Carnegie Mellon

School of Computer Science

Deep Reinforcement Learning and Control

Natural Policy Gradients, TRPO, PPO

CMU 10703

Katerina Fragkiadaki

Part of the slides adapted from John Shulman and Joshua Achiam

Stochastic policies

continuous actions

θ

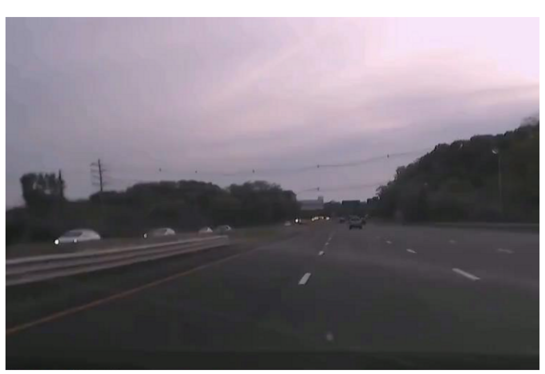
$\mu_\theta(s)$

$\sigma_\theta(s)$

usually multivariate Gaussian

$$a \sim \mathcal{N}(\mu_\theta(s), \sigma_\theta^2(s))$$

discrete actions



θ

$p_\theta(s)$

almost always categorical

$$a \sim \text{Cat}(p_\theta(s))$$

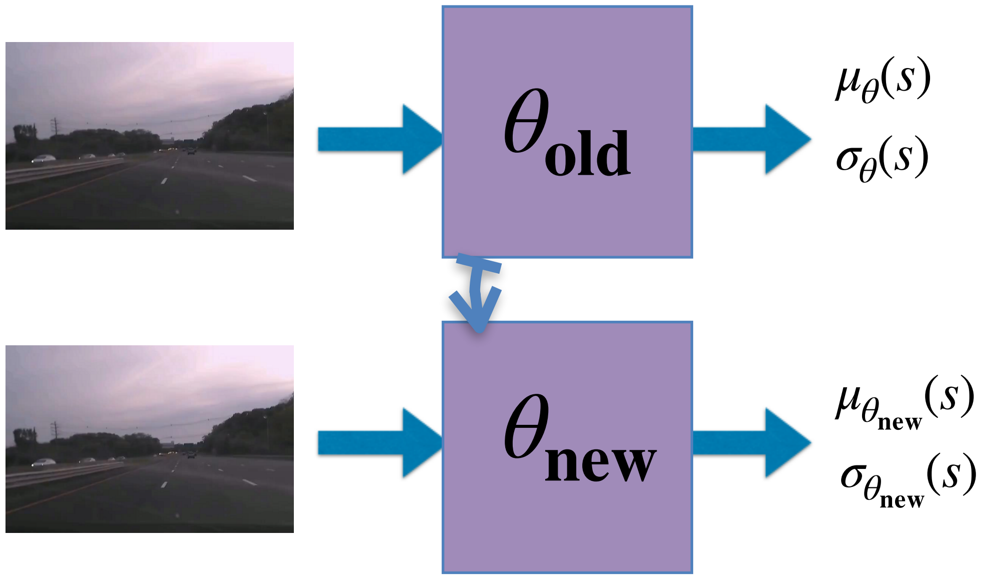
Policy Gradients

Monte Carlo Policy Gradients (REINFORCE), gradient direction: $\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \hat{A}_t \right]$

Actor-Critic Policy Gradient: $\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) A_{\mathbf{w}}(s_t) \right]$

1. Collect trajectories for policy π_θ
2. Estimate advantages A
3. Compute policy gradient $\hat{g}$
4. Update policy parameters $\theta_{new} = \theta + \epsilon \cdot \hat{g}$
5. GOTO 1

This lecture is all about the stepwise



What is the underlying objective function?

Policy gradients: $$\hat{g} \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\alpha_t^{(i)} | s_t^{(i)}) A(s_t^{(i)}, a_t^{(i)}), \quad \tau_i \sim \pi_\theta$$

What is our objective? Result from differentiating the objective function:

$$J^{PG}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \log \pi_\theta(\alpha_t^{(i)} | s_t^{(i)}) A(s_t^{(i)}, a_t^{(i)}) \quad \tau_i \sim \pi_\theta$$

Is this our objective? We cannot both maximize over a variable and sample from it.

Well, we cannot optimize it too far, our advantage estimates are from samples of \pi_theta_{old}. However, this constraint of "cannot optimize too far from \theta_{old}" does not appear anywhere in the objective.

Compare to supervised learning and maximum likelihood estimation (MLE). Imagine we have access to expert actions, then the loss function we want to optimize is:

$$J^{SL}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \log \pi_\theta(\tilde{\alpha}_t^{(i)} | s_t^{(i)}), \quad \tau_i \sim \pi^* \qquad \text{+regularization}$$

which maximizes the probability of expert actions in the training set.
Is this our SL objective?

Well, as a matter of fact, we care about test error, but this is a long story, the short answer is yes, this is good enough for us to optimize if we regularize.

Policy Gradients

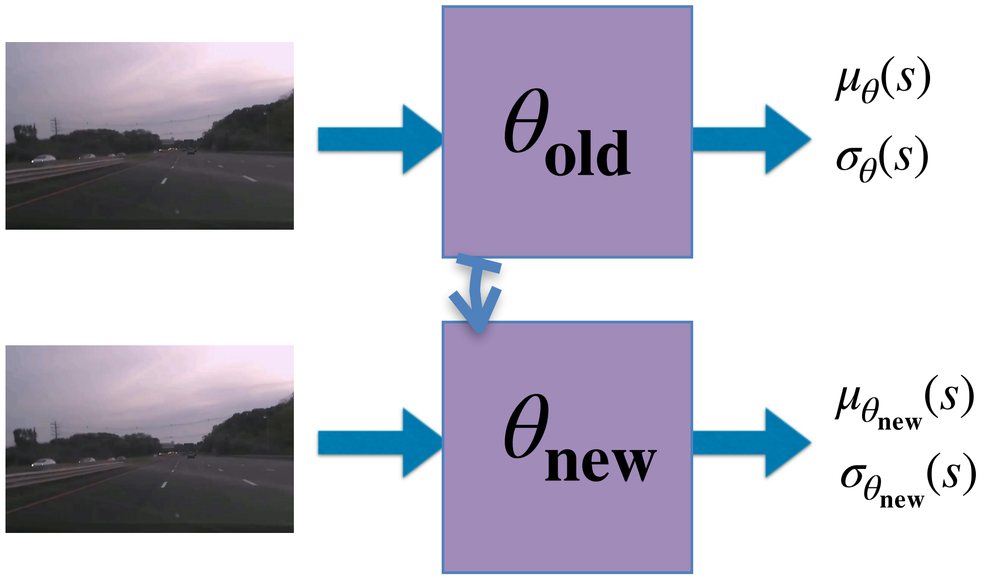
Monte Carlo Policy Gradients (REINFORCE), gradient direction: $\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \hat{A}_t \right]$

Actor-Critic Policy Gradient: $\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) A_{\mathbf{w}}(s_t) \right]$

1. Collect trajectories for policy π_θ
2. Estimate advantages A
3. Compute policy gradient $\hat{g}$
4. Update policy parameters $\theta_{new} = \theta + \epsilon \cdot \hat{g}$
5. GOTO 1

This lecture is all about the stepwise

It is also about writing down an objective that we can optimize with PG, and the procedure 1,2,3,4,5 will be the result of this objective maximization



