# Lecture 14
# Deterministic Policy Gradient
# TRPO, PPO

# Recall: Policy Approximation

$$\pi(a|s, \boldsymbol{\theta})$$

We want to learn this directly!

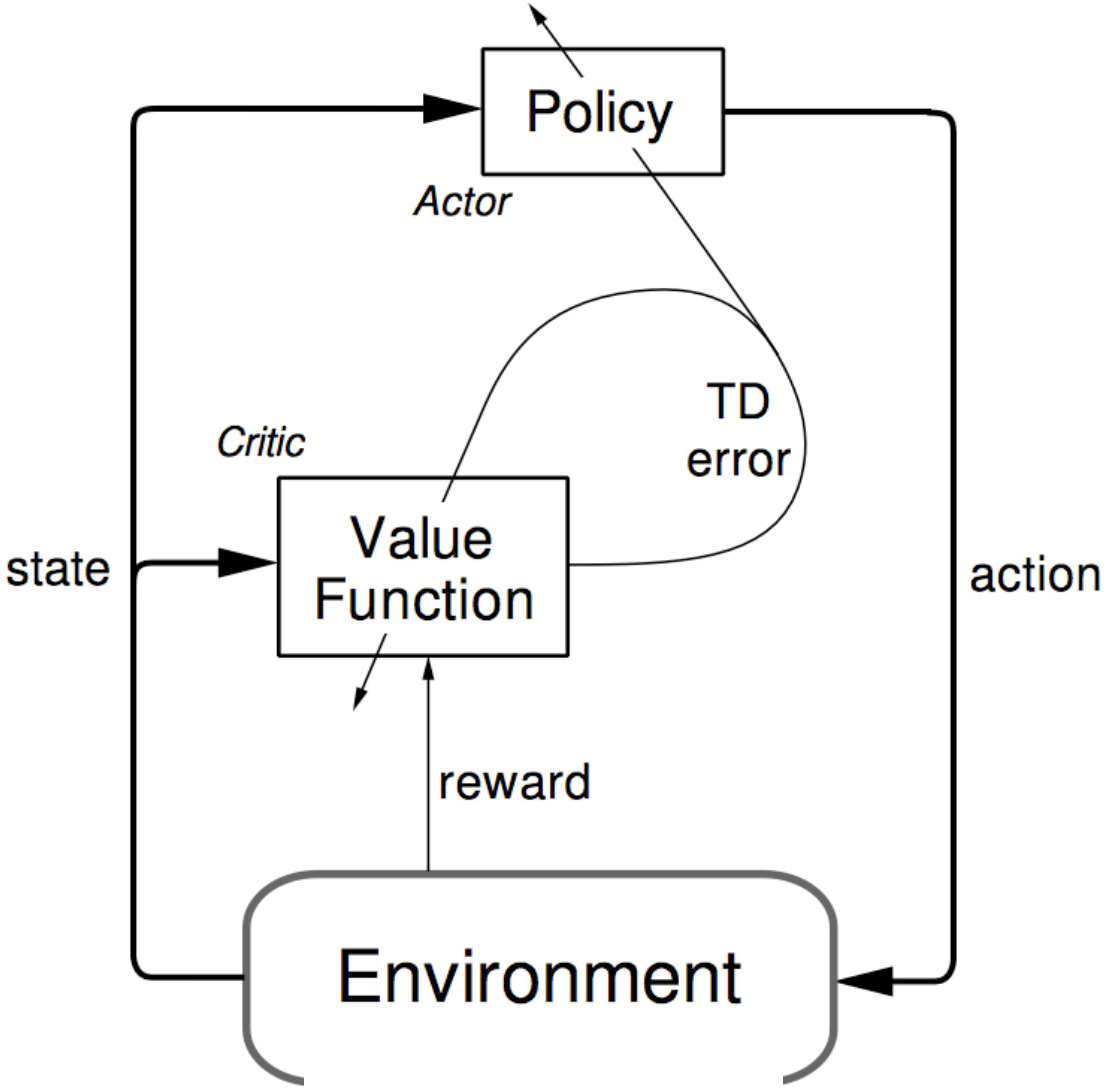
# Recall: Actor-Critic Algorithms

- ACTOR: policy $\pi$
- CRITIC: value fct V (or Q)

