Advanced Policy Gradients

Deep Reinforcement Learning

Mohammad Hossein Rohban, Ph.D.

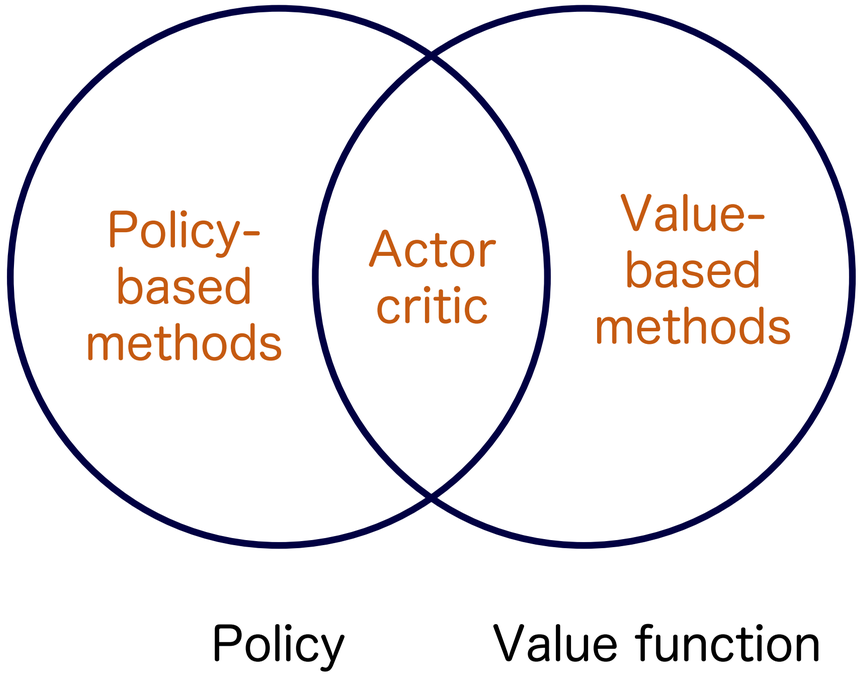
Courtesy: Some slides are adopted from CS 285 Berkeley, and CS 234 Stanford, and Pieter Abbeel's compact series on RL.

RIML Lab, CE Department, Sharif University of Technology

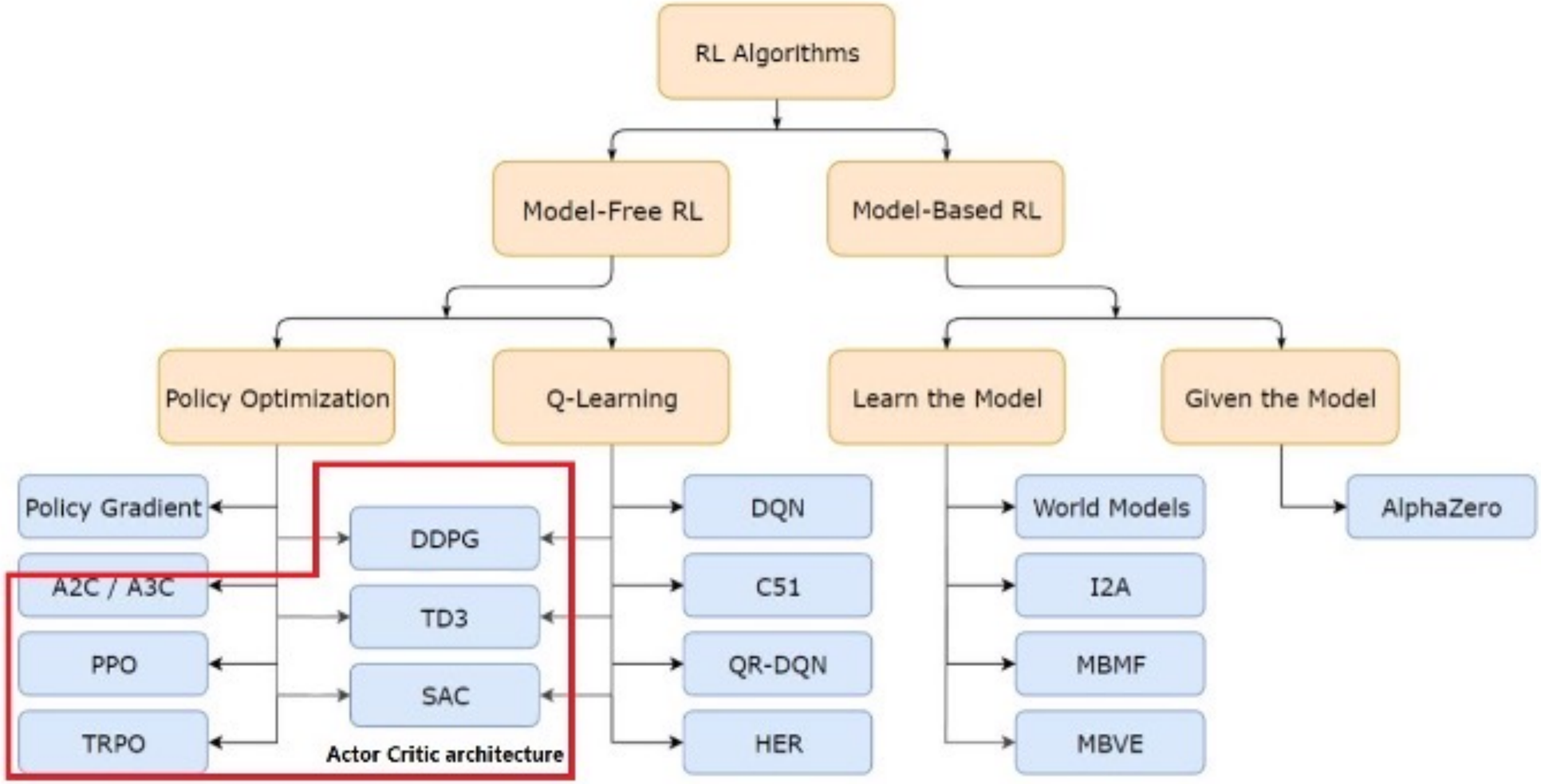
Spring 2026

Model-Free RL

- Value-based methods
  - Learnt value function
  - Implicit policy
- Policy-based methods
  - No value function
  - Learnt policy
- Actor-critic methods
  - Learnt value function
  - Learnt policy



