Modern Policy Gradient Methods

Miguel Calvo-Fullana and Santiago Paternain
Electrical and Systems Engineering, University of Pennsylvania
{spater,cfullana}@seas.upenn.edu

November 21 —December 5, 2019

- A Markov Decision Process is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, P)$
- P is a Markov transition probability if for any $\mathcal{S}' \subseteq \mathcal{S}$ and $\mathcal{R}' \subseteq \mathcal{R}$

$$P[S_{t+1} \in \mathcal{S}', R_{t+1} \in \mathcal{R}' | S_t, A_t, \ldots, S_0, A_0] = P[S_{t+1} \in \mathcal{S}', R_{t+1} \in \mathcal{R}' | S_t, A_t]$$

- We select the actions based on parametrerized policies $\pi_\theta(a|s)$
  - $\Rightarrow$ We cannot work with general continuous functions
  - $\Rightarrow$ Parameterization is necessary
- Find the best policy within the functions that our parameterization defines
  - $\Rightarrow$ "Best" is defined by the value function

$$v_{\pi_\theta}(s) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \Big| S_0 = s\right]$$

  - $\Rightarrow$ Recall that the expectation is with respect to all the rewards seen
  - $\Rightarrow$ We write $\mathbb{E}_{\pi_\theta}$ to denote the we are following the policy $\pi_\theta(a|s)$

- Expanding the expectation, the value function is written as

$$v_{\pi_\theta}(s) = \sum_{k=0}^{\infty} \int_{\mathcal{R}^k \mathcal{A}^k \mathcal{S}^{k-1}} \gamma^k r_k \Pi_{j=0}^{k-1} p(s_{j+1}, r_{j+1} | s_j, a_j) \pi_\theta(a_j | s_j) \, d\mathbf{s}_k d\mathbf{a}_{k-1} d\mathbf{r}_k$$

- Where the policy is parameterized by $\theta \in \mathbb{R}^d$
- We will update the parameters via gradient ascent

  $\Rightarrow$ This gradient is given by the policy gradient theorem

- The gradient of v with respect to $\theta \in \mathbb{R}^d$ is given by

$$\nabla_\theta v(\theta) = \left(\frac{\partial v(\theta)}{\partial \theta_1}, \ldots, \frac{\partial v(\theta)}{\partial \theta_d} \right)^T$$

- To find the maximum, we update the parameters θ via

$$\theta_{k+1} = \theta_k + \alpha_k \nabla_\theta v(\theta), \text{ with } \alpha_k > 0$$

- Policy gradient for Episodic Tasks

$$\nabla_\theta v(\theta) = \mathbb{E}_\tau \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(A_t|S_t) G_t \right]$$

- Policy gradient for Continuing Tasks

$$\nabla_\theta v_{\pi_\theta}(s_i) = (1-\gamma)^{-1} \mathbb{E}_{S\sim\rho_\theta, A\sim\pi_\theta} \left[\nabla_\theta \log \pi_\theta(A|S) q_{\pi_\theta}(S,A) \right]$$

$$\rho_\theta(s_i, s') = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t P(S_0 = s_i \to S_t = s')$$

- There are several issues with the classical policy gradient
  - ⇒ We have attempted to address some of them
  - ⇒ Via baselines and actor-critic methods
- There are still issues that we have not tackled
- Modern methods for policy gradients attempt to tackle these issues
- We measure distance in the space of parameters
  - ⇒ Parameters that are close can generate very different distributions
  - ⇒ Especially problematic with deep neural networks
  - ⇒ Makes the choice of step size difficult
  - ⇒ We can attempt to characterize distances in the space of distributions
- We can be more efficient by reusing data from previous trajectories
  - ⇒ We have studied some of this for off-policy methods
  - ⇒ We can attempt to do so via importance sampling

- The Kullback-Leibler (KL) divergence is an asymmetric measure of difference between probability distributions
- For discrete probability distributions P and Q it is defined as