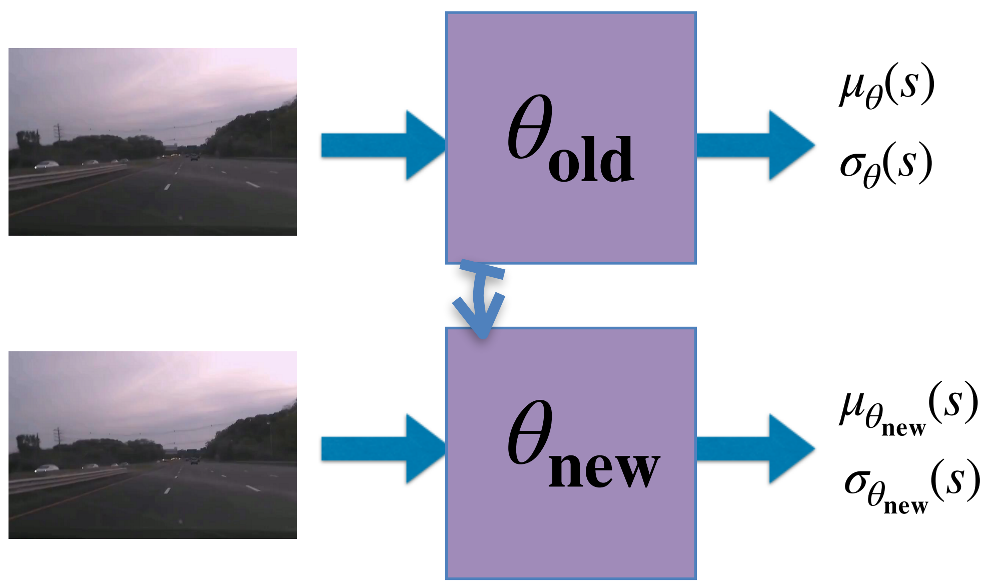
Policy Gradients

Monte Carlo Policy Gradients (REINFORCE), gradient direction: $\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \hat{A}_t \right]$

Actor-Critic Policy Gradient: $\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) A_{\mathbf{w}}(s_t) \right]$

1. Collect trajectories for policy π_θ
2. Estimate advantages A
3. Compute policy gradient $\hat{g}$
4. Update policy parameters $\theta_{new} = \theta + \epsilon \cdot \hat{g}$
5. GOTO 1

Two problems with the vanilla formulation:
1. Hard to choose stepwise ϵ
2. Sample inefficient: we cannot use data collected with policies of previous iterations



Hard to choose stepsizes

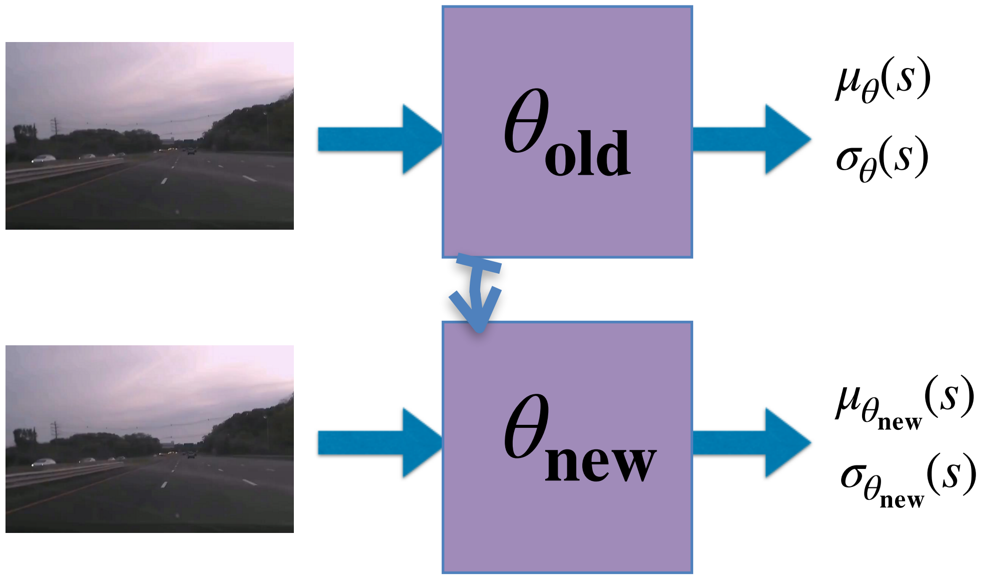
Monte Carlo Policy Gradients (REINFORCE), gradient direction: $\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \hat{A}_t \right]$

Actor-Critic Policy Gradient: $\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) A_{\mathbf{w}}(s_t) \right]$

1. Collect trajectories for policy π_θ
2. Estimate advantages A
3. Compute policy gradient $\hat{g}$
4. Update policy parameters $\boxed{\theta_{new} = \theta + \epsilon \cdot \hat{g}}$
5. GOTO 1

- Step too big
  Bad policy->data collected under bad policy-> we cannot recover
  (in Supervised Learning, data does not depend on neural network weights)
- Step too small
  Not efficient use of experience
  (in Supervised Learning, data can be trivially re-used)

Gradient descent in parameter space does not take into account the resulting distance in the (output) policy space between $\pi_{\theta_{\text{old}}}(s)$ and $\pi_{\theta_{\text{new}}}(s)$



Hard to choose stepsizes

Monte Carlo Policy Gradients (REINFORCE), gradient direction: $\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \hat{A}_t \right]$

Actor-Critic Policy Gradient: $\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) A_{\mathbf{w}}(s_t) \right]$

1. Collect trajectories for policy π_θ
2. Estimate advantages A
3. Compute policy gradient $\hat{g}$
4. Update policy parameters $\boxed{\theta_{new} = \theta + \epsilon \cdot \hat{g}}$
5. GOTO 1

Consider a family of policies with parametrization:

$$\pi_\theta(a) = \begin{cases} \sigma(\theta) & a = 1 \\ 1 - \sigma(\theta) & a = 2 \end{cases}$$

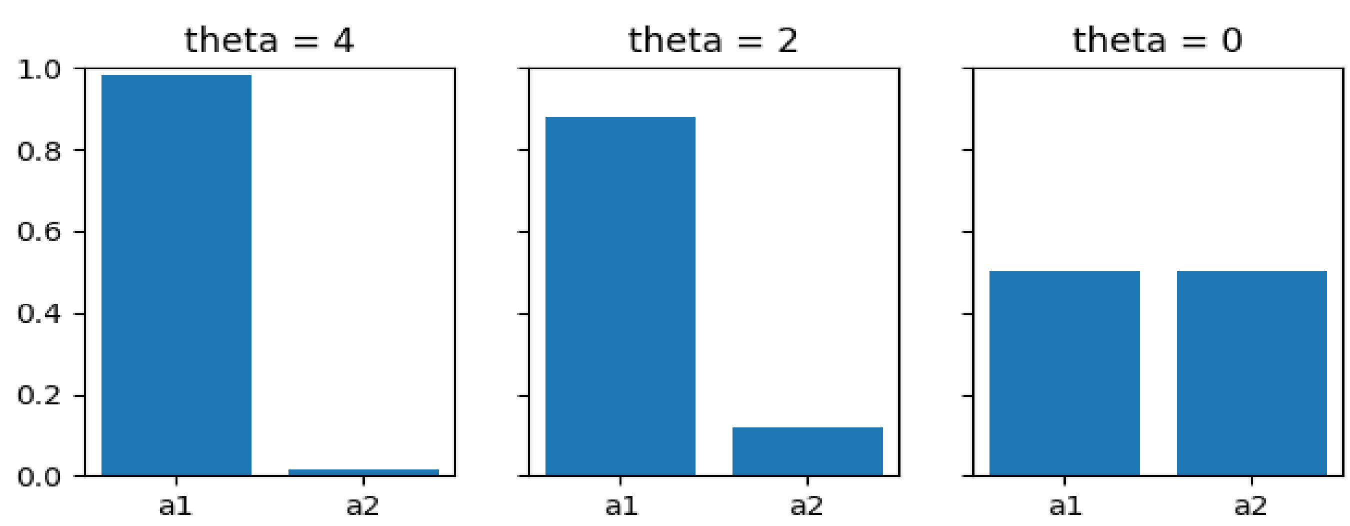


Figure: Small changes in the policy parameters can unexpectedly lead to **big** changes in the policy.