Overview of Modern RL Methods

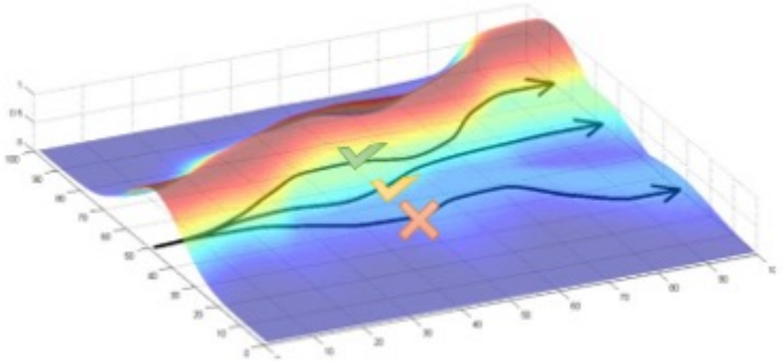


Policy Gradient Intuition

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a_{i,t}}|\mathbf{s_{i,t}})\right)\left(\sum_{t=1}^{T} r(\mathbf{s_{i,t}}, \mathbf{a_{i,t}})\right)$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N} \underbrace{\nabla_\theta \log \pi_\theta(\tau_i)}_{\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a_{i,t}}|\mathbf{s_{i,t}})} r(\tau_i)$$

maximum likelihood: $\nabla_\theta J_{\mathrm{ML}}(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\nabla_\theta \log \pi_\theta(\tau_i)$



- Good stuff is made more likely
- Bad stuff is made less likely
- Simply formalizes the notion of “trial and error”!

Bias and Variance of Policy Gradient

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau_i)\, r(\tau_i)$$

The main source of high variance

- Unbiased estimation:

$$E\left[\frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau_i)\, r(\tau_i)\right] = \nabla_\theta J(\theta)$$

- But suffers from high variance!

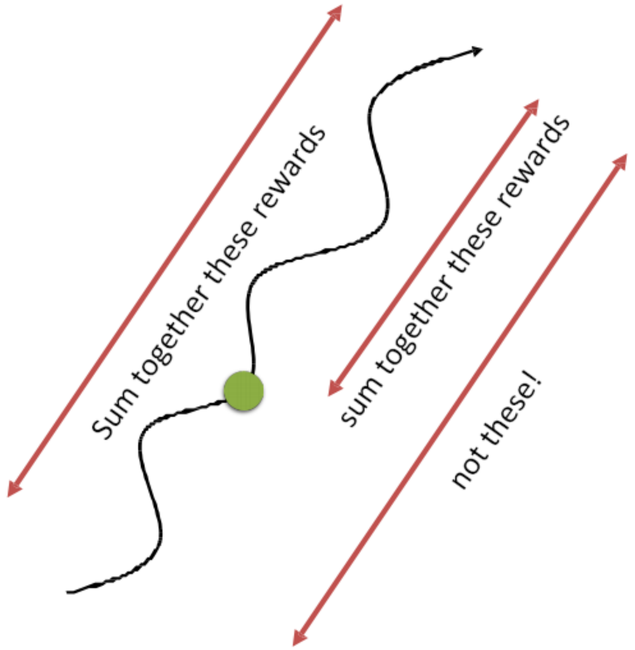
Reducing Variance

- Everything in the gradient whose expected is zero could be removed, without affecting the optimization, but could lead to lower gradient variance!
- Causality trick
- Discount factor
- Baseline
- Actor-critic
- Optimization techniques:
  - ☑ Natural gradient
  - ☑ Trust region

Reducing Variance: Causality

- $\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta\left(\mathbf{a_{i,t}}|\mathbf{s_{i,t}}\right)\right)\left(\sum_{t=1}^{T} r\left(\mathbf{s_{i,t}}, \mathbf{a_{i,t}}\right)\right)$
- Causality: policy at tome t' cannot affect reward at time t when t<t'
- $\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\left(\sum_{t'=1}^{T} r(\mathbf{a}_{i,t'}, \mathbf{s}_{i,t'})\right)$
- $\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\left(\sum_{t'=t}^{T} r(\mathbf{a}_{i,t'}, \mathbf{s}_{i,t'})\right)$

"reward to go" $\widehat{Q_{i,t}}$



Reducing Variance: Discount Factor

$$\text{option 1:} \quad \nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta\left(\mathbf{a_{i,t}}|\mathbf{s_{i,t}}\right)\left(\sum_{t'=t}^{T}\gamma^{t'-t}r\left(\mathbf{s_{i,t'}},\mathbf{a_{i,t'}}\right)\right)$$

$$\text{option 2:} \quad \nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta\left(\mathbf{a_{i,t}}|\mathbf{s_{i,t}}\right)\right)\left(\sum_{t=1}^{T}\gamma^{t-1}r\left(\mathbf{s_{i,t'}},\mathbf{a_{i,t'}}\right)\right)$$

Not the same!

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta\left(\mathbf{a_{i,t}}|\mathbf{s_{i,t}}\right)\left(\sum_{t=1}^{T}\gamma^{t-1}r\left(\mathbf{s_{i,t'}},\mathbf{a_{i,t'}}\right)\right)$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\gamma^{t-1}\nabla_\theta \log \pi_\theta\left(\mathbf{a_{i,t}}|\mathbf{s_{i,t}}\right)\left(\sum_{t'=t}^{T}\gamma^{t'-t}r\left(\mathbf{s_{i,t'}},\mathbf{a_{i,t'}}\right)\right)$$

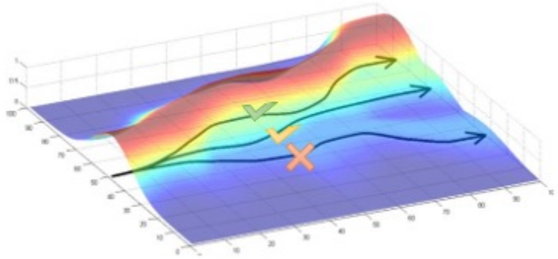
Reducing Variance: Baselines

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log p_\theta(\tau)\,[r(\tau) - b]$$

A convenient identity

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

$$b = \frac{1}{N}\sum_{i=1}^{N} r(\tau)$$



$$E[\nabla_\theta \log p_\theta(\tau)\, b] = \int p_\theta(\tau)\nabla_\theta \log p_\theta(\tau)\, b\, d\tau = \int \nabla_\theta p_\theta(\tau) b\, d\tau = b\nabla_\theta \int p_\theta(\tau) d\tau = b\nabla_\theta 1 = 0$$

Subtracting a baseline is unbiased expectation!