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

- Properties of the KL divergence
  - $\Rightarrow$ $D_{\mathrm{KL}}(P\|Q) \geq 0$
  - $\Rightarrow$ $D_{\mathrm{KL}}(P\|P) = 0$
  - $\Rightarrow$ $D_{\mathrm{KL}}(P\|Q) \neq D_{\mathrm{KL}}(Q\|P)$
- The KL divergence is not a metric as it is not symmetric
  - $\Rightarrow$ Nonetheless, it is a good measure of closeness between distributions

- Importance sampling is a technique for estimating a probability distribution
  - $\Rightarrow$ With only samples from a different distribution
- For distributions P and Q and an arbitrary function f we have that

$$\mathbb{E}_{x\sim P}\left[f(x)\right] = \mathbb{E}_{x\sim Q}\left[\frac{P(x)}{Q(x)}f(x)\right]$$

- We denote the ratio $P(x)/Q(x)$ as the importance sampling weight
- The variance of the stochastic approximation estimator is then given by

$$\begin{aligned} Var\left(\frac{P(x)}{Q(x)}f(x)\right) &= \mathbb{E}_{x\sim Q}\left[\left(\frac{P(x)}{Q(x)}f(x)\right)^2\right] - \mathbb{E}_{x\sim Q}\left[\frac{P(x)}{Q(x)}f(x)\right]^2 \\ &= \mathbb{E}_{x\sim P}\left[\frac{P(x)}{Q(x)}f(x)^2\right] - \mathbb{E}_{x\sim P}\left[f(x)\right]^2 \end{aligned}$$

- If the importance sampling weight $P(x)/Q(x)$ is large
  - $\Rightarrow$ The variance of the estimator can blow up

Example: Importance Sampling in the Policy Gradient

- Consider a trajectory $\tau = (S_0, A_0, S_1, R_1, A_1, S_2, R_2 \ldots, S_T, R_T)$
- Recall that the distribution of the trajectory is given by

$$p(\tau) = p(S_0) \prod_{t=0}^{T-1} \pi_\theta(A_t|S_t) p(S_{t+1}, R_{t+1}|S_t, A_t)$$

- Which depends on the policy π_θ with parametrization θ
- Further Recall that the policy gradient for finite horizon is given by

$$\nabla_\theta v(\theta) = \mathbb{E}_\tau \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(A_t|S_t) G_t \right]$$

- We want to reuse information from a different trajectory
  $\Rightarrow$ Trajectory τ' generated from a policy $\pi_{\theta'}$
- Via importance sampling we can write

$$\nabla_\theta v(\theta) = \mathbb{E}_{\tau'}\left[\frac{p(\tau)}{p(\tau')}\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(A_t|S_t)G_t\right]$$

- By taking a closer look at the ratio $p(\tau)/p(\tau')$ we have that

$$\frac{p(\tau)}{p(\tau')} = \frac{p(S_0)\prod_{t=0}^{T-1}\pi_\theta(A_t|S_t)p(S_{t+1},R_{t+1}|S_t,A_t)}{p(S_0)\prod_{t=0}^{T-1}\pi_{\theta'}(A_t|S_t)p(S_{t+1},R_{t+1}|S_t,A_t)} = \prod_{t=0}^{T-1}\frac{\pi_\theta(A_t|S_t)}{\pi_{\theta'}(A_t|S_t)}$$

- Small differences can multiply and become quite large
  $\Rightarrow$ Not the most stable way to reuse information from a different policy

- First step to tackle the previous issues
  - $\Rightarrow$ Characterize the difference between two policies
- Recall that the advantage is defined as the difference between q and v

$$a(S_t, A_t) = q(S_t, A_t) - v(S_t)$$

- It is a normalization with respect to the state
  - $\Rightarrow$ How much an action can improve over the value of the current state
  - $\Rightarrow$ Or the advantage of choosing a specific action
- Given policies π and π' and respective value functions $v(\pi)$ and $v(\pi')$

$$v(\pi') - v(\pi) = \mathbb{E}_{\pi'}\left[\sum_{t=0}^{\infty} \gamma^t a_\pi(S_t, A_t)\right]$$