Notation

We will use the following to denote values of parameters and corresponding policies before and after an update:

$$\theta_{old} \to \theta_{new}$$

$$\pi_{old} \to \pi_{new}$$

$$\theta \to \theta'$$

$$\pi \to \pi'$$

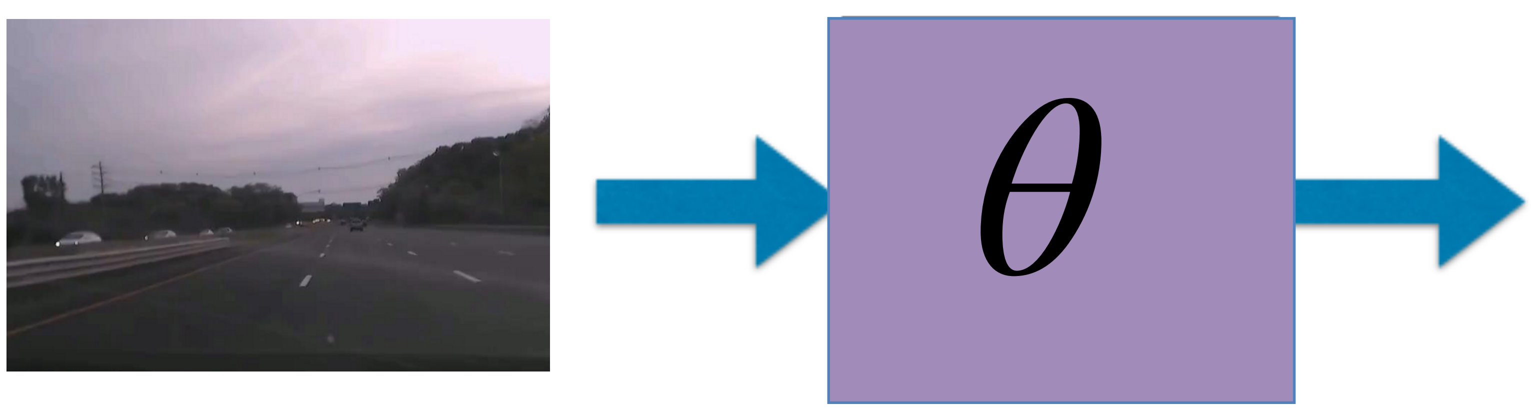
Gradient Descent in Parameter Space

The stepwise in gradient descent results from solving the following optimization problem, e.g., using line search:

$$d* = \arg\max_{\boxed{\|d\| \leq \epsilon}} J(\theta + d)$$

Euclidean distance in parameter space

It is hard to predict the result on the parameterized distribution..



$$\mu_\theta(s)$$
$$\sigma_\theta(s)$$

SGD: $\theta_{new} = \theta_{old} + d*$

Gradient Descent in Distribution Space

The stepwise in gradient descent results from solving the following optimization problem, e.g., using line search:

$$d^* = \arg\max_{\|d\| \leq \epsilon} J(\theta + d)$$

SGD: $\theta_{new} = \theta_{old} + d^*$

Euclidean distance in parameter space

It is hard to predict the result on the parameterized distribution.. hard to pick the threshold epsilon

Natural gradient descent: the stepwise in parameter space is determined by considering the KL divergence in the distributions before and after the update:

$$\boxed{d^* = \arg \max_{d,\ s.t.\ \mathrm{KL}(\pi_\theta \| \pi_{\theta+d}) \leq \epsilon} J(\theta + d)}$$

KL divergence in distribution space

Easier to pick the distance threshold!!!

$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i) \log\left(\frac{P(i)}{Q(i)}\right)$$

$$D_{\mathrm{KL}}(P\|Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx$$

Solving the KL Constrained Problem

Unconstrained penalized objective:

$$d^* = \arg\max_d J(\theta + d) - \lambda(\mathrm{D_{KL}}\left[\pi_\theta \| \pi_{\theta+d}\right] - \epsilon)$$

First order Taylor expansion for the loss and second order for the KL:

$$\approx \arg\max_d J(\theta_{old}) + \nabla_\theta J(\theta) \mid_{\theta=\theta_{old}} \cdot d - \frac{1}{2}\lambda(d^\top \nabla_\theta^2 \mathrm{D_{KL}}\left[\pi_{\theta_{old}} \| \pi_\theta\right] \mid_{\theta=\theta_{old}} d) + \lambda\epsilon$$

Taylor expansion of KL

$$\mathrm{D_{KL}}(p_{\theta_{old}}|p_\theta) \approx \mathrm{D_{KL}}(p_{\theta_{old}}|p_{\theta_{old}}) + d^\top \nabla_\theta \mathrm{D_{KL}}(p_{\theta_{old}}|p_\theta)|_{\theta=\theta_{old}} + \frac{1}{2} d^\top \nabla^2_\theta \mathrm{D_{KL}}(p_{\theta_{old}}|p_\theta)|_{\theta=\theta_{old}}$$

$$\begin{aligned}
\nabla_\theta \mathrm{D_{KL}}(p_{\theta_{old}}|p_\theta)|_{\theta=\theta_{old}} &= -\nabla_\theta \mathbb{E}_{x\sim p_{\theta_{old}}} \log P_\theta(x)|_{\theta=\theta_{old}} \\
&= -\mathbb{E}_{x\sim p_{\theta_{old}}} \nabla_\theta \log P_\theta(x)|_{\theta=\theta_{old}} \\
&= -\mathbb{E}_{x\sim p_{\theta_{old}}} \frac{1}{P_{\theta_{old}}(x)} \nabla_\theta P_\theta(x)|_{\theta=\theta_{old}} \\
&= \int_x P_{\theta_{old}}(x) \frac{1}{P_{\theta_{old}}(x)} \nabla_\theta P_\theta(x)|_{\theta=\theta_{old}} \\
&= \int_x \nabla_\theta P_\theta(x)|_{\theta=\theta_{old}} \\
&= \nabla_\theta \int_x P_\theta(x)|_{\theta=\theta_{old}}. \\
&= 0
\end{aligned}$$

$$\mathrm{KL}(p_{\theta_{old}}|p_\theta) = \mathbb{E}_{x\sim p_{\theta_{old}}} \log\left(\frac{P_{\theta_{old}}(x)}{P_\theta(x)}\right)$$

Taylor expansion of KL

$$\mathrm{D_{KL}}(p_{\theta_{old}}|p_\theta) \approx \mathrm{D_{KL}}(p_{\theta_{old}}|p_{\theta_{old}}) + d^\top \nabla_\theta \mathrm{KL}(p_{\theta_{old}}|p_\theta)|_{\theta=\theta_{old}} + \frac{1}{2} d^\top \nabla^2_\theta \mathrm{D_{KL}}(p_{\theta_{old}}|p_\theta)|_{\theta=\theta_{old}} d$$

$$\begin{aligned}
\nabla^2_\theta \mathrm{D_{KL}}(p_{\theta_{old}}|p_\theta)|_{\theta=\theta_{old}} &= -\mathbb{E}_{x\sim p_{\theta_{old}}} \nabla^2_\theta \log P_\theta(x)|_{\theta=\theta_{old}} \\
&= -\mathbb{E}_{x\sim p_{\theta_{old}}} \nabla_\theta \left(\frac{\nabla_\theta P_\theta(x)}{P_\theta(x)} \right)|_{\theta=\theta_{old}} \\
&= -\mathbb{E}_{x\sim p_{\theta_{old}}} \left(\frac{\nabla^2_\theta P_\theta(x) P_\theta(x) - \nabla_\theta P_\theta(x) \nabla_\theta P_\theta(x)^\top}{P_\theta(x)^2} \right)|_{\theta=\theta_{old}} \\
&= -\mathbb{E}_{x\sim p_{\theta_{old}}} \frac{\nabla^2_\theta P_\theta(x)|_{\theta=\theta_{old}}}{P_{\theta_{old}}(x)} + \mathbb{E}_{x\sim p_{\theta_{old}}} \nabla_\theta \log P_\theta(x) \nabla_\theta \log P_\theta(x)^\top|_{\theta=\theta_{old}} \\
&= \mathbb{E}_{x\sim p_{\theta_{old}}} \nabla_\theta \log P_\theta(x) \nabla_\theta \log P_\theta(x)^\top|_{\theta=\theta_{old}}
\end{aligned}$$

$$\mathrm{D_{KL}}(p_{\theta_{old}}|p_\theta) = \mathbb{E}_{x\sim p_{\theta_{old}}} \log \left(\frac{P_{\theta_{old}}(x)}{P_\theta(x)} \right)$$