Average reward is not the best baseline, but it's pretty good!

Analyzing Variance

$$\mathrm{Var}[x] = E[x^2] - E[x]^2$$

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

$$\mathrm{Var} = E_{\tau\sim p_\theta(\tau)}\left[\underbrace{\left(\nabla_\theta \log p_\theta(\tau)\,(r(\tau) - b)\right)^2}_{g(\tau)}\right] - E_{\tau\sim p_\theta(\tau)}[\nabla_\theta \log p_\theta(\tau)\,(r(\tau) - b)]^2$$

this bit is just $E_{\tau\sim p_\theta(\tau)}[\nabla_\theta \log p_\theta(\tau)\, r(\tau)]$
baselines are unbiased in expectation

$$\frac{d\mathrm{Var}}{db} = \frac{d}{db} E[g(\tau)^2(r(\tau) - b)^2] = \frac{d}{db}\left(E[g(\tau)^2 r(\tau)^2] - 2E[g(\tau)^2 r(\tau) b] + b^2 E[g(\tau)^2]\right)$$

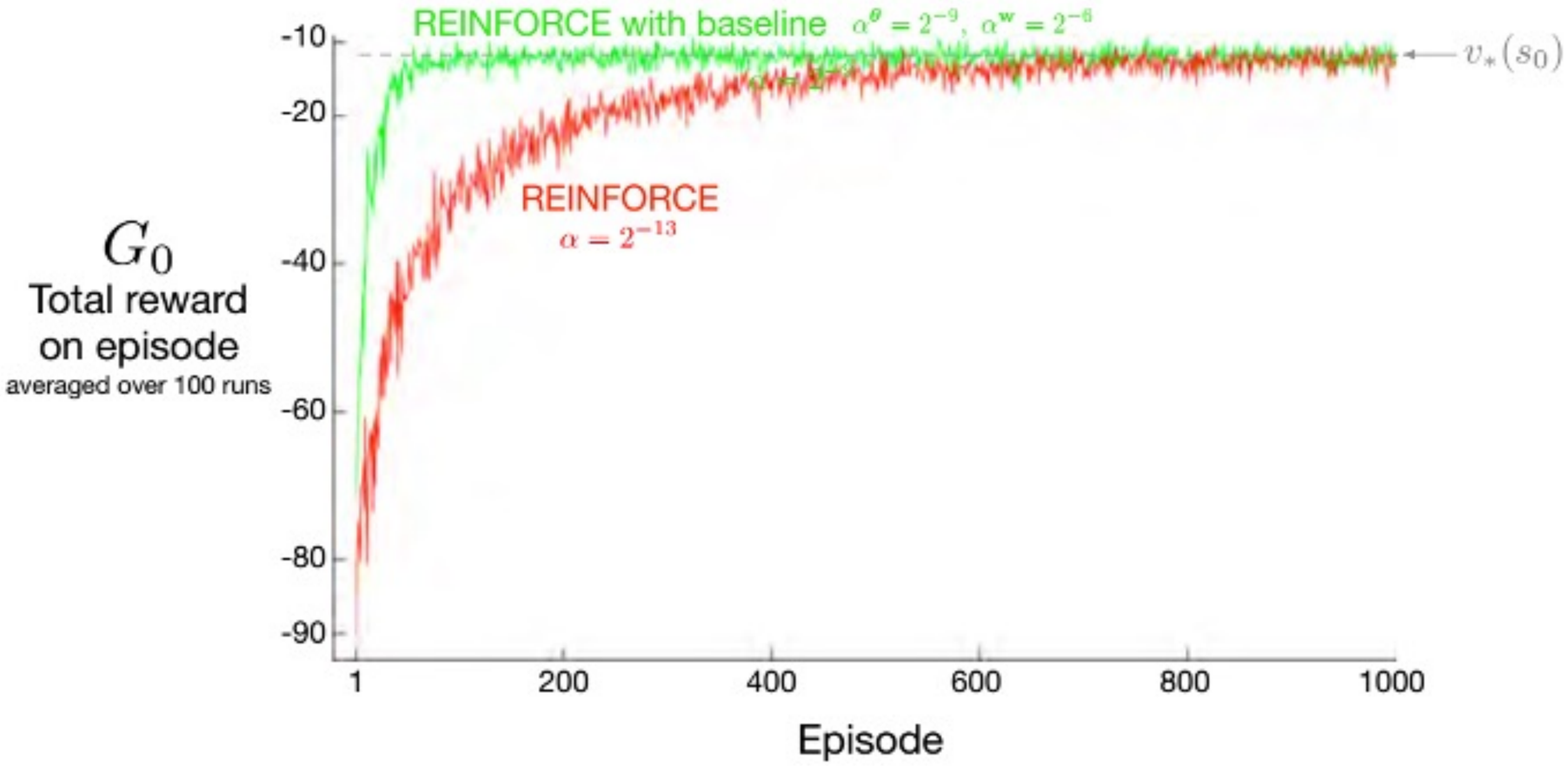
$$= -2E[g(\tau)^2 r(\tau)] + 2bE[g(\tau)^2] = 0$$

$$b = \frac{E[g(\tau)^2 r(\tau)]}{E[g(\tau)^2]}$$

This is just expected reward, but weighted by gradient magnitudes!

Reducing Variance: Baselines

Faster convergence:



Reducing Variance: Review

- Exploiting causality
  - ☑ Future doesn't affect the past
- Discount factor
  - ☑ Two different version
- Baselines
  - ☑ Analyzing variance for deriving optimal baselines
- Now: Introducing actor-critic methods!