- We relate the return of a policy in terms of the advantage over another
- We call this the relative policy performance identity

Relative Policy Performance Identity: Proof

- Note that the advantage is given by

$$a_\pi(S_t, A_t) = R_{t+1} + \gamma v_\pi(S_{t+1}) - v_\pi(S_t)$$

- Then we can rewrite the term on the right hand side

$$\begin{aligned}
\mathbb{E}_{\pi'}\left[\sum_{t=0}^{\infty} \gamma^t a_\pi(S_t, A_t)\right] &= \mathbb{E}_{\pi'}\left[\sum_{t=0}^{\infty} \gamma^t \Big(R_{t+1} + \gamma v_\pi(S_{t+1}) - v_\pi(S_t)\Big)\right] \\
&= v(\pi') + \mathbb{E}_{\pi'}\left[\sum_{t=0}^{\infty} \gamma^{t+1} v_\pi(S_{t+1}) - \sum_{t=0}^{\infty} \gamma^t v_\pi(S_t)\right] \\
&= v(\pi') - \mathbb{E}_{\pi'}\left[v_\pi(S_0)\right] \\
&= v(\pi') - v(\pi)
\end{aligned}$$

Finding a Policy π'

- We can use this identity to find a policy π' that maximizes $v(\pi')$

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

- By the relative policy performance identity this is equivalent to

$$\max_{\pi'} v(\pi') = \mathbb{E}_{\pi'}\left[\sum_{t=0}^{\infty} \gamma^t a_\pi(S_t, A_t)\right]$$

- This allows us to assess the quality of policy π' via the advantage of π
- This expression still depends on trajectories generated from the policy π'

  $\Rightarrow$ We want to remove this dependence

Dependence on Trajectories Generated From π'

- We try to remove the dependence on trajectories sampled from π'
- Let us define the discounted state distribution d_π as

$$d_\pi(s) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t P(S_t = s|\pi)$$

- Then we can rewrite the relative performance identity as

$$\begin{aligned} v(\pi') - v(\pi) &= \mathbb{E}_{\pi'}\left[\sum_{t=0}^{\infty} \gamma^t a_\pi(S_t, A_t)\right] \\ &= \frac{1}{1-\gamma}\mathbb{E}_{S\sim d_{\pi'}, A\sim\pi'}\left[a_\pi(S, A)\right] \end{aligned}$$

- We can remove the dependence $a \sim \pi'$ via importance sampling

$$v(\pi') - v(\pi) = \frac{1}{1-\gamma}\mathbb{E}_{S\sim d_{\pi'}, A\sim\pi}\left[\frac{\pi'(A|S)}{\pi(A|S)} a_\pi(S, A)\right]$$

- All that is left is to take care of the states $s \sim d_{\pi'}$

Relative Performance Bound

- How can we get rid of the dependency on $s \sim d_{\pi'}$?
- Let us assume for a second that $d_{\pi'} \approx d_\pi$
- We then have the local approximation $L_\pi(\pi')$ as follows

$$\begin{aligned} v(\pi') - v(\pi) &\approx \frac{1}{1-\gamma}\mathbb{E}_{S\sim d_\pi, A\sim\pi}\left[\frac{\pi'(A|S)}{\pi(A|S)} a_\pi(S,A)\right] \\ &\triangleq L_\pi(\pi') \end{aligned}$$

- Under these conditions, we have the following approximation guarantee

$$\left|v(\pi') - \left(v(\pi) + L_\pi(\pi')\right)\right| \leq C\sqrt{\mathbb{E}_{S\sim d_\pi}\left[D_{\mathrm{KL}}(\pi'\|\pi)\right]}$$

- Where C is a system-dependent constant
- The approximation is good if the policies are close in KL divergence

Using the Relative Performance Bound

- The local approximation is given by

$$L_\pi(\pi') = \frac{1}{1-\gamma}\mathbb{E}_{S\sim d_\pi, A\sim\pi}\left[\frac{\pi'(A|S)}{\pi(A|S)} a_\pi(S,A)\right]$$