Fisher Information Matrix

Exactly equivalent to the Hessian of KL divergence!

$$\mathbf{F}(\theta) = \mathbb{E}_{\theta}\left[\nabla_{\theta}\log p_{\theta}(x)\,\nabla_{\theta}\log p_{\theta}(x)^{\top}\right]$$

$$\mathbf{F}(\theta_{old}) = \nabla^2_{\theta}\mathrm{D}_{\mathrm{KL}}(p_{\theta_{old}} | p_{\theta})\,|_{\theta=\theta_{old}}$$

$$\begin{aligned}\mathrm{D}_{\mathrm{KL}}(p_{\theta_{old}} | p_{\theta}) &\approx \mathrm{D}_{\mathrm{KL}}(p_{\theta_{old}} | p_{\theta_{old}}) + d^{\top}\nabla_{\theta}\mathrm{D}_{\mathrm{KL}}(p_{\theta_{old}} | p_{\theta})\,|_{\theta=\theta_{old}} + \frac{1}{2}d^{\top}\nabla^2_{\theta}\mathrm{D}_{\mathrm{KL}}(p_{\theta_{old}} | p_{\theta})\,|_{\theta=\theta_{old}}\,d \\ &= \frac{1}{2}d^{\top}\mathbf{F}(\theta_{old})d \\ &= \frac{1}{2}(\theta-\theta_{old})^{\top}\mathbf{F}(\theta_{old})(\theta-\theta_{old})\end{aligned}$$

Since KL divergence is roughly analogous to a distance measure between distributions, Fisher information serves as a local distance metric between distributions: how much you change the distribution if you move the parameters a little bit in a given direction.

Solving the KL Constrained Problem

Unconstrained penalized objective:

$$d* = \arg\max_{d} J(\theta + d) - \lambda(\mathrm{D}_{\mathrm{KL}}\left[\pi_\theta \| \pi_{\theta+d}\right] - \epsilon)$$

First order Taylor expansion for the loss and second order for the KL:

$$\approx \arg\max_{d} J(\theta_{old}) + \nabla_\theta J(\theta)\,|_{\theta=\theta_{old}} \cdot d - \frac{1}{2}\lambda(d^\top \nabla_\theta^2 \mathrm{D}_{\mathrm{KL}}\left[\pi_{\theta_{old}} \| \pi_\theta\right]|_{\theta=\theta_{old}} d) + \lambda\epsilon$$

Substitute for the information matrix:

$$= \arg\max_{d} \nabla_\theta J(\theta)\,|_{\theta=\theta_{old}} \cdot d - \frac{1}{2}\lambda(d^\top \mathbf{F}(\theta_{old}) d)$$

$$= \arg\min_{d} - \nabla_\theta J(\theta)\,|_{\theta=\theta_{old}} \cdot d + \frac{1}{2}\lambda(d^\top \mathbf{F}(\theta_{old}) d)$$

Natural Gradient Descent

Setting the gradient to zero:

$$0 = \frac{\partial}{\partial d}\left(- \nabla_\theta J(\theta)\,|_{\theta=\theta_{old}} \cdot d + \frac{1}{2}\lambda(d^\top \mathbf{F}(\theta_{old}) d)\right)$$

$$= - \nabla_\theta J(\theta)\,|_{\theta=\theta_{old}} + \frac{1}{2}\lambda(\mathbf{F}(\theta_{old}))d$$

$$d = \frac{2}{\lambda}\mathbf{F}^{-1}(\theta_{old}) \nabla_\theta J(\theta)\,|_{\theta=\theta_{old}}$$

The natural gradient:

$$\boxed{\tilde{\nabla} J(\theta) = \mathbf{F}^{-1}(\theta_{old}) \nabla_\theta J(\theta)}$$

$$\theta_{new} = \theta_{old} + \alpha \cdot \mathbf{F}^{-1}(\theta_{old})\hat{g}$$

$$\mathrm{D_{KL}}(\pi_{\theta_{old}} | \pi_\theta) \approx \frac{1}{2}(\theta - \theta_{old})^\top \mathbf{F}(\theta_{old})(\theta - \theta_{old})$$

$$\frac{1}{2}(\alpha g_N)^\top \mathbf{F}(\alpha g_N) = \epsilon$$

$$\alpha = \sqrt{\frac{2\epsilon}{(g_N^\top \mathbf{F} g_N)}}$$

Natural Gradient Descent

Algorithm 1 Natural Policy Gradient

Input: initial policy parameters θ_0
for $k = 0, 1, 2, \ldots$ **do**
Collect set of trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
Form sample estimates for

- policy gradient $\hat{g}_k$ (using advantage estimates)
- and KL-divergence Hessian / Fisher Information Matrix $\hat{H}_k$

Compute Natural Policy Gradient update:

$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\,\epsilon}{\hat{g}_k^T \hat{H}_k^{-1} \hat{g}_k}} \hat{H}_k^{-1} \hat{g}_k$$

end for

Both use samples from the current policy \pi_k

Natural Gradient Descent

Algorithm 1 Natural Policy Gradient

Input: initial policy parameters θ_0
for $k = 0, 1, 2, \ldots$ **do**
Collect set of trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
Form sample estimates for

- policy gradient $\hat{g}_k$ (using advantage estimates)
- and KL-divergence Hessian / Fisher Information Matrix $\hat{H}_k$

Compute Natural Policy Gradient update:

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

end for

---

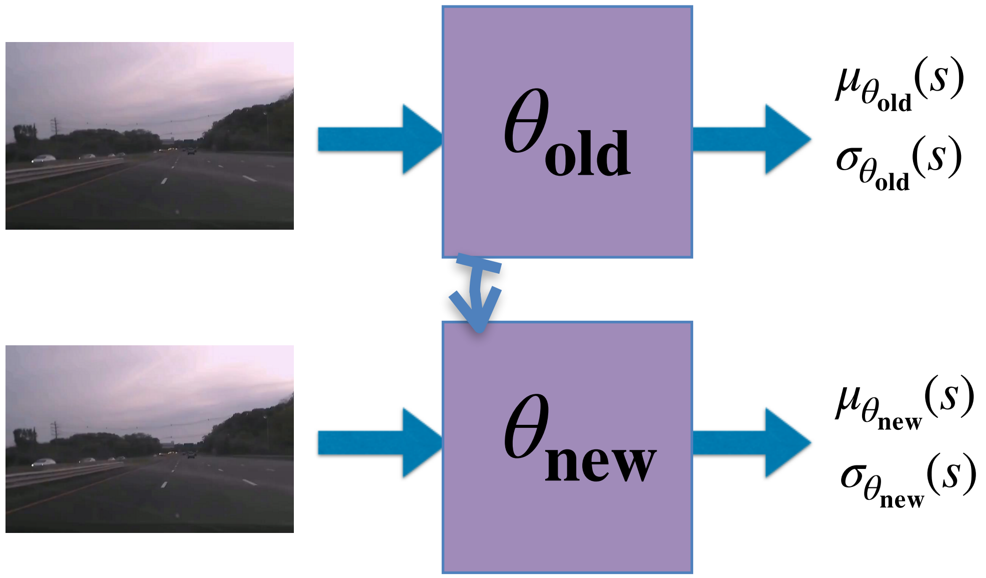
very expensive to compute for a large number of parameters!