Policy Gradients so Far

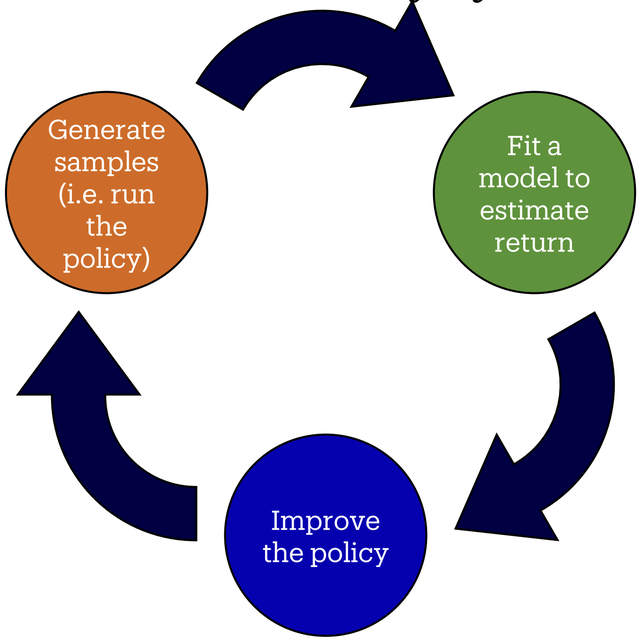
Algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run the policy)

2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t^i|\mathbf{s}_t^i\right)\right)\left(\sum_t r\left(\mathbf{s}_t^i, \mathbf{a}_t^i\right)\right)$

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

$$\hat{Q}^\pi(\mathbf{x_t}, \mathbf{u_t}) = \sum_{t'=t}^{T} r(\mathbf{x_{t'}}, \mathbf{u_{t'}})$$



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

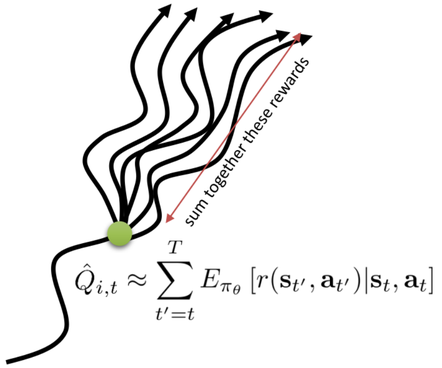
$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a_{i,t}}|\mathbf{s_{i,t}}\right) \underbrace{\hat{Q}_{i,t}^\pi}_{reward\ to\ go}$$

Improving Estimation of Reward to Go

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta\left(\mathbf{a_{i,t}}|\mathbf{s_{i,t}}\right)\underbrace{\left(\sum_{t'=1}^{T} r\left(\mathbf{s_{i,t'}}, \mathbf{a_{i,t'}}\right)\right)}_{reward\ to\ go}$$

$\hat{Q}_{i,t}$: estimate of expected reward if we take action $a_{i,t}$ in state $s_{i,t}$

☑ How to make a better estimate?



$$Q(\mathbf{s_t}, \mathbf{a_t}) = \sum_{t'=t}^{T} E_{\pi_\theta}[r(\mathbf{s_{t'}}, \mathbf{a_{t'}})|\mathbf{s_t}, \mathbf{a_t}]: \text{ true expected reward-to-go}$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta\left(\mathbf{a_{i,t}}|\mathbf{s_{i,t}}\right) Q\left(\mathbf{s_{i,t}}, \mathbf{a_{i,t}}\right)$$

Improving Estimation of Reward to Go (cont.)

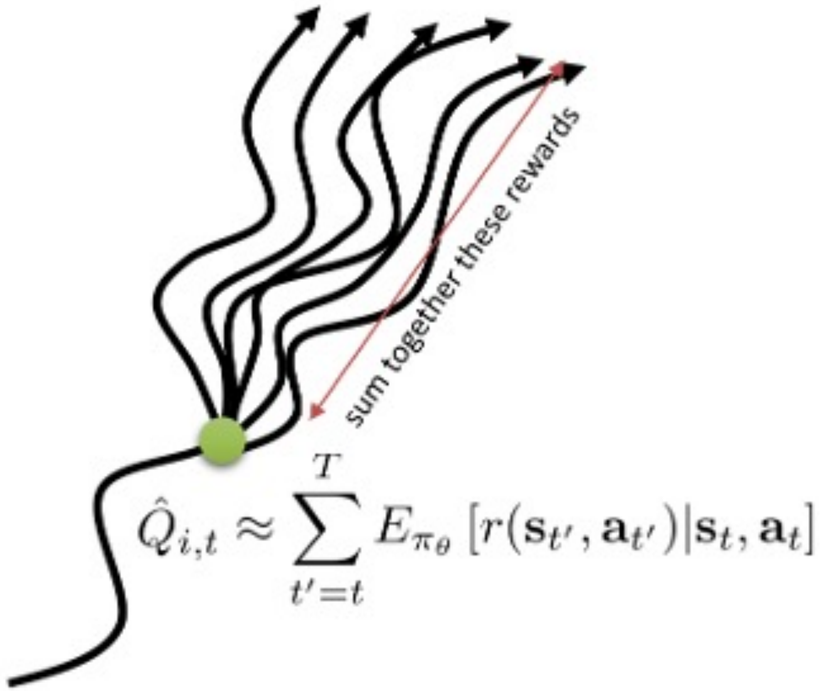
- Further improvement: Adding a baseline!

$$Q(\mathbf{s_t}, \mathbf{a_t}) = \sum_{t'=t}^{T} E_{\pi_\theta}[r(\mathbf{s_{t'}}, \mathbf{a_{t'}})|\mathbf{s_t}, \mathbf{a_t}]: \text{ true expected reward—to—go}$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a_{i,t}}|\mathbf{s_{i,t}})\left(Q(\mathbf{s_{i,t}}, \mathbf{a_{i,t}}) - b_t\right)$$

$$b_t = \frac{1}{N}\sum_{i} Q(\mathbf{s_{i,t}}, \mathbf{a_{i,t}})$$

$$V(\mathbf{s_t}) = E_{\mathbf{a_t}\sim\pi_\theta(\mathbf{a_t}|\mathbf{s_t})}[Q(\mathbf{s_t}, \mathbf{a_t})]$$