- We can rewrite it as

$$L_\pi(\pi') = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t \frac{\pi'(A_t|S_t)}{\pi(A_t|S_t)} a_\pi(S_t,A_t)\right]$$

- We can use this approximation to optimize with respect to π'
  - $\Rightarrow$ Using only trajectories generated from π
  - $\Rightarrow$ Valid as long as policies are close in KL divergence
- Compared with the importance sampling of the policy gradient
  - $\Rightarrow$ The ratio $\pi'(A_t|S_t)/\pi(A_t|S_t)$ only appears per time instance
  - $\Rightarrow$ It is not multiplied over the whole trajectory

- We have another version of the relative performance bound

$$v(\pi') \geq L_\pi(\pi') - CD_{\mathrm{KL}}^{\max}(\pi' \| \pi)$$

- Where $C = \frac{4\epsilon\gamma}{(1-\gamma)^2}$ and $D_{\mathrm{KL}}^{\max}(\pi'\|\pi) = \max_s D_{\mathrm{KL}}(\pi'\|\pi)$
- If we maximize the right hand side of this bound

$$\pi_{k+1} = \underset{\pi}{\operatorname{argmax}} \left[L_{\pi_k}(\pi) - CD_{\mathrm{KL}}^{\max}(\pi \| \pi_k) \right]$$

- We are guaranteed to generate a monotonically improving sequence

$$v(\pi_0) \leq v(\pi_1) \leq v(\pi_2) \leq \cdots$$

- To see this let $M_k(\pi) = L_{\pi_k}(\pi) - CD_{\mathrm{KL}}^{\max}(\pi\|\pi_k)$ then

$$v(\pi_{k+1}) \geq M_k(\pi_{k+1})$$
$$v(\pi_k) = M_k(\pi_k)$$
$$v(\pi_{k+1}) - v(\pi_k) \geq M_k(\pi_{k+1}) - M_k(\pi_k)$$

Input: Policy π_θ
for *episode* $k = 0, 1, 2, \ldots$ **do**
  Generate an episode following $\pi_\theta : S_0, A_0, R_1, S_1, A_1, \ldots, S_{T-1}, A_{T-1}, R_T$
  Compute values of the advantage function $a_{\pi_k}(S_t, A_t)$
  Update policy according to

$$\pi_{k+1} = \operatorname*{argmax}_{\pi} \left[L_{\pi_k}(\pi) - C D_{\mathrm{KL}}^{\max}(\pi \| \pi_k) \right]$$

  Where $C = \frac{4\epsilon\gamma}{(1-\gamma)^2}$ and $D_{\mathrm{KL}}^{\max}(\pi' \| \pi) = \max_s D_{\mathrm{KL}}(\pi' \| \pi)$
end

Algorithm 1: Policy Iteration with Guaranteed Improvement

- The previous algorithm is a Majorization-Minimization (MM) algorithm
  - $\Rightarrow$ In our case a Minorization-Maximization
  - $\Rightarrow$ We construct a lower bound on the original objective function
  - $\Rightarrow$ We maximize this lower bound
- This approach guarantees monotonic improvement
- However in practice computing $D_{\text{KL}}^{\max}(\pi' \| \pi)$ is complicated
  - $\Rightarrow$ Requires evaluating the KL divergence at all states
  - $\Rightarrow$ Computationally expensive or simply impossible
- Substitute it for the average KL divergence over states

$$v(\pi') \geq L_\pi(\pi') - C\mathbb{E}_{S \sim d_\pi}\left[D_{\text{KL}}(\pi' \| \pi)\right]$$

Outline

Optimization Problem

- From now on we will consider parameterized policies π_θ
- From previously we have the following maximization

$$\theta_{k+1} = \underset{\theta}{\operatorname{argmax}} \, L_{\theta_k}(\theta) - C\mathbb{E}_{S\sim d_{\theta_k}}\left[D_{\mathrm{KL}}(\theta \| \theta_k)\right]$$