**Policy Gradient Theorem:**

$$\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\pi) = \mathbb{E}_{\pi}[\sum_{t=0}^{T} (\gamma^t G_t) \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t)]$$

# Actor-critic architecture



# Recall: Actor-Critic Algorithms

- ACTOR: policy $\pi$
- CRITIC: value fct V (or Q)

**Policy Gradient Theorem:**

$$\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\pi) = \mathbb{E}_{\pi}[\sum_{t=0}^{T} (\gamma^t G_t) \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t)]$$

**REINFORCE Estimates G with Monte-Carlo**

# Recall: Actor-Critic Algorithms

- ACTOR: policy $\pi$
- CRITIC: value fct V (or Q)

**Policy Gradient Theorem:**

$$\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\pi) = \mathbb{E}_{\pi}[\sum_{t=0}^{T} (\gamma^t G_t) \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t)]$$

**Actor-Critic: use V and/or Q to estimate G, e.g. TD(0)**

# Recall: Actor-Critic 1-step TD / TD(0) estimate:

**Policy Gradient Theorem:**

$$\nabla_\theta J_\theta(\pi) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t \underbrace{\left(q_\pi(S_t, A_t) - v_\pi(S_t)\right)}_{\text{Advantage}} \nabla_\theta \log(\pi)\right]$$

# What about if we want a Deterministic Policy?

We can't use the Policy Gradient Theorem :

$$\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\pi) = \mathbb{E}_{\pi}[\sum_{t=0}^{T} \left(\gamma^t G_t\right) \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t)]$$

How can we estimate $\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\pi)$ ?  When $A = \pi(S, \theta)$

# What about if we want a Deterministic Policy?

We can't use the Policy Gradient Theorem :

$$\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\pi) = \mathbb{E}_{\pi}[\sum_{t=0}^{T} \left(\gamma^t G_t\right) \nabla_{\boldsymbol{\theta}} \log \pi(A_t | S_t)]$$

How can we estimate $\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\pi)$ ?  When $A = \pi(S, \theta)$

$$J_{\theta}(\pi | S_0 = S) = q_{\pi}(\pi(S), S) \approx Q_{\pi}(\pi(S, \theta), S)$$

# What about if we want a Deterministic Policy?

How can we estimate $\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\pi)$ ?  When $A = \pi(S, \theta)$

$$A = (a_1, \ldots, a_m),\ \pi = (\pi_1, \ldots, \pi_m)$$

$$J_\theta(\pi \,|\, S_0 = S) = q_\pi(\pi(S), S) \approx Q_\pi(\pi(S, \theta), S)$$

# What about if we want a Deterministic Policy?

How can we estimate $\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\pi)$ ? When $A = \pi(S, \theta)$

$A = (a_1, \ldots, a_m),\ \pi = (\pi_1, \ldots, \pi_m)$

$$J_\theta(\pi \,|\, S_0 = S) = q_\pi(\pi(S), S) \approx Q_\pi(\pi(S, \theta), S)$$

$$\nabla_\theta J_\theta(\pi \,|\, S_0 = S) \approx \nabla_\theta Q_\pi(\pi(S, \theta), S)$$

# What about if we want a Deterministic Policy?

How can we estimate $\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\pi)$ ? When $A = \pi(S, \theta)$

$$A = (a_1, \ldots, a_m),\ \pi = (\pi_1, \ldots, \pi_m)$$

$$J_\theta(\pi \,|\, S_0 = S) = q_\pi(\pi(S), S) \approx Q_\pi(\pi(S, \theta), S)$$

$$\nabla_\theta J_\theta(\pi \,|\, S_0 = S) \approx \nabla_\theta Q_\pi(\pi(S, \theta), S)$$

$$= \sum_i^m \frac{\partial Q_\pi(A = \pi(S, \theta), S)}{\partial a_i} \nabla_\theta \pi_i(S, \theta)$$

# What about if we want a Deterministic Policy?

How can we estimate $\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\pi)$ ? When $A = \pi(S, \theta)$