Advantage Value

$$Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_t, \mathbf{a}_t]: \textit{total reward from taking } \mathbf{a}_t \textit{ in } \mathbf{s}_t$$

$$V^{\pi}(\mathbf{s}_t) = E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}[Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t)]: \textit{total reward from } \mathbf{s}_t$$

$$A^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) - V^{\pi}(\mathbf{s}_t): \textit{how much better } \mathbf{a}_t \textit{ is}$$

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

The better this estimate, the lower the variance

Advantage Value Approximation

$$Q^{\pi}(\mathbf{s_t}, \mathbf{a_t}) = \sum_{t'=t}^{T} E_{\pi_\theta}[r(\mathbf{s_{t'}}, \mathbf{a_{t'}})|\mathbf{s_t}, \mathbf{a_t}]$$

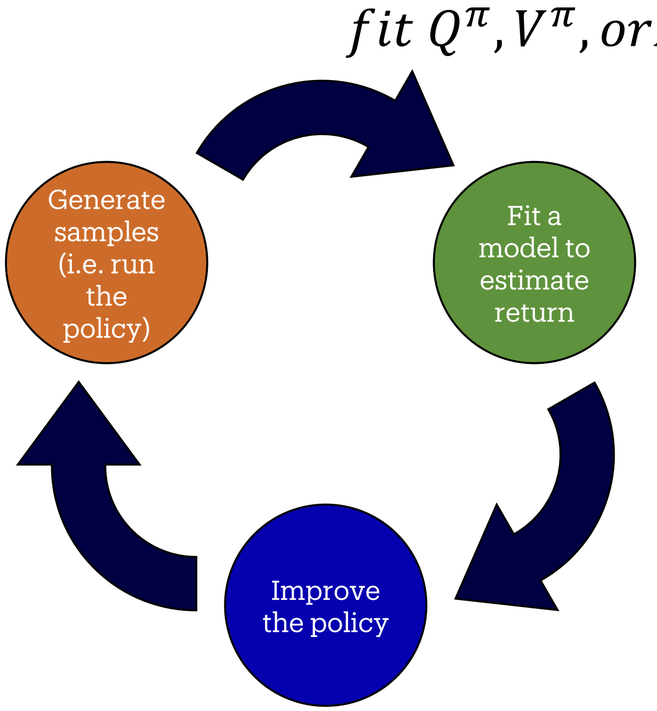
$$V^{\pi}(\mathbf{s_t}) = E_{\mathbf{a_t}\sim\pi_\theta(\mathbf{a_t}|\mathbf{s_t})}[Q^{\pi}(\mathbf{s_t}, \mathbf{a_t})]$$

$$A^{\pi}(\mathbf{s_t}, \mathbf{a_t}) = Q^{\pi}(\mathbf{s_t}, \mathbf{a_t}) - V^{\pi}(\mathbf{s_t})$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a_{i,t}}\middle|\mathbf{s_{i,t}}\right) A^{\pi}\left(\mathbf{s_{i,t}}, \mathbf{a_{i,t}}\right)$$

$$Q^{\pi}(\mathbf{s_t}, \mathbf{a_t}) = \sum_{t'=t}^{T} E_{\pi_\theta}[r(\mathbf{s_{t'}}, \mathbf{a_{t'}})|\mathbf{s_t}, \mathbf{a_t}]$$

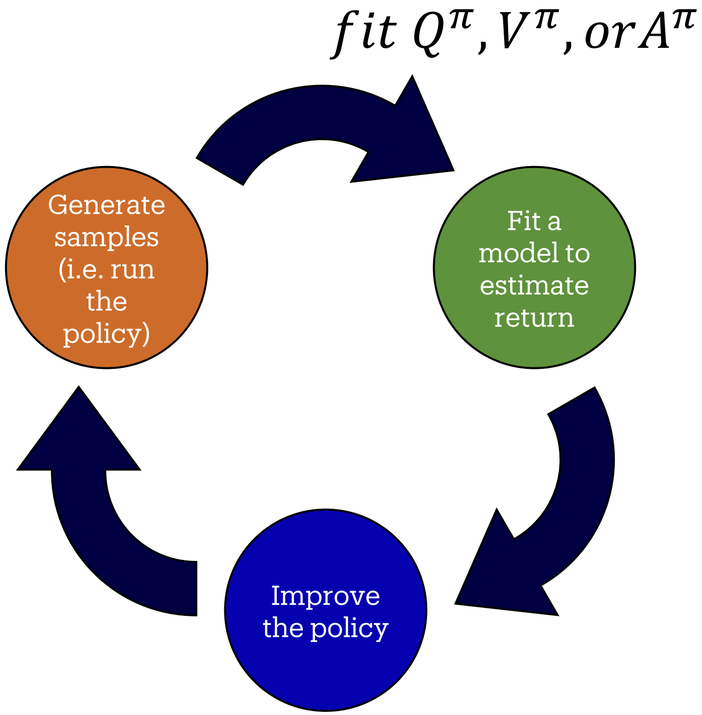
$$= r(\mathbf{s_t}, \mathbf{a_t}) + \underbrace{\sum_{t'=t+1}^{T} E_{\pi_\theta}[r(\mathbf{s_{t'}}, \mathbf{a_{t'}})|\mathbf{s_t}, \mathbf{a_t}]}_{\approx V^{\pi}(\mathbf{s_{t+1}})}$$



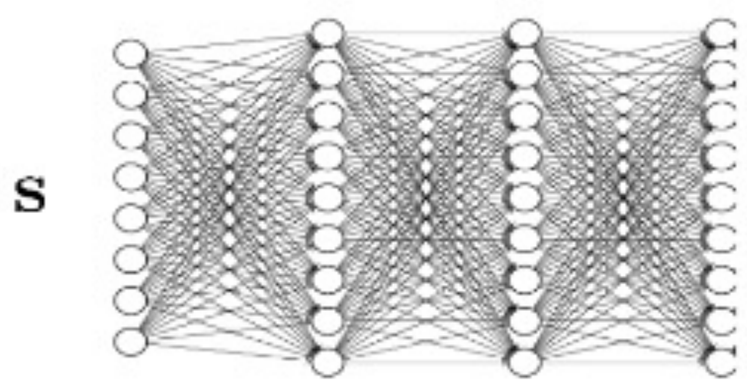
$$\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$$

Advantage Value Approximation (cont.)

