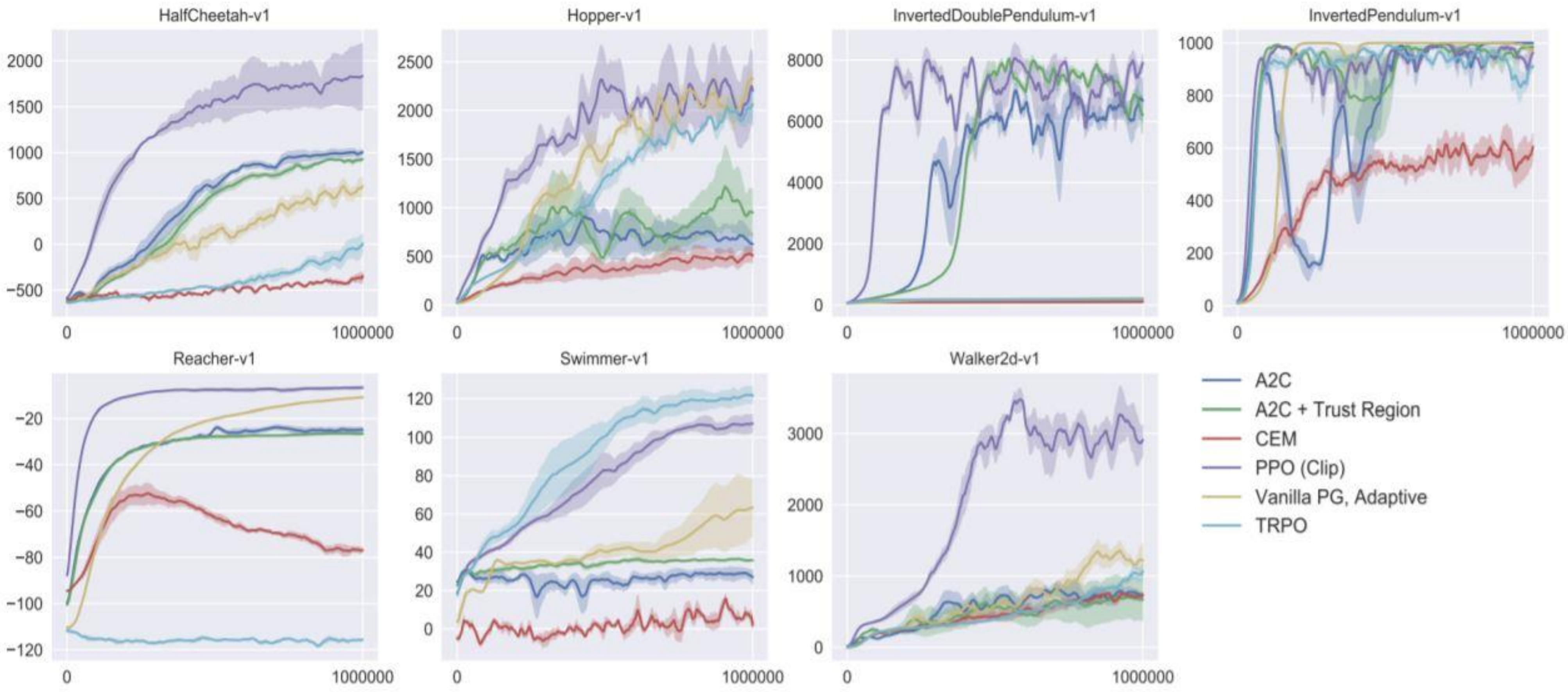
could be easily implemented with auto-diff packages

where $clip(x, 1-\epsilon, 1+\epsilon) = \begin{cases} 1-\epsilon & if\ x < 1-\epsilon \\ x & if\ 1-\epsilon \le x \le 1+\epsilon \\ 1+\epsilon & if\ x > 1+\epsilon \end{cases}$





PPO vs TRPO


HalfCheetah-v1
Hopper-v1
InvertedDoublePendulum-v1
InvertedPendulum-v1
Reacher-v1
Swimmer-v1
Walker2d-v1
A2C
A2C + Trust Region
CEM
PPO (Clip)
Vanilla PG, Adaptive
TRPO

Review

- Policy gradient = policy iteration
- Optimize advantage under new policy state distribution
- Using old policy state distribution optimizes a bound, if the policies are close enough
- Results in constrained optimization problem
- Gradient ascent: first order approximation to objective
- Regular gradient ascent has the wrong constraint (in parameter space): use natural gradient with D_{KL} constraint
- PPO simply uses the importance sampling ratio for regularization

Policy Gradient in Practice

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \underline{\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})} \hat{Q}_{i,t}$$

pretty inefficient to compute these explicitly!

How can we compute policy gradients with automatic differentiation?

We need a graph such that its gradient is the policy gradient!

Implementing Policy Gradient

How can we compute policy gradients with automatic differentiation?

We need a graph such that its gradient is the policy gradient!

maximum likelihood: $\nabla_\theta J_{\mathrm{ML}}(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})$ $\quad J_{\mathrm{ML}}(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})$

Just implement "pseudo-loss" as a weighted maximum likelihood:

$$\tilde{J}(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\hat{Q}_{i,t}$$

cross entropy (discrete) or squared error (Gaussian)

Implementing Policy Gradient

Policy gradient:

```
# Given:
# actions - (N*T) x Da tensor of actions
# states - (N*T) x Ds tensor of states
# q_values – (N*T) x 1 tensor of estimated state-action values
# Build the graph:
logits = policy.predictions(states) # This should return (N*T) x Da tensor of action logits
negative_likelihoods = tf.nn.softmax_cross_entropy_with_logits(labels=actions, logits=logits)
weighted_negative_likelihoods = tf.multiply(negative_likelihoods, q_values)
loss = tf.reduce_mean(weighted_negative_likelihoods)
gradients = loss.gradients(loss, variables)
```

$$\tilde{J}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \hat{Q}_{i,t}$$

q_values

Related Papers

- Williams (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning: introduces REINFORCE algorithm
- Sutton, McAllester, Singh, Mansour (1999). Policy gradient methods for reinforcement learning with function approximation: actor-critic algorithms with value function approximation
- Mnih, Badia, Mirza, Graves, Lillicrap, Harley, Silver, Kavukcuoglu (2016). Asynchronous methods for deep reinforcement learning: A3C, parallel online actor-critic
- Schulman, Moritz, L., Jordan, Abbeel (2016). High-dimensional continuous control using generalized advantage estimation: TD(λ) actor-critic

Related Papers

- Degris, White, Sutton. (2012). Off-policy actor-critic: off-policy actor-critic with importance sampling
- Silver et al. (2014). Deterministic policy gradient algorithms: DPG
- Lillicrap et al. (2016). Continuous control with deep reinforcement learning: continuous Q-learning with actor network for approximate maximization: DDPG
- Kakade (2001). A Natural Policy Gradient: natural policy gradient
- Schulman, L., Moritz, Jordan, Abbeel (2015). Trust region policy optimization: TRPO
- Schulman, Wolski, Dhariwal, Radford, Klimov (2017). Proximal policy optimization algorithms: PPO