Policy Gradients

Monte Carlo Policy Gradients (REINFORCE), gradient direction: $\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \hat{A}_t \right]$

Actor-Critic Policy Gradient: $\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) A_{\mathbf{w}}(s_t) \right]$

1. Collect trajectories for policy $\pi_{\theta_{old}}$
2. Estimate advantages A
3. Compute policy gradient $\hat{g}$
4. Update policy parameters $\theta_{new} = \theta_{old} + \epsilon \cdot \hat{g}$
5. GOTO 1



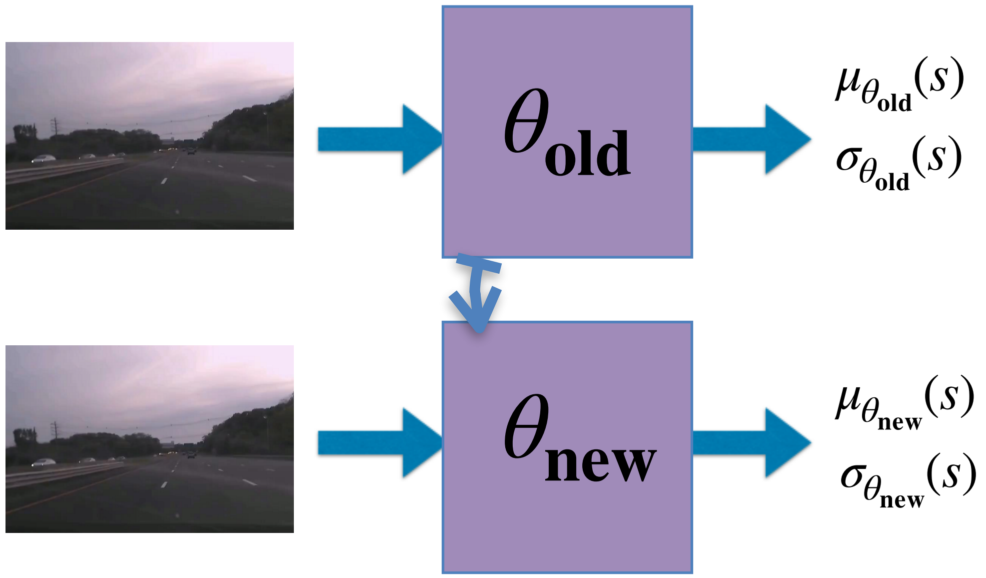
Policy Gradients

Monte Carlo Policy Gradients (REINFORCE), gradient direction: $\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \hat{A}_t \right]$

Actor-Critic Policy Gradient: $\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) A_{\mathbf{w}}(s_t) \right]$

1. Collect trajectories for policy $\pi_{\theta_{old}}$
2. Estimate advantages A
3. Compute policy gradient $\hat{g}$
4. Update policy parameters $\theta_{new} = \theta_{old} + \epsilon \cdot \hat{g}$
5. GOTO 1

- On policy learning can be extremely inefficient
- The policy changes only a little bit with each gradient step
- I want to be able to use earlier data..how to do that?



Off policy learning with Importance Sampling

$$
\begin{aligned}
J(\theta) &= \mathbb{E}_{\tau \sim \pi_\theta(\tau)}\left[R(\tau)\right] \\
&= \sum_{\tau} \pi_\theta(\tau) R(\tau) \\
&= \sum_{\tau} \pi_{\theta_{old}}(\tau) \frac{\pi_\theta(\tau)}{\pi_{\theta_{old}}(\tau)} R(\tau) \\
&= \sum_{\tau \sim \pi_{\theta_{old}}} \frac{\pi_\theta(\tau)}{\pi_{\theta_{old}}(\tau)} R(\tau) \\
&= \mathbb{E}_{\tau \sim \pi_{\theta_{old}}} \frac{\pi_\theta(\tau)}{\pi_{\theta_{old}}(\tau)} R(\tau)
\end{aligned}
$$

$$
\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta_{old}}} \frac{\nabla_\theta \pi_\theta(\tau)}{\pi_{\theta_{old}}(\tau)} R(\tau)
$$

$$
\nabla_\theta J(\theta)\,|_{\theta=\theta_{old}} = \mathbb{E}_{\tau \sim \pi_{\theta_{old}}} \nabla_\theta \log \pi_\theta(\tau)\,|_{\theta=\theta_{old}} R(\tau)
$$

<-Gradient evaluated at theta_old is unchanged

Off policy learning with Importance Sampling

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\tau)} \left[R(\tau) \right]$$
$$= \sum_{\tau} \pi_\theta(\tau) R(\tau)$$
$$= \sum_{\tau} \pi_{\theta_{old}}(\tau) \frac{\pi_\theta(\tau)}{\pi_{\theta_{old}}(\tau)} R(\tau)$$
$$= \sum_{\tau \sim \pi_{\theta_{old}}} \frac{\pi_\theta(\tau)}{\pi_{\theta_{old}}(\tau)} R(\tau)$$
$$= \mathbb{E}_{\tau \sim \pi_{\theta_{old}}} \frac{\pi_\theta(\tau)}{\pi_{\theta_{old}}(\tau)} R(\tau)$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta_{old}}} \frac{\nabla_\theta \pi_\theta(\tau)}{\pi_{\theta_{old}}(\tau)} R(\tau)$$

$$\nabla_\theta J(\theta) |_{\theta = \theta_{old}} = \mathbb{E}_{\tau \sim \pi_{\theta_{old}}} \nabla_\theta \log \pi_\theta(\tau) |_{\theta = \theta_{old}} R(\tau)$$

$$\frac{\pi_\theta(\tau)}{\pi_{\theta_{old}}(\tau)} = \prod_{i=1}^{T} \frac{\pi_\theta(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)}$$

$$\Rightarrow \quad J(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta_{old}}} \sum_{t=1}^{T} \prod_{t'=1}^{t} \frac{\pi_\theta(a'_t | s'_t)}{\pi_{\theta_{old}}(a'_t | s'_t)} \hat{A}_t$$

Now we can use data from the old policy, but the variance has increased by a lot! Those multiplications can explode or vanish!

Trust region Policy Optimization

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t \right] \\ \text{subject to} \quad & \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \leq \delta. \end{aligned}$$

- Also worth considering using a penalty instead of a constraint

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t \right] - \beta \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]]$$

Again the KL penalized problem!

J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. "Trust Region Policy Optimization".

Trust region Policy Optimization

- maximize$_\theta$ $L_{\pi_{\theta_{\mathrm{old}}}}(\pi_\theta) - \beta \cdot \overline{\mathrm{KL}}_{\pi_{\theta_{\mathrm{old}}}}(\pi_\theta)$
- Make linear approximation to $L_{\pi_{\theta_{\mathrm{old}}}}$ and quadratic approximation to KL term:

$$\underset{\theta}{\text{maximize}} \quad g \cdot (\theta - \theta_{\mathrm{old}}) - \tfrac{\beta}{2}(\theta - \theta_{\mathrm{old}})^T F(\theta - \theta_{\mathrm{old}})$$

$$\text{where} \quad g = \frac{\partial}{\partial \theta} L_{\pi_{\theta_{\mathrm{old}}}}(\pi_\theta)\big|_{\theta=\theta_{\mathrm{old}}}, \quad F = \frac{\partial^2}{\partial^2 \theta} \overline{\mathrm{KL}}_{\pi_{\theta_{\mathrm{old}}}}(\pi_\theta)\big|_{\theta=\theta_{\mathrm{old}}}$$

Exactly what we saw with natural policy gradient!
One important detail!