$A = (a_1, \ldots, a_m),\ \pi = (\pi_1, \ldots, \pi_m)$

$$J_\theta(\pi \,|\, S_0 = S) = q_\pi(\pi(S), S) \approx Q_\pi(\pi(S, \theta), S)$$

$$\nabla_\theta J_\theta(\pi \,|\, S_0 = S) \approx \nabla_\theta Q_\pi(\pi(S, \theta), S)$$

$$= \sum_i^m \frac{\partial Q_\pi(A = \pi(S, \theta), S)}{\partial a_i} \nabla_\theta \pi_i(S, \theta)$$

$$= \nabla_A Q_\pi \left(A = \pi(S, \theta), S\right) \nabla_\theta \pi(S, \theta)$$

# Deterministic Policy Gradient:

How can we estimate $\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\pi)$ ? When $A = \pi(S, \theta)$

$$A = (a_1, \ldots, a_m),\ \pi = (\pi_1, \ldots, \pi_m)$$

$$\nabla_\theta J_\theta(\pi \,|\, S_0 = S) \approx \sum_i^m \frac{\partial Q_\pi(A = \pi(S,\theta), S)}{\partial a_i} \nabla_\theta \pi_i(S, \theta)$$

$$= \nabla_A Q_\pi \left(A = \pi(S,\theta), S\right) \nabla_\theta \pi(S, \theta)$$

http://proceedings.mlr.press/v32/silver14.pdf

# Deterministic Policy Gradient (on Continuous Control Tasks):

Actor Critic with stochastic policy

Deterministic Policy Gradient

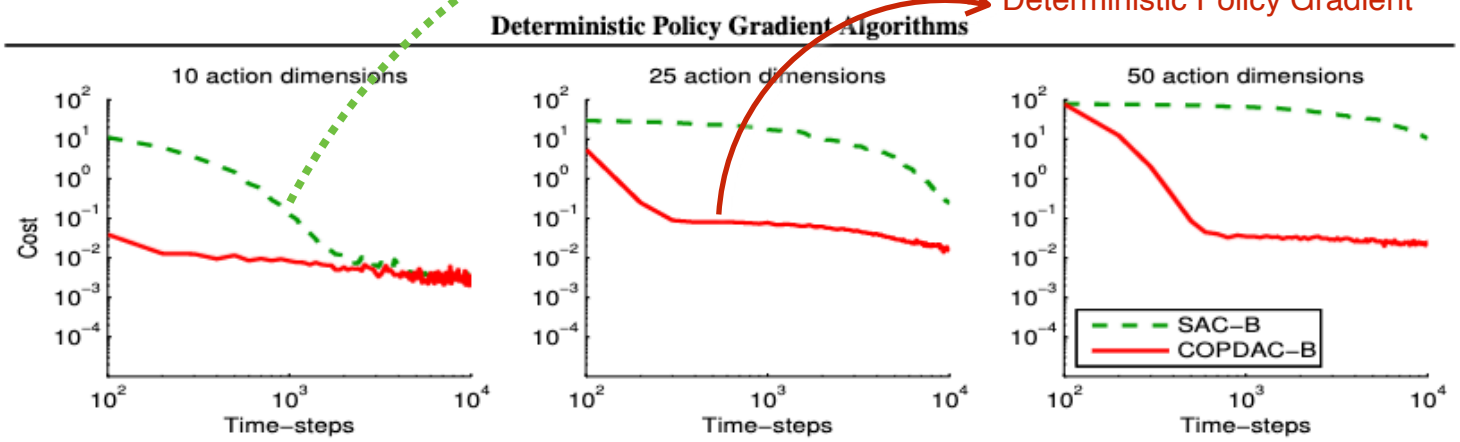
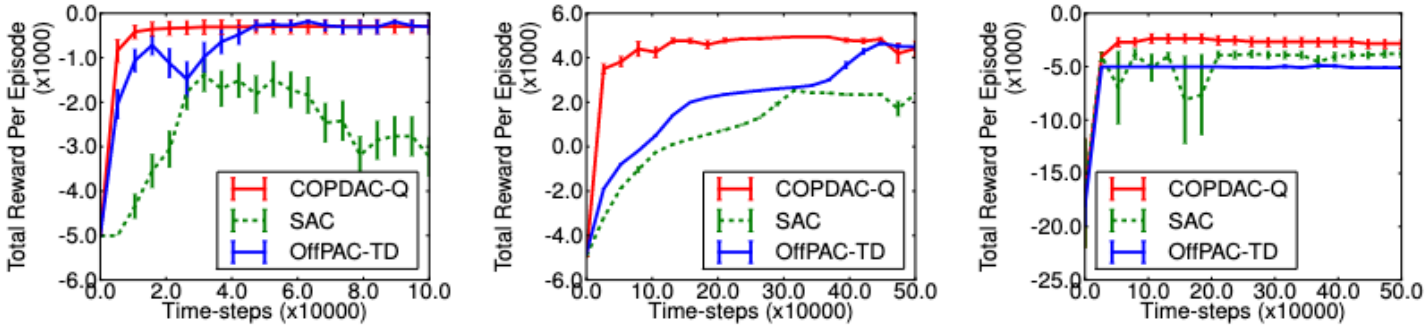