- $Q^\pi(\mathbf{s}_t, \mathbf{a}_t) \approx r(\mathbf{s}_t, \mathbf{a}_t) + V^\pi(\mathbf{s}_{t+1})$
- $A^\pi(\mathbf{s}_t, \mathbf{a}_t) \approx r(\mathbf{s}_t, \mathbf{a}_t) + V^\pi(\mathbf{s}_{t+1}) - V^\pi(\mathbf{s}_t)$



- Let's just fit $V^\pi(s)$!



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

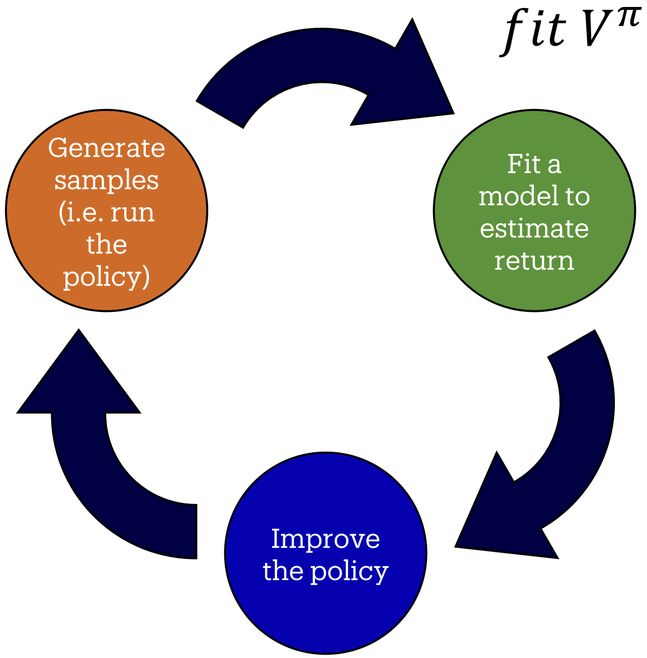
Policy Evaluation

$$V^{\pi}(\mathbf{s_t}) = \sum_{t'=t}^{T} E_{\pi_\theta}[r(\mathbf{s_{t'}}, \mathbf{a_{t'}})|\mathbf{s_t}]$$

$$J(\theta) = E_{\mathbf{s_1} \sim p(\mathbf{s_1})}[V^{\pi}(\mathbf{s_1})]$$

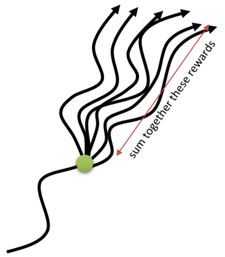
How can perform policy evaluation?

Monte Carlo policy evaluation (this is what policy gradient does)



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

$$V^{\pi}(\mathbf{s_t}) \approx \sum_{t'=t}^{T} r(\mathbf{s_{t'}}, \mathbf{a_{t'}})$$



$$V^{\pi}(\mathbf{s_t}) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t'=t}^{T} r(\mathbf{s_{t'}}, \mathbf{a_{t'}})$$

(requires us to reset the simulator)

Policy Evaluation (cont.)

✦ Monte Carlo estimation with function approximator:

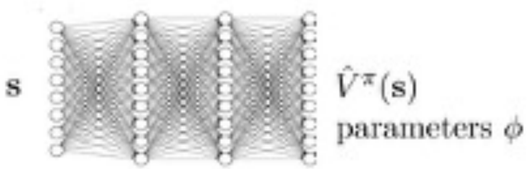
$$V^{\pi}(\mathbf{s_t}) \approx \sum_{t'=t}^{T} r(\mathbf{s_{t'}}, \mathbf{a_{t'}})$$

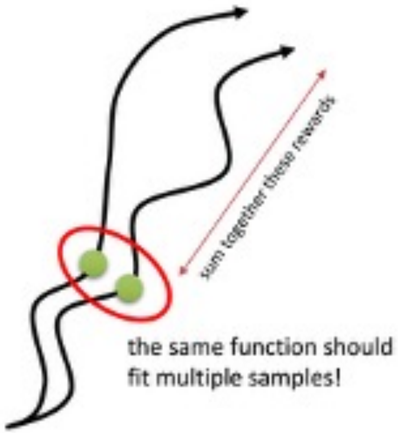
not as good as this: $V^{\pi}(\mathbf{s_t}) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t'=t}^{T} r(\mathbf{s_{t'}}, \mathbf{a_{t'}})$

but still pretty good!

training data: $\left\{\left(\mathbf{s_{i,t}}, \underbrace{\sum_{t'=t}^{T} r(\mathbf{s_{i,t'}}, \mathbf{a_{i,t'}})}_{y_{i,t}}\right)\right\}$

supervised regression $\mathcal{L}(\phi) = \frac{1}{2}\sum_i |\hat{V}_{\phi}^{\pi}(\mathbf{s_i}) - y_i|^2$





Policy Evaluation (cont.)

✦ How to make a better estimate?

ideal target:

$$y_{i,t} = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s_{t'}}, \mathbf{a_{t'}})\middle|\mathbf{s_{i,t}}\right] \approx r\left(\mathbf{s_{i,t}}, \mathbf{a_{i,t}}\right) + V^{\pi}\left(\mathbf{s_{i,t+1}}\right) \approx r\left(\mathbf{s_{i,t}}, \mathbf{a_{i,t}}\right) + \underbrace{\hat{V}_{\phi}^{\pi}\left(\mathbf{s_{i,t+1}}\right)}_{\text{directly use previous fitted value function!}}$$

Monte Carlo target:

$$y_{i,t} = \sum_{t'=t}^{T} r\left(\mathbf{s_{i,t'}}, \mathbf{a_{i,t'}}\right)$$

Policy Evaluation (cont.)