Trust region Policy Optimization

Due to the quadratic approximation, the KL constraint may be violated! What if we just do a line search to find the best stepsize, making sure:

- I am improving my objective J(\theta)
- The KL constraint is not violated!

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t \right] \\ \text{subject to} \quad & \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \le \delta. \end{aligned}$$

Algorithm 2 Line Search for TRPO

Compute proposed policy step $\Delta_k = \sqrt{\frac{2\delta}{\hat{g}_k^T \hat{H}_k^{-1} \hat{g}_k}} \hat{H}_k^{-1} \hat{g}_k$

for $j = 0, 1, 2, ..., L$ **do**

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

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

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

    break

  end if

end for

Trust region Policy Optimization

TRPO= NPG +Linesearch

Algorithm 3 Trust Region Policy Optimization

Input: initial policy parameters θ_0

for $k = 0, 1, 2, \ldots$ **do**

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

Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm

Form sample estimates for

- policy gradient $\hat{g}_k$ (using advantage estimates)
- and KL-divergence Hessian-vector product function $f(v) = \hat{H}_k v$

Use CG with n_{cg} iterations to obtain $x_k \approx \hat{H}_k^{-1} \hat{g}_k$

Estimate proposed step $\Delta_k \approx \sqrt{\frac{2\delta}{x_k^T \hat{H}_k x_k}} x_k$

Perform backtracking line search with exponential decay to obtain final update

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

end for

Trust region Policy Optimization

TRPO= NPG +Linesearch+monotonic improvement theorem!

Algorithm 3 Trust Region Policy Optimization

Input: initial policy parameters θ_0

for $k = 0, 1, 2, \ldots$ **do**

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

Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm

Form sample estimates for

- policy gradient $\hat{g}_k$ (using advantage estimates)
- and KL-divergence Hessian-vector product function $f(v) = \hat{H}_k v$

Use CG with n_{cg} iterations to obtain $x_k \approx \hat{H}_k^{-1} \hat{g}_k$

Estimate proposed step $\Delta_k \approx \sqrt{\frac{2\delta}{x_k^T \hat{H}_k x_k}} x_k$

Perform backtracking line search with exponential decay to obtain final update

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

end for

Relating objectives of two policies

Policy objective:

$$J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \sum_{t=0}^{\infty} \gamma^t r_t$$

Policy objective can be written in terms of old one:

$$J(\pi_{\theta'}) - J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi'_\theta} \sum_{t=0}^{\infty} \gamma^t A^{\pi_\theta}(s_t, a_t)$$

Equivalently for succinctness:

$$J(\pi') - J(\pi) = \mathbb{E}_{\tau \sim \pi'} \sum_{t=0}^{\infty} \gamma^t A^{\pi}(s_t, a_t)$$

Relating objectives of two policies

$$
\begin{aligned}
J(\pi') - J(\pi) &= \mathop{\mathrm{E}}_{\tau \sim \pi'} \left[\sum_{t=0}^{\infty} \gamma^t A^{\pi}(s_t, a_t) \right] \\
&= \mathop{\mathrm{E}}_{\tau \sim \pi'} \left[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t, a_t, s_{t+1}) + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t) \right) \right] \\
&= J(\pi') + \mathop{\mathrm{E}}_{\tau \sim \pi'} \left[\sum_{t=0}^{\infty} \gamma^{t+1} V^{\pi}(s_{t+1}) - \sum_{t=0}^{\infty} \gamma^t V^{\pi}(s_t) \right] \\
&= J(\pi') + \mathop{\mathrm{E}}_{\tau \sim \pi'} \left[\sum_{t=1}^{\infty} \gamma^t V^{\pi}(s_t) - \sum_{t=0}^{\infty} \gamma^t V^{\pi}(s_t) \right] \\
&= J(\pi') - \mathop{\mathrm{E}}_{\tau \sim \pi'} \left[V^{\pi}(s_0) \right] \\
&= J(\pi') - J(\pi)
\end{aligned}
$$

The initial state distribution is the same for both!

Approximately Optimal Approximate Reinforcement Learning, Kakade and Langford 2002

Relating objectives of two policies

Discounted state visitation distribution:

$$d^{\pi}(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t P(s_t = s \mid \pi)$$

$$\begin{aligned} J(\pi') - J(\pi) &= \mathbb{E}_{\tau\sim\pi'}\sum_{t=0}^{\infty}\gamma^t A^{\pi}(s_t, a_t) \\ &= \mathbb{E}_{s\sim d^{\pi'}, a\sim\pi'} A^{\pi}(s,a) \\ &= \mathbb{E}_{s\sim d^{\pi'}, a\sim\pi}\left[\frac{\pi'(a \mid s)}{\pi(a \mid s)} A^{\pi}(s,a)\right] \end{aligned}$$

But how are we supposed to sample states from the policy we are trying to optimize for…
Let's use the previous policy to sample them.

$$\begin{aligned} J(\pi') - J(\pi) &\approx \mathbb{E}_{s\sim d^{\pi}, a\sim\pi}\frac{\pi'(a \mid s)}{\pi(a \mid s)} A^{\pi}(s,a) \\ &= \mathscr{L}_{\pi}(\pi') \end{aligned}$$

It turns out we can bound this approximation error:

$$\left|J(\pi') - \big(J(\pi) + \mathcal{L}_{\pi}(\pi')\big)\right| \leq C\sqrt{\mathop{\mathrm{E}}_{s\sim d^{\pi}}\left[D_{KL}(\pi' \| \pi)[s]\right]}$$

Constrained Policy Optimization, Achiam et al. 2017

Relating objectives of two policies

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

$$= \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\frac{\pi'(a_t\,|\,s_t)}{\pi(a_t\,|\,s_t)}A^{\pi}(s_t,a_t)\right]$$

This is something we can optimize using trajectories from the old policy!

Compare to Importance Sampling:

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_{\theta_{old}}}\sum_{t=1}^{T}\prod_{t'=1}^{t}\frac{\pi_{\theta}(a'_t\,|\,s'_t)}{\pi_{\theta_{old}}(a'_t\,|\,s'_t)}\hat{A}_t$$

Now we do not have the product! So, the gradient will have much smaller variance! (Yes, but we have approximated, that's why!) What is the gradient?

$$\nabla_{\theta}\mathcal{L}_{\theta_k}^{\theta}\,|_{\theta=\theta_k} = \mathbb{E}_{\tau\sim\pi_{\theta_k}}\left[\sum_{t=0}^{\infty}\gamma^t\frac{\nabla_{\theta}\pi_{\theta}(a_t\,|\,s_t)\,|_{\theta=\theta_k}}{\pi_{\theta_k}(a_t\,|\,s_t)}A^{\pi_{\theta_k}}(s_t,a_t)\right]$$

$$= \mathbb{E}_{\tau\sim\pi_{\theta_k}}\left[\sum_{t=0}^{\infty}\gamma^t\nabla_{\theta}\log\pi_{\theta}(a_t\,|\,s_t)\,|_{\theta=\theta_k}A^{\pi_{\theta_k}}(s_t,a_t)\right]$$

Monotonic Improvement Theorem

$$|J(\pi') - \left(J(\pi) + \mathscr{L}_\pi(\pi')\right)| \leq C\sqrt{\mathbb{E}_{s\sim d^\pi}\left[\mathrm{KL}(\pi' | \pi)[s]\right]}$$

$$\Rightarrow J(\pi') - J(\pi) \geq \mathscr{L}_\pi(\pi') - C\sqrt{\mathbb{E}_{s\sim d^\pi}\left[\mathrm{KL}(\pi' | \pi)[s]\right]}$$

Given policy π, we want to optimize over policy π' to maximize .