- The value of $C = \frac{4\epsilon\gamma}{(1-\gamma)^2}$ is given by the relative performance bound

  $\Rightarrow$ In practice this value is is too conservative

- Instead we propose to solve the following optimization problem

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & L_{\theta_k}(\theta) \\ \text{subject to} \quad & \mathbb{E}_{S\sim d_{\theta_k}}\left[D_{\mathrm{KL}}(\theta \| \theta_k)\right] \leq \delta \end{aligned}$$

- Where we have control over a parameter δ

  $\Rightarrow$ This is called a trust region method

  $\Rightarrow$ δ controls the extend of the region of trust

- The natural policy gradient is a special case of the previous problem
  - $\Rightarrow$ Linearly approximate the objective function $L_{\theta_k}(\theta)$
  - $\Rightarrow$ Quadratically approximate the constraint $\mathbb{E}_{S\sim d_{\theta_k}}\left[D_{\text{KL}}(\theta\|\theta_k)\right]$
- Done via Taylor's expansion resulting in the following approximations

$$L_{\theta_k}(\theta) \approx L_{\theta_k}(\theta_k) + g^T(\theta - \theta_k)$$

$$\mathbb{E}_{S\sim d_{\theta_k}}\left[D_{\text{KL}}(\theta\|\theta_k)\right] \approx \frac{1}{2}(\theta - \theta_k)^T H(\theta - \theta_k)^T$$

- Where $g \triangleq \nabla_\theta L_{\theta_k}(\theta_k)\mid_{\theta=\theta_k}$ and $H \triangleq \nabla_\theta^2 \mathbb{E}_{S\sim d_{\theta_k}}\left[D_{\text{KL}}(\theta\|\theta_k)\right]\mid_{\theta=\theta_k}$
- We can then write the approximate optimization problem

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & L_{\theta_k}(\theta_k) + g^T(\theta - \theta_k) \\ \text{subject to} \quad & \frac{1}{2}(\theta - \theta_k)^T H(\theta - \theta_k)^T \leq \delta \end{aligned}$$

- The gradient ascent update on which is given by

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

- The term $g \triangleq \nabla_\theta L_{\theta_k}(\theta_k) \mid_{\theta=\theta_k}$ corresponds to the policy gradient

$$\nabla_\theta L_{\theta_k}(\theta_k) \mid_{\theta=\theta_k} = \mathbb{E}_{\pi_{\theta_k}}\left[\sum_{t=0}^{\infty} \gamma^t \frac{\nabla_\theta \pi_\theta(A_t|S_t) \mid_{\theta=\theta_k}}{\pi_{\theta_k}(A_t|S_t)} a_{\pi_{\theta_k}}(S_t, A_t)\right]$$

$$= \mathbb{E}_{\pi_{\theta_k}}\left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi_\theta(A_t|S_t) \mid_{\theta=\theta_k} a_{\pi_{\theta_k}}(S_t, A_t)\right]$$

- The Hessian matrix $H \triangleq \nabla_\theta^2 \mathbb{E}_{S\sim d_{\theta_k}}\left[D_{\mathrm{KL}}(\theta \| \theta_k)\right] \mid_{\theta=\theta_k}$ is given by

$$H = \mathbb{E}_{\pi_{\theta_k}}\left[\nabla_\theta \log \pi_\theta(A|S) \mid_{\theta=\theta_k} \nabla_\theta \log \pi_\theta(A|S) \mid_{\theta=\theta_k}^T\right]$$

- Called the Fisher information matrix
- Important property is that $H^{-1}g$ is invariant to parametrization

Input: Policy π_θ, KL divergence target δ

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

Generate an episode following π_{θ_k} : $S_0, A_0, R_1, S_1, A_1, \ldots, S_{T-1}, A_{T-1}, R_T$

Estimate policy gradient g_k

$$g_k = \mathbb{E}_{\pi_{\theta_k}} \left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi_\theta(A_t|S_t) \mid_{\theta=\theta_k} a_{\pi_{\theta_k}}(S_t, A_t) \right]$$