✦ Bootstrap Estimation with Function Approximator

ideal target:

$$y_{i,t} = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s_{t'}}, \mathbf{a_{t'}})\middle|\mathbf{s_{i,t}}\right] \approx r\left(\mathbf{s_{i,t}}, \mathbf{a_{i,t}}\right) + \hat{V}_\phi^\pi\left(\mathbf{s_{i,t+1}}\right)$$

training data: $\left\{\left(\mathbf{s_{i,t}}, \underbrace{r\left(\mathbf{s_{i,t}}, \mathbf{a_{i,t}}\right) + \hat{V}_\phi^\pi\left(\mathbf{s_{i,t+1}}\right)}_{y_{i,t}}\right)\right\}$

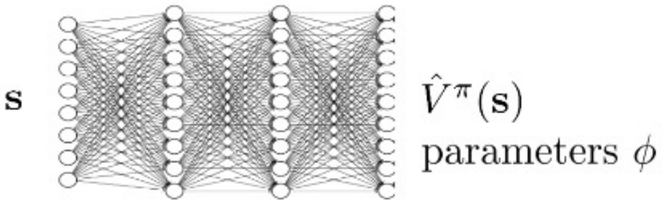
supervised regression:

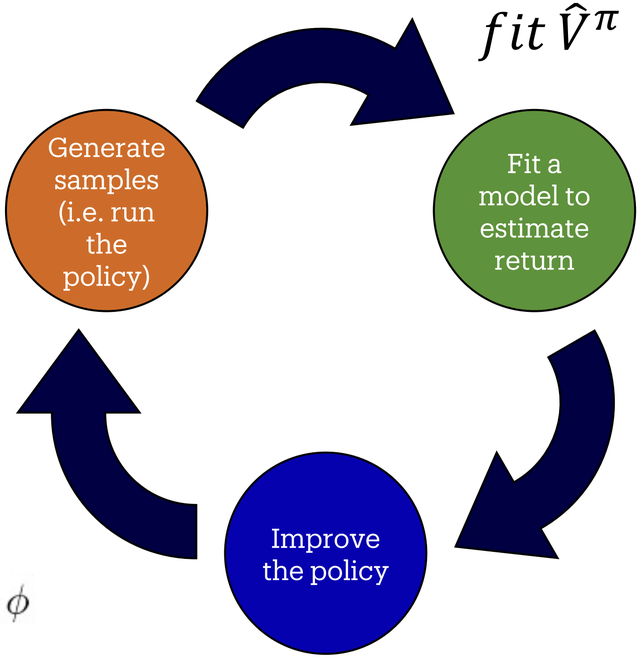
$$\mathcal{L}(\phi) = \frac{1}{2}\sum_i |\hat{V}_\phi^\pi(\mathbf{s_i}) - y_i|^2$$

Batch Actor-Critic Algorithm

batch actor-critic algorithm:

1. sample $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a}|\mathbf{s})$ (run it on the robot)
2. fit $\hat{V}^\pi_\phi(\mathbf{s})$ to sampled reward sums
3. Evaluate
4. > $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(a_i|s_i)\, \hat{A}^\pi(s_i, a_i)$
5. > $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$





$$V^\pi(\mathbf{s_t}) = \sum_{t'=t}^{T} E_{\pi_\theta}[r(\mathbf{s_{t'}}, \mathbf{a_{t'}})|\mathbf{s_t}]$$

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

Actor-Critic Algorithm: Architecture Design

batch actor-critic algorithm:

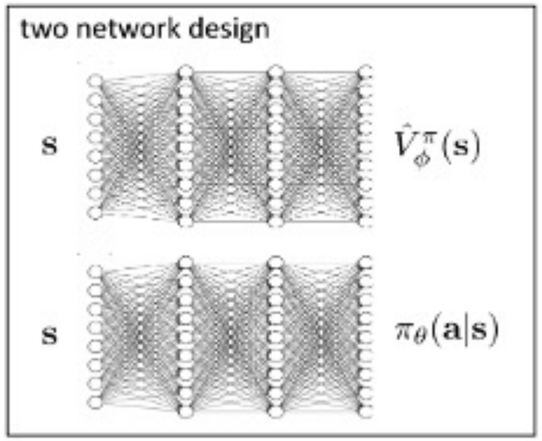
1. sample $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a}|\mathbf{s})$ (run it on the robot)

2. fit $\hat{V}^\pi_\phi(\mathbf{s})$ to sampled reward sums

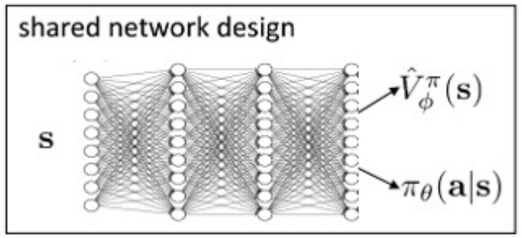
3. Evaluate

4. > $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(a_i|s_i)\, \hat{A}^\pi(s_i, a_i)$

5. > $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$



+simple & stable
- no shared features between actor and critic



Actor-Critic Algorithm: Architecture Design

batch actor-critic algorithm:

1. sample $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a}|\mathbf{s})$ (run it on the robot)
2. fit $\hat{V}^\pi_\phi(\mathbf{s})$ to sampled reward sums
3. Evaluate
4. > $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(a_i|s_i)\, \hat{A}^\pi(s_i, a_i)$
5. > $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

online actor-critic algorithm:

1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$
2. update $\hat{V}^\pi_\phi$ using target r + $\gamma \hat{V}^\pi_\phi(\mathbf{s}')$
3. Evaluate
4. > $\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s})\ \hat{A}^\pi(\mathbf{s}, \mathbf{a})$
5. > $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Proximal Policy Optimization

- We don't want the new policy to change a lot in an iteration. Why?

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

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

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

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

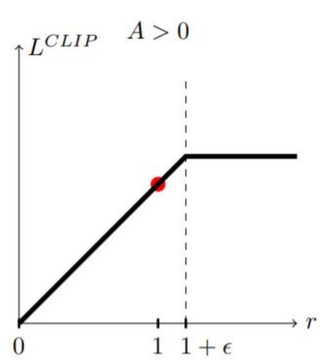
Proximal Policy Optimization

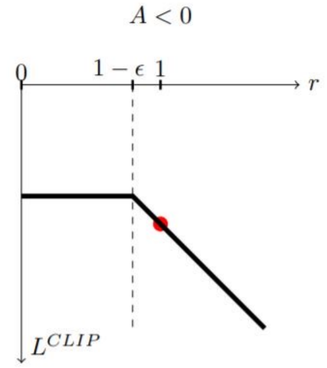
Let's design a simpler objective that directly constrains

$$\text{✦ argmax}_{\theta'} E_{\{s\sim\mu_\theta, a\sim\pi_\theta\}} \; min \begin{cases} \frac{\pi_{\theta'}(a|s)}{\pi_\theta(a|s)} A^{\pi_\theta}(s,a), \\ clip(\frac{\pi_{\theta'}(a|s)}{\pi_\theta(a|s)}, 1-\epsilon, 1+\epsilon) A^{\pi_\theta}(s,a) \end{cases}$$

could be easily implemented with auto-diff packages

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





Proximal Policy Optimization (cont.)

PPO-Clip Algorithm:

1. **Input: initial policy parameters** θ_0**, initial value function parameters** ϕ_0
2. **for** $k = 0,1,2,\ldots$ **do**
3. **Collect set of trajectories** $\mathcal{D}_k = \{\tau_i\}$ **by running policy** $\pi_k = \pi(\theta_k)$ **in the environment.**
4. **Compute rewards-to-go** $\widehat{R_t}$**.**
5. **Compute advantage estimates,** $\hat{A}_t$ **(using any method of advantage estimation) based on the current value function** V_{ϕ_k}**.**
6. **Update the policy by maximizing the PPO-Clip objective:**

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t),\ g\left(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)\right) \right),$$

typically via stochastic gradient ascent with Adam.

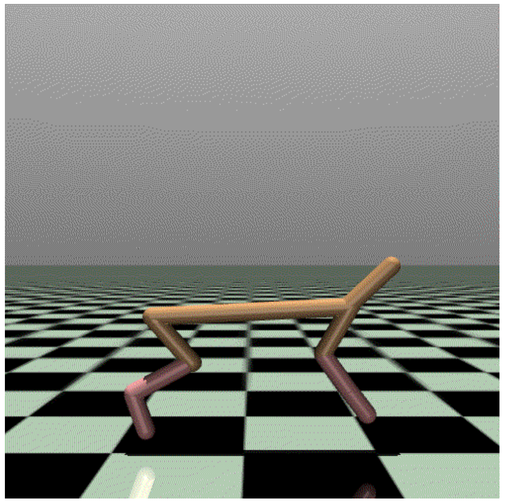
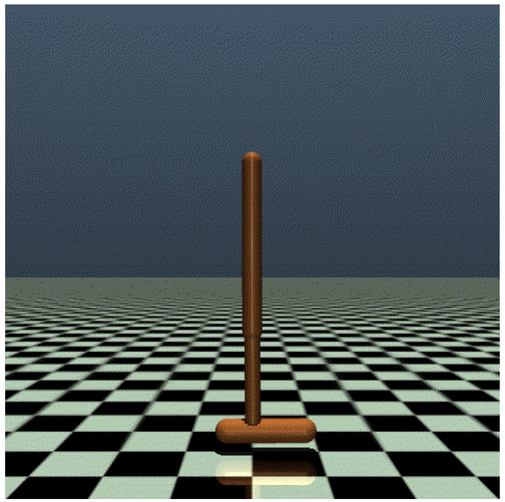
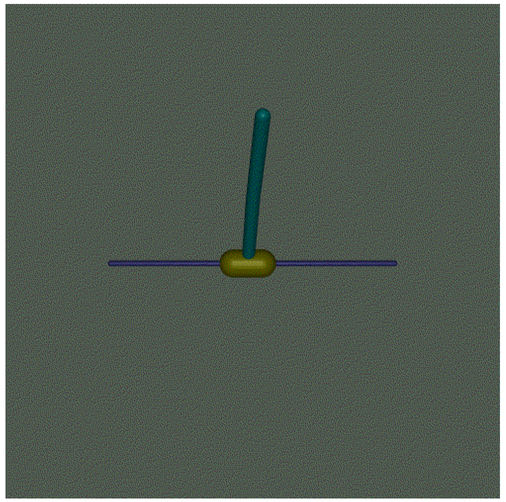
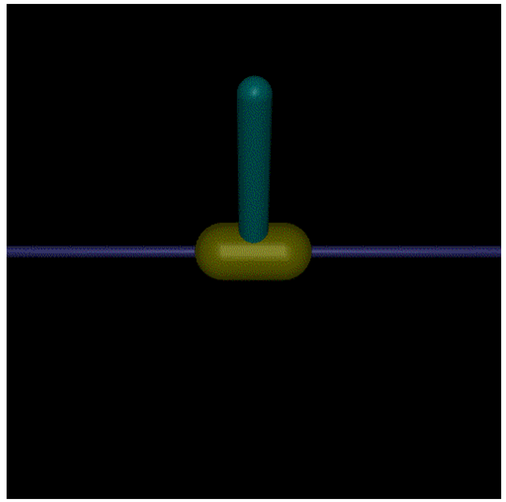
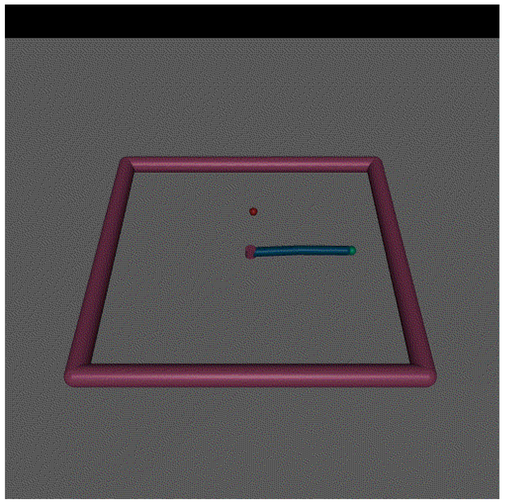
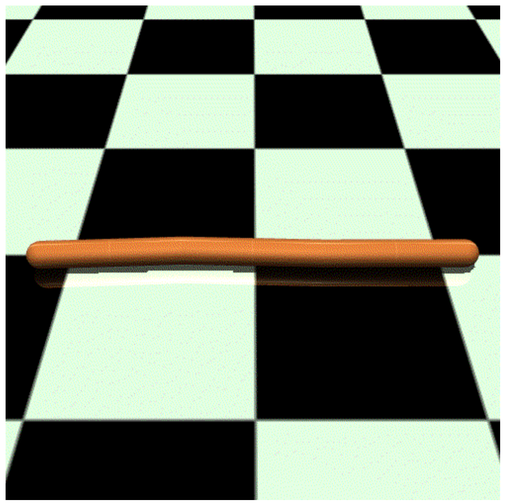