- If we maximize the RHS we are guaranteed to maximize the LHS.
- We know how to maximize the RHS. I can estimate both quantities of \pi' with sampled from \pi

- But will i have a better policy \pi'? (knowing that the distance of the objectives is maximized is not enough, there needs to be positive or equal to zero)

Monotonic Improvement Theorem

Proof of improvement guarantee: Suppose π_{k+1} and π_k are related by

$$\pi_{k+1} = \arg\max_{\pi'} \mathcal{L}_{\pi_k}(\pi') - C\sqrt{\mathop{\mathrm{E}}_{s\sim d^{\pi_k}}\left[D_{KL}(\pi'||\pi_k)[s]\right]}.$$

- π_k is a feasible point, and the objective at π_k is equal to 0.
  - $\mathcal{L}_{\pi_k}(\pi_k) \propto \mathop{\mathrm{E}}_{s,a\sim d^{\pi_k},\pi_k}\left[A^{\pi_k}(s,a)\right] = 0$
  - $D_{KL}(\pi_k||\pi_k)[s] = 0$
- $\implies$ optimal value ≥ 0
- $\implies$ by the performance bound, $J(\pi_{k+1}) - J(\pi_k) \geq 0$

Approximate Monotonic Improvement

- Theory is very conservative (high value of C) and we will use KL distance of pi' and pi as a constraint (trust region) as opposed to a penalty:

$$
\begin{aligned}
\pi_{k+1} = \arg\max_{\pi'} \; & \mathcal{L}_{\pi_k}(\pi') \\
\text{s.t.} \; & \mathop{\mathrm{E}}_{s \sim d^{\pi_k}} \left[D_{KL}(\pi' || \pi_k)[s] \right] \leq \delta
\end{aligned}
$$

Trust region Policy Optimization

TRPO= NPG +Linesearch+monotonic improvement theorem!

Algorithm 3 Trust Region Policy Optimization

Input: initial policy parameters θ_0

for $k = 0, 1, 2, \ldots$ **do**

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

Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm

Form sample estimates for

- policy gradient $\hat{g}_k$ (using advantage estimates)
- and KL-divergence Hessian-vector product function $f(v) = \hat{H}_k v$

Use CG with n_{cg} iterations to obtain $x_k \approx \hat{H}_k^{-1} \hat{g}_k$

Estimate proposed step $\Delta_k \approx \sqrt{\frac{2\delta}{x_k^T \hat{H}_k x_k}} x_k$

Perform backtracking line search with exponential decay to obtain final update

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

end for

Proximal Policy Optimization

Can I achieve similar performance without second order information (no Fisher matrix!)

- Adaptive KL Penalty
  - Policy update solves unconstrained optimization problem

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

  - Penalty coefficient β_k changes between iterations to approximately enforce KL-divergence constraint
- Clipped Objective
  - New objective function: let $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_k}(a_t|s_t)$. Then

$$\mathcal{L}^{CLIP}_{\theta_k}(\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_k}\left[\sum_{t=0}^{T}\left[\min(r_t(\theta)\hat{A}_t^{\pi_k}, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k})\right]\right]$$

    where ϵ is a hyperparameter (maybe $\epsilon = 0.2$)
  - Policy update is $\theta_{k+1} = \arg\max_\theta \mathcal{L}^{CLIP}_{\theta_k}(\theta)$

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. "Proximal Policy Optimization Algorithms". (2017)

PPO: Adaptive KL Penalty

Input: initial policy parameters θ_0, initial KL penalty β_0, target KL-divergence δ
for $k = 0, 1, 2, \ldots$ **do**
Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
Compute policy update

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

by taking K steps of minibatch SGD (via Adam)
if $\bar{D}_{KL}(\theta_{k+1} || \theta_k) \geq 1.5\delta$ **then**
$\beta_{k+1} = 2\beta_k$
else if $\bar{D}_{KL}(\theta_{k+1} || \theta_k) \leq \delta / 1.5$ **then**
$\beta_{k+1} = \beta_k / 2$
end if
end for

Don't use second order approximation for Kl which is expensive, use standard gradient descent

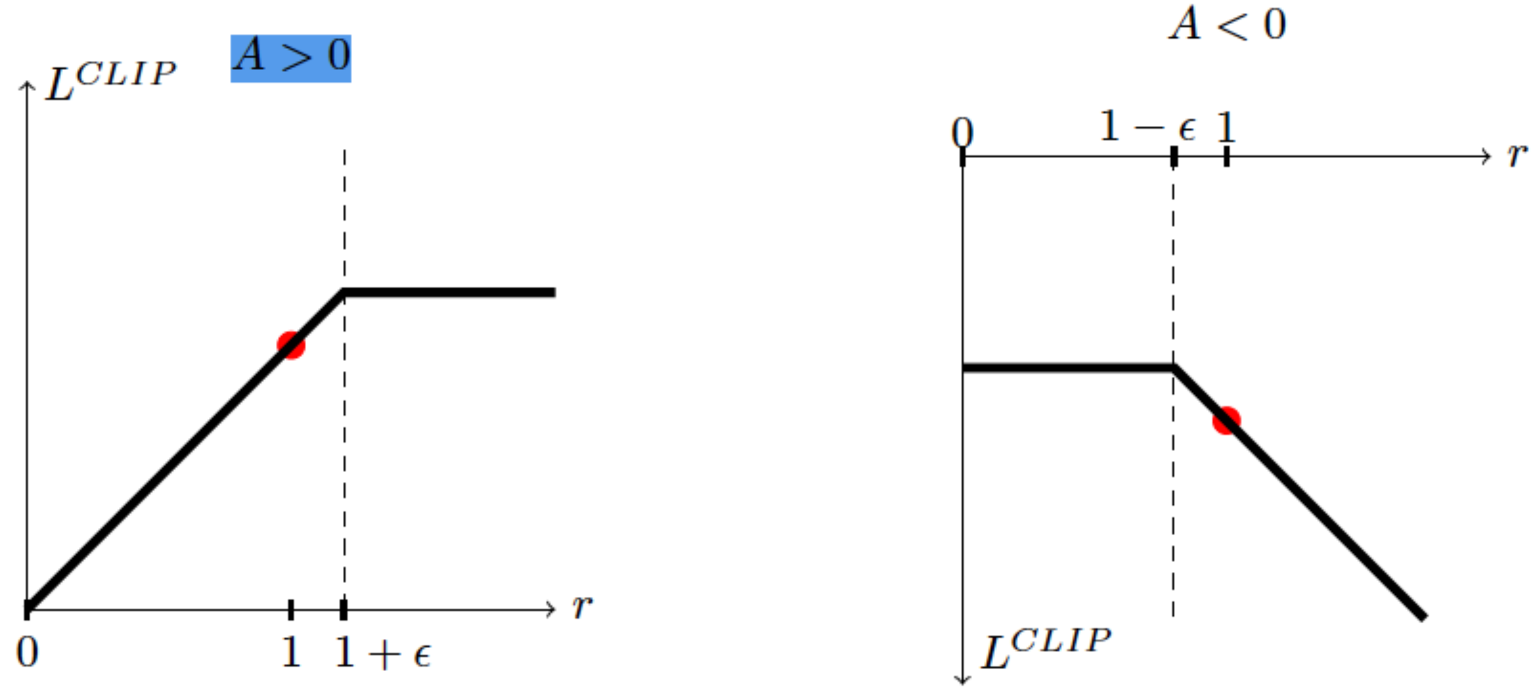
PPO: Clipped Objective

- Recall the surrogate objective

$$L^{IS}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right]. \quad (1)$$

- Form a lower bound via clipped importance ratios

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right] \quad (2)$$



J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. "Proximal Policy Optimization Algorithms". (2017)