Estimate Hessian H_k

$$H_k = \mathbb{E}_{\pi_{\theta_k}} \left[\nabla_\theta \log \pi_\theta(A|S) \mid_{\theta=\theta_k} \nabla_\theta \log \pi_\theta(A|S) \mid_{\theta=\theta_k}^T \right]$$

Compute natural policy gradient update

$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{g_k^T H_k^{-1} g_k}} H_k^{-1} g_k$$

end

Algorithm 2: Natural Policy Gradient

- The previous approach has a severe downfall
  - $\Rightarrow$ We need to compute the inverse H^{-1}
  - $\Rightarrow$ This can be problematic as it does not scale well
- Instead of finding H^{-1} use the Conjugate Gradient (CG) to compute $H^{-1}g$
- The conjugate gradient method can be used to solve for x in $Ax = b$
  - $\Rightarrow$ Widely used iterative method to solve linear system of equations
  - $\Rightarrow$ Finds it via projections to the Krylov subspace $\text{span}\{b, Ab, A^2b, \ldots\}$
- Conjugate gradient only needs to evaluate Hessian-vector products

- We want to ensure that the KL divergence constraint is satisfied
  - $\Rightarrow$ KL might not be satisfied due to the quadratic approximation
  - $\Rightarrow$ At some iterations, the trust region given by δ can be too large
- We include a line search step
  - $\Rightarrow$ Backtracking line search with exponential decay
  - $\Rightarrow$ Enforce improvement in the approximation $L_{\theta_k}(\theta) \geq 0$
  - $\Rightarrow$ Ensure the KL divergence constraint $\mathbb{E}_{S \sim d_{\pi_{\theta_k}}}\left[D_{\mathrm{KL}}(\theta \| \theta_k)\right] \leq \delta$ is met
- Trust Region Policy Optimization (TRPO) consists of
  - $\Rightarrow$ The natural policy gradient
  - $\Rightarrow$ Efficient Hessian-vector product computation via conjugate gradient
  - $\Rightarrow$ A line search to enforce the KL divergence

Input: Policy π_θ, KL divergence target δ
for *episode* $k = 0, 1, 2, \ldots$ **do**

Generate an episode following $\pi_{\theta_k}: S_0, A_0, R_1, S_1, A_1, \ldots, S_{T-1}, A_{T-1}, R_T$

Estimate policy gradient g_k

$$g_k = \mathbb{E}_{\pi_{\theta_k}}\left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi_\theta(A_t|S_t)\,|_{\theta=\theta_k}\, a_{\pi_{\theta_k}}(S_t, A_t)\right]$$

Use Conjugate Gradient to estimate $x_k = H_k^{-1} g_k$

Conduct a line search with the proposed update

$$\Delta_k = \sqrt{\frac{2\delta}{g_k^T x_k}}\, x_k$$

Update parameters after line search

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

end

Algorithm 3: Trust Region Policy Optimization (TRPO)

Input: Update step $\Delta_k = \sqrt{\frac{2\delta}{g_k^T H_k^{-1} g_k}} H_k^{-1} g_k$, step size decay $\alpha \in (0, 1)$

for $j = 0, 1, 2, \ldots, L$ *steps* **do**

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

  if $L_{\theta_k}(\theta) \geq 0$ *and* $\mathbb{E}_{s \sim d_{\pi_{\theta_k}}} \left[D_{KL}(\theta \| \theta_k) \right] \leq \delta$ **then**

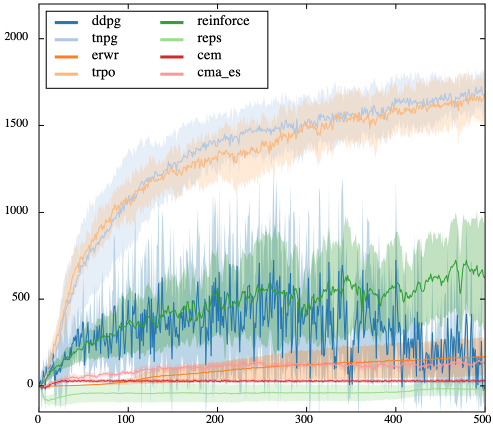
    Set $\theta_{k+1} = \theta_k + \alpha^j \Delta_k$