*Figure 1.* Comparison of stochastic actor-critic (SAC-B) and deterministic actor-critic (COPDAC-B) on the continuous bandit task.




(a) Mountain Car (b) Pendulum (c) 2D Puddle World

*Figure 2.* Comparison of stochastic on-policy actor-critic (SAC), stochastic off-policy actor-critic (OffPAC), and deterministic off-policy actor-critic (COPDAC) on continuous-action reinforcement learning. Each point is the average test performance of the mean policy.

http://proceedings.mlr.press/v32/silver14.pdf

# Deep Deterministic Policy Gradient (DDPG):

**Algorithm 1** DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$.
Initialize target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer $R$
**for** episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state $s_1$
  **for** t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action $a_t$ and observe reward $r_t$ and observe new state $s_{t+1}$
    Store transition $(s_t, a_t, r_t, s_{t+1})$ in $R$
    Sample a random minibatch of $N$ transitions $(s_i, a_i, r_i, s_{i+1})$ from $R$
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  **end for**
**end for**

https://arxiv.org/pdf/1509.02971.pdf

# Deep Deterministic Policy Gradient (DDPG)

- Incorporate replay buffer and target network ideas from DQN for increased stability
- Use lagged (Polyak-averaging) version of $Q_\phi$ and $\pi_\theta$ for fitting $Q_\phi$ (towards $Q^{\pi,\gamma}$) with TD(0)

$$\hat{Q}_t = r_t + \gamma Q_{\phi'}(s_{t+1}, \pi(s_{t+1}; \theta'))$$

- Pseudocode:

**for** iteration$=1, 2, \ldots$ **do**
  Act for several timesteps, add data to replay buffer
  Sample minibatch
  Update $\pi_\theta$ using $g \propto \nabla_\theta \sum_{t=1}^{T} Q(s_t, \pi(s_t, z_t; \theta))$
  Update $Q_\phi$ using $g \propto \nabla_\phi \sum_{t=1}^{T} (Q_\phi(s_t, a_t) - \hat{Q}_t)^2$,
**end for**

Applied to 2D and 3D robotics tasks and driving with pixel input

# TRPO -> PPO

- Trust Region Policy Optimization: https://arxiv.org/pdf/1502.05477.pdf
- Proximal Policy Optimization: https://arxiv.org/pdf/1707.06347.pdf

# Notation

$$\eta(\pi) = \mathbb{E}_{s_0, a_0, \ldots}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right], \text{ where}$$

$$s_0 \sim \rho_0(s_0),\ a_t \sim \pi(a_t|s_t),\ s_{t+1} \sim P(s_{t+1}|s_t, a_t).$$

$$Q_\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}, a_{t+1}, \ldots}\left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l})\right],$$

$$V_\pi(s_t) = \mathbb{E}_{a_t, s_{t+1}, \ldots}\left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l})\right],$$

$$A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s), \text{ where}$$

$$a_t \sim \pi(a_t|s_t), s_{t+1} \sim P(s_{t+1}|s_t, a_t) \text{ for } t \geq 0.$$