PPO: Clipped Objective

But *how* does clipping keep policy close? By making objective as pessimistic as possible about performance far away from θ_k:

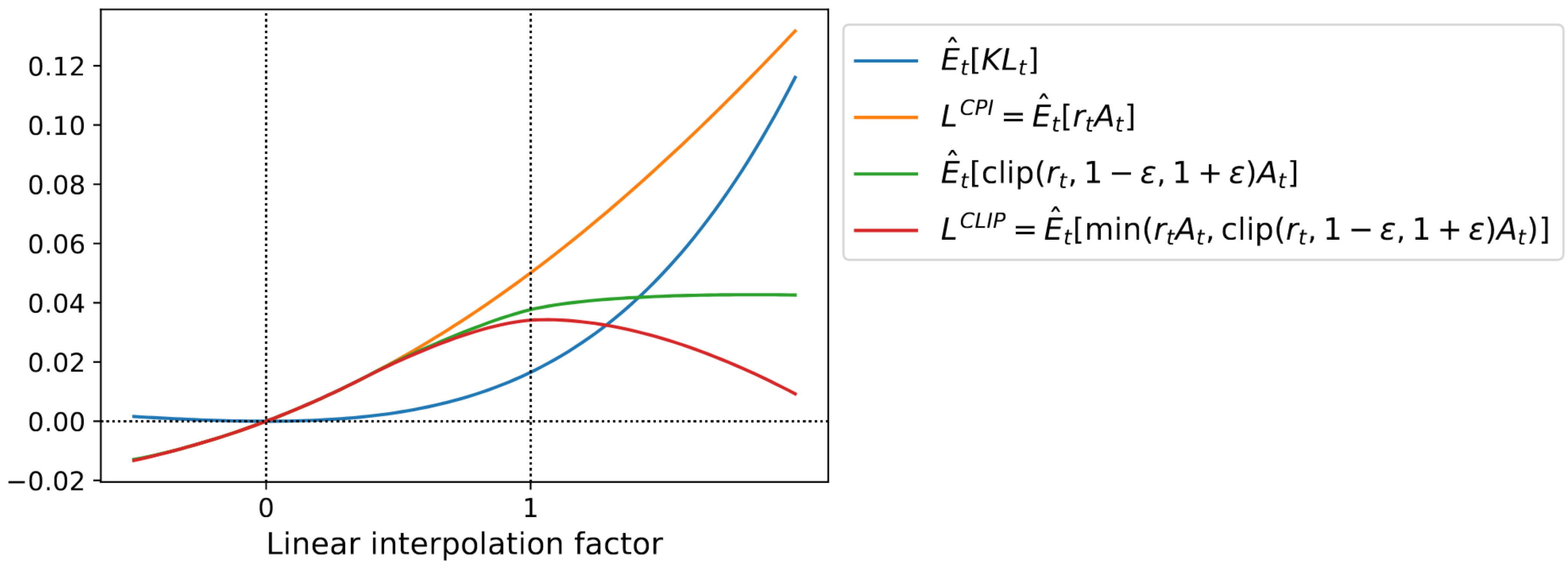


Figure: Various objectives as a function of interpolation factor α between θ_{k+1} and θ_k after one update of PPO-Clip [9]

PPO: Clipped Objective

Input: initial policy parameters θ_0, clipping threshold ϵ
for $k = 0, 1, 2, \ldots$ **do**
Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
Compute policy update

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

by taking K steps of minibatch SGD (via Adam), where

$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_k}\left[\sum_{t=0}^{T}\left[\min(r_t(\theta)\hat{A}_t^{\pi_k}, \mathrm{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k})\right]\right]$$

end for

- Clipping prevents policy from having incentive to go far away from θ_{k+1}
- Clipping seems to work at least as well as PPO with KL penalty, but is simpler to implement

PPO: Clipped Objective

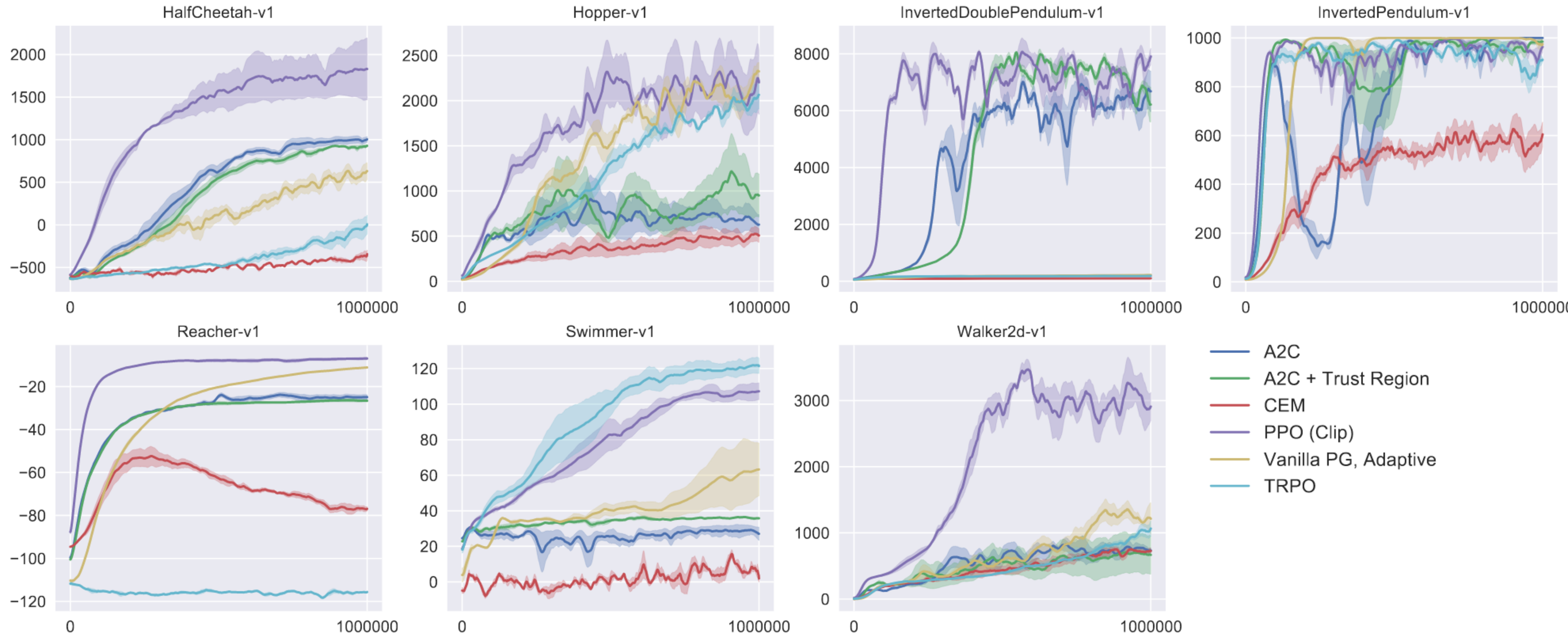


Figure: Performance comparison between PPO with clipped objective and various other deep RL methods on a slate of MuJoCo tasks. [10]

Summary

- Gradient Descent in Parameter VS distribution space
- Natural gradients: we need to keep track of how the KL changes from iteration to iteration
- Natural policy gradients
- Clipped objective works well

Related Readings

S. Kakade. "A Natural Policy Gradient." *NIPS*. 2001

S. Kakade and J. Langford. "Approximately optimal approximate reinforcement learning". *ICML*. 2002

J. Peters and S. Schaal. "Natural actor-critic". *Neurocomputing* (2008)

J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. "Trust Region Policy Optimization". *ICML* (2015)

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. "Proximal Policy Optimization Algorithms". (2017)

J. Achiam, D. Held, A. Tamar, P. Abeel "Constrained Policy Optimization". (2017)