    Stop

  end

end

Algorithm 4: Line Search for TRPO



- Comparison of various RL methods on a MuJoCo walker task.[1]

[1] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel "Benchmarking Deep Reinforcement Learning for Continuous Control", in *International Conference on Machine Learning*, pp. 1329-1338, 2016.

Outline

The Issues With Natural Policy Gradients

- Methods based on natural gradients are computationally expensive
- Even with the modifications made to TRPO it is still expensive
  - $\Rightarrow$ They are fundamentally second order methods
- Can we obtain similar performance with a first-order method?
  - $\Rightarrow$ Avoid Hessian-related computations
- Proximal Policy Optimization (PPO) attempt to do so
  - $\Rightarrow$ It is a first order approach based on heuristics
  - $\Rightarrow$ Matches or surpasses TRPO performance in practice
- Proximal Policy Optimization has two variants
  - $\Rightarrow$ Proximal Policy Optimization with KL penalty
  - $\Rightarrow$ Proximal Policy Optimization with clipped objective

- Recall the previous unconstrained optimization problem

$$\theta_{k+1} = \underset{\theta}{\operatorname{argmax}}\, L_{\theta_k}(\theta) - \beta_k \mathbb{E}_{S \sim d_{\theta_k}} \left[D_{\mathrm{KL}}(\theta \| \theta_k) \right]$$

- Choosing a fixed value of β_k is complicated
- PPO with KL penalty attempts to adjust the value of β_k dynamically
  - $\Rightarrow$ This is done via a heuristic check at each iteration
- The policy adapts the value β_k
  - $\Rightarrow$ Individual iterations can violate KL constraints as β_k adapts
  - $\Rightarrow$ This is not much of an issue in practice as adaptation occurs fast

Input: Policy π_θ, KL divergence target δ

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

  Generate an episode following $\pi_\theta : S_0, A_0, R_1, S_1, A_1, \ldots, S_{T-1}, A_{T-1}, R_T$

  Update policy according to

$$\theta_{k+1} = \operatorname*{argmax}_{\theta} L_{\theta_k}(\theta) - \beta_k \mathbb{E}_{S \sim d_{\theta_k}} \left[D_{\mathrm{KL}}(\theta \| \theta_k) \right]$$

  if $\mathbb{E}_{S \sim d_{\theta_k}} \left[D_{KL}(\theta_{k+1} \| \theta_k) \right] \geq 1.5\delta$ **then**

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

  end

  if $\mathbb{E}_{S \sim d_{\theta_k}} \left[D_{KL}(\theta_{k+1} \| \theta_k) \right] \leq \delta / 1.5$ **then**

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

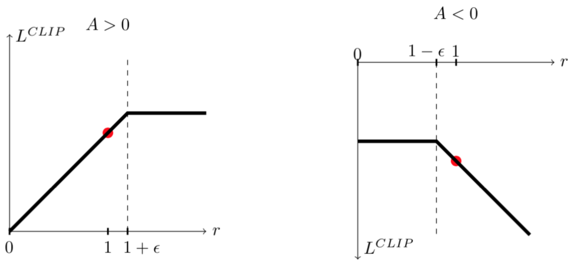
  end

end

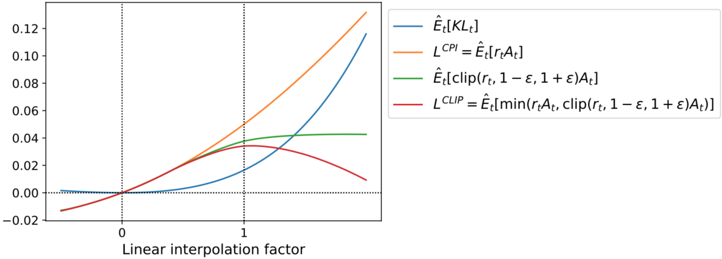
Algorithm 5: PPO with KL Penalty

- Clipping PPO modifies the objective to penalize policies moving far away
  $\Rightarrow$ Extremely simple to implement
- The update is given by the maximization of the objective function