# TRPO:

$$\eta(\pi) = \mathbb{E}_{s_0, a_0, \ldots}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right], \text{ where}$$

$$s_0 \sim \rho_0(s_0), \; a_t \sim \pi(a_t|s_t), \; s_{t+1} \sim P(s_{t+1}|s_t, a_t).$$

The following useful identity expresses the expected return of another policy $\tilde{\pi}$ in terms of the advantage over $\pi$, accumulated over timesteps (see Kakade & Langford (2002) or Appendix A for proof):

$$\eta(\tilde{\pi}) = \eta(\pi) + \mathbb{E}_{s_0, a_0, \cdots \sim \tilde{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t)\right] \qquad (1)$$

# TRPO:

$$\eta(\pi) = \mathbb{E}_{s_0, a_0, \ldots}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right], \text{ where}$$

$$s_0 \sim \rho_0(s_0),\ a_t \sim \pi(a_t|s_t),\ s_{t+1} \sim P(s_{t+1}|s_t, a_t).$$

$$\eta(\tilde{\pi}) = \eta(\pi) + \mathbb{E}_{s_0, a_0, \cdots \sim \tilde{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t)\right] \qquad (1)$$

$$L_\pi(\tilde{\pi}) = \eta(\pi) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A_\pi(s, a). \qquad (3)$$

# TRPO:

$$\eta(\pi) = \mathbb{E}_{s_0, a_0, \ldots}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right], \text{ where}$$

$$s_0 \sim \rho_0(s_0), \; a_t \sim \pi(a_t|s_t), \; s_{t+1} \sim P(s_{t+1}|s_t, a_t).$$

$$\eta(\tilde{\pi}) = \eta(\pi) + \mathbb{E}_{s_0, a_0, \cdots \sim \tilde{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t)\right] \qquad (1)$$

This is an approximation!

$$L_\pi(\tilde{\pi}) = \eta(\pi) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A_\pi(s, a). \qquad (3)$$

# TRPO:

$$\eta(\pi) = \mathbb{E}_{s_0, a_0, \ldots}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right], \text{ where}$$

$$s_0 \sim \rho_0(s_0),\ a_t \sim \pi(a_t|s_t),\ s_{t+1} \sim P(s_{t+1}|s_t, a_t).$$

$$\eta(\tilde{\pi}) = \eta(\pi) + \mathbb{E}_{s_0, a_0, \cdots \sim \tilde{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t)\right] \quad (1)$$

This is an approximation!
Should be $\rho_{\tilde{\pi}}$ instead of $\rho_\pi$

$$L_\pi(\tilde{\pi}) = \eta(\pi) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A_\pi(s, a). \quad (3)$$

TRPO: $$\eta(\pi) = \mathbb{E}_{s_0, a_0, \ldots}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right], \text{ where}$$

$$s_0 \sim \rho_0(s_0), \; a_t \sim \pi(a_t|s_t), \; s_{t+1} \sim P(s_{t+1}|s_t, a_t).$$

This is an approximation!
Should be $\rho_{\tilde{\pi}}$ instead of $\rho_{\pi}$

$$L_\pi(\tilde{\pi}) = \eta(\pi) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A_\pi(s, a). \quad (3)$$

TRPO says: Approximately valid only if $\tilde{\pi} \sim \pi$

TRPO:

TRPO says: Approximately valid only if $\tilde{\pi} \sim \pi$

$$\sum_a \pi_\theta(a|s_n) A_{\theta_{\text{old}}}(s_n, a) = \mathbb{E}_{a \sim q}\left[\frac{\pi_\theta(a|s_n)}{q(a|s_n)} A_{\theta_{\text{old}}}(s_n, a)\right]$$

Our optimization problem in Equation (13) is exactly equivalent to the following one, written in terms of expectations:

$$\begin{aligned} &\underset{\theta}{\text{maximize}}\, \mathbb{E}_{s \sim \rho_{\theta_{\text{old}}}, a \sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)} Q_{\theta_{\text{old}}}(s, a)\right] \qquad (14) \\ &\text{subject to } \mathbb{E}_{s \sim \rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot|s) \parallel \pi_\theta(\cdot|s))\right] \leq \delta. \end{aligned}$$

# TRPO:

**Algorithm 1** Policy iteration algorithm guaranteeing non-decreasing expected return $\eta$

Initialize $\pi_0$.

**for** $i = 0, 1, 2, \ldots$ until convergence **do**

Compute all advantage values $A_{\pi_i}(s, a)$.

Solve the constrained optimization problem

$$\pi_{i+1} = \arg\max_{\pi} \left[ L_{\pi_i}(\pi) - C D_{\mathrm{KL}}^{\max}(\pi_i, \pi) \right]$$

where $C = 4\epsilon\gamma/(1-\gamma)^2$

and $L_{\pi_i}(\pi) = \eta(\pi_i) + \sum_s \rho_{\pi_i}(s) \sum_a \pi(a|s) A_{\pi_i}(s, a)$

**end for**

# Proximal Policy Gradient (PPO)

- Use penalty instead of constraint

$$\underset{\theta}{\text{minimize}} \sum_{n=1}^{N} \frac{\pi_\theta(a_n \mid s_n)}{\pi_{\theta_{\text{old}}}(a_n \mid s_n)} \hat{A}_n - \beta \overline{\text{KL}}[\pi_{\theta_{\text{old}}}, \pi_\theta]$$

- Pseudocode:

  **for** iteration=$1, 2, \ldots$ **do**
    Run policy for $T$ timesteps or $N$ trajectories
    Estimate advantage function at all timesteps
    Do SGD on above objective for some number of epochs
    If KL too high, increase $\beta$. If KL too low, decrease $\beta$.
  **end for**

- $\approx$ same performance as TRPO, but only first-order optimization

PPO:

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

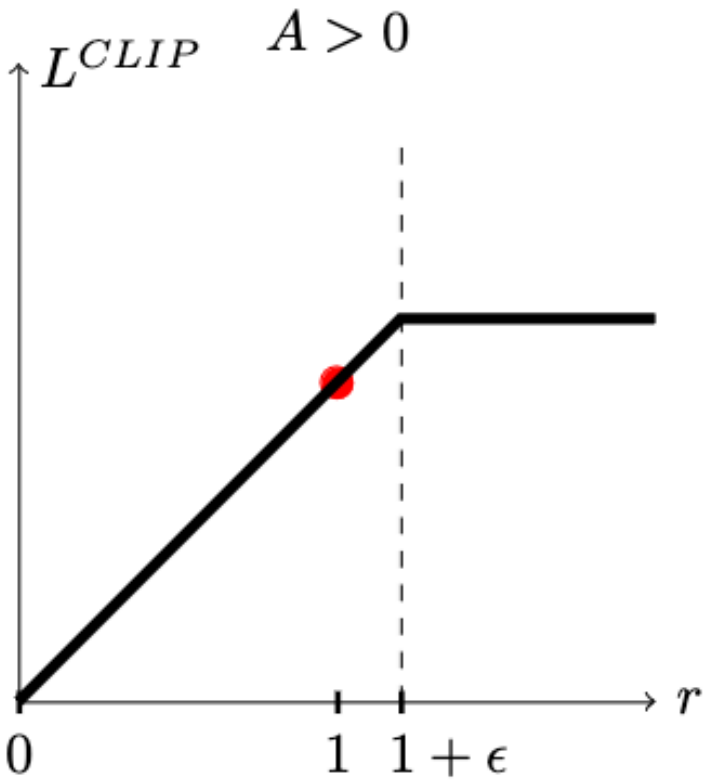
PPO:

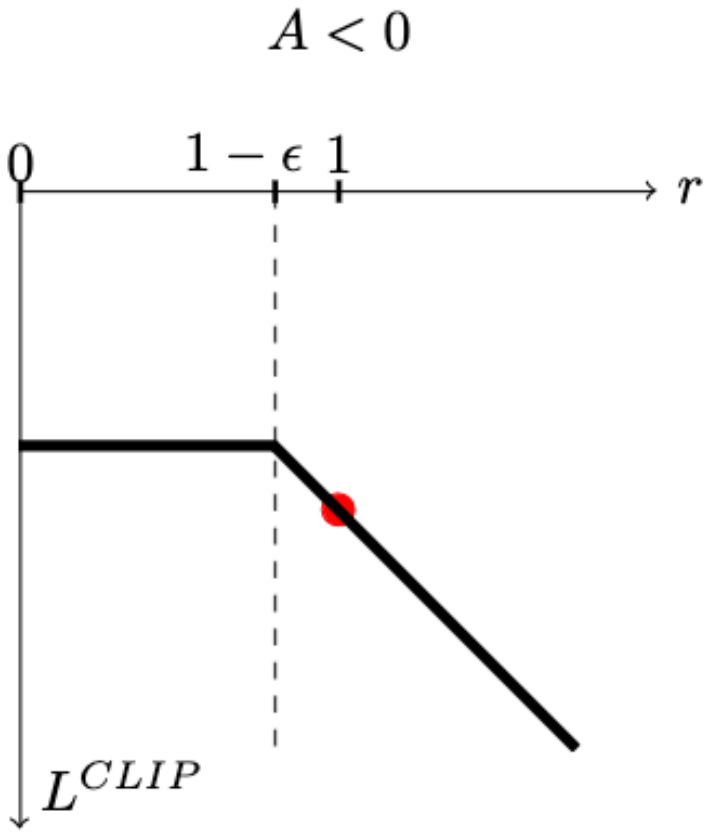
$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right].$$

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

# PPO:

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



# PPO:

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

**Algorithm 1** PPO, Actor-Critic Style

**for** iteration=$1, 2, \ldots$ **do**
  **for** actor=$1, 2, \ldots, N$ **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for $T$ timesteps
    Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
  **end for**
  Optimize surrogate $L$ wrt $\theta$, with $K$ epochs and minibatch size $M \leq NT$
  $\theta_{\text{old}} \leftarrow \theta$
**end for**

# PPO:

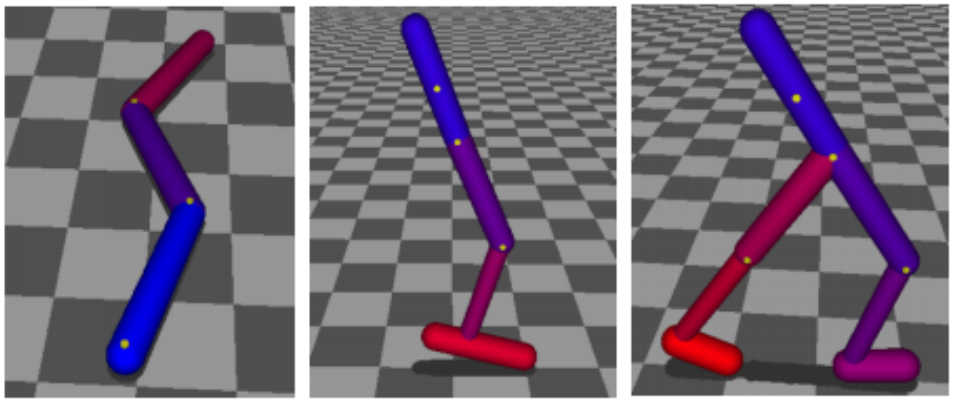
$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

| algorithm | avg. normalized score |
|---|---|
| No clipping or penalty | -0.39 |
| Clipping, $\epsilon = 0.1$ | 0.76 |
| **Clipping, $\epsilon = 0.2$** | **0.82** |
| Clipping, $\epsilon = 0.3$ | 0.70 |
| Adaptive KL $d_{\text{targ}} = 0.003$ | 0.68 |
| Adaptive KL $d_{\text{targ}} = 0.01$ | 0.74 |
| Adaptive KL $d_{\text{targ}} = 0.03$ | 0.71 |
| Fixed KL, $\beta = 0.3$ | 0.62 |
| Fixed KL, $\beta = 1.$ | 0.71 |
| Fixed KL, $\beta = 3.$ | 0.72 |
| Fixed KL, $\beta = 10.$ | 0.69 |

# Results

Applied to

- Locomotion controllers in 2D



- Atari games with pixel input

# PPO:

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

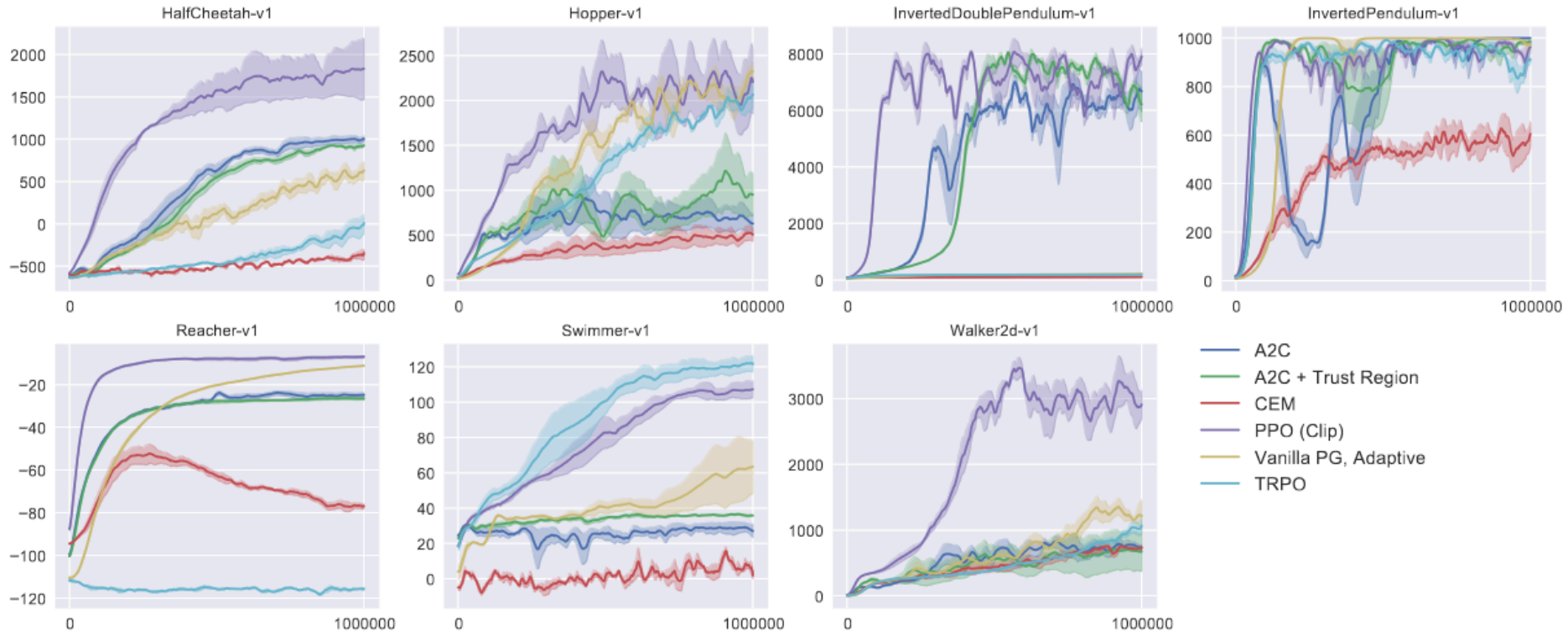



Figure 3: Comparison of several algorithms on several MuJoCo environments, training for one million timesteps.

# Conclusion

- Policy Gradient Theorem: $\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\pi) = \mathbb{E}_{\pi}[\sum_{t=0}^{T} (\gamma^t G_t) \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t)]$
- REINFORCE: PGT + MC for estimate of G
- Actor-Critic: PGT + V,Q for estimate of G
- Deterministic Policy Gradient: $\nabla_\theta J_\theta(\pi \,|\, S_0 = S) \approx \nabla_\theta Q_\pi(\pi(S, \theta), S)$
- PPO is SOTA in many tasks!

# Value-based or policy-based?

- This is an application-dependent choice!
- If policy space is simple to parameterize, policy search/AC work very well
- Eg. powerplant control
- If policy space is complicated, value-based is better
- Using a value function can greatly reduce variance