$$\theta_{k+1} = \underset{\theta}{\text{argmax}} \left[\mathbb{E}_{\pi_{\theta_k}} \left[\sum_{t=0}^{\infty} \min \left(\frac{\pi_\theta(A_t|S_t)}{\pi_{\theta_k}(A_t|S_t)} a_{\pi_{\theta_k}}(S_t, A_t), \right.\right.\right.$$
$$\left.\left.\left. \text{clip}\left(\frac{\pi_\theta(A_t|S_t)}{\pi_{\theta_k}(A_t|S_t)}, 1-\epsilon, 1+\epsilon \right) a_{\pi_{\theta_k}}(S_t, A_t) \right) \right] \right]$$



Different Clipping Objectives



- Objective function vs interpolation factor between θ_k and θ_{k+1}
- The objective is penalized as the policy moves away from θ_k

Input: Policy π_θ, Clipping value ϵ

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

Generate an episode following π_{θ_k} : $S_0, A_0, R_1, S_1, A_1, \ldots, S_{T-1}, A_{T-1}, R_T$

Update policy according to

$$\theta_{k+1} = \underset{\theta}{\operatorname{argmax}}\, L^{\text{CLIP}}_{\theta_k}(\theta)$$

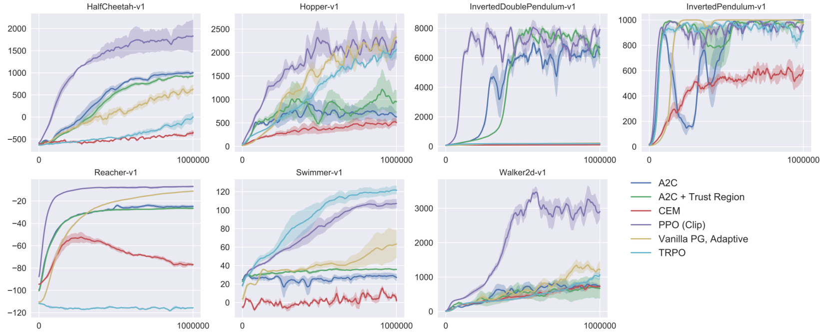
Where $L^{\text{CLIP}}_{\theta_k}$ is given by

$$L^{\text{CLIP}}_{\theta_k} = \sum_{t=0}^{T-1} \min\left(\frac{\pi_\theta(A_t|S_t)}{\pi_{\theta_k}(A_t|S_t)} a_{\pi_{\theta_k}}(S_t, A_t), \text{clip}\left(\frac{\pi_\theta(A_t|S_t)}{\pi_{\theta_k}(A_t|S_t)}, 1-\epsilon, 1+\epsilon\right) a_{\pi_{\theta_k}}(S_t, A_t)\right)$$

end

Algorithm 6: PPO with Clipped Objective

Performance of PPO



- Performance of PPO on various MuJoCo tasks.[2]

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. "Proximal policy optimization algorithms", *arXiv preprint arXiv:1707.06347*, 2017.

- These methods are based on the KL divergence between policies
  - $\Rightarrow$ This is an asymmetric measure and hence not a metric
- TRPO provides a fundamentally solid approach
  - $\Rightarrow$ However it is computationally expensive
- PPO is based on intuitive heuristics
  - $\Rightarrow$ In practice works surprisingly well
- Still, there is an important question that we have not answered
  - $\Rightarrow$ What is the right measure of similarity between two policies?
- We should use a true metric (Wasserstein distance)
  - $\Rightarrow$ Wasserstein Reinforcement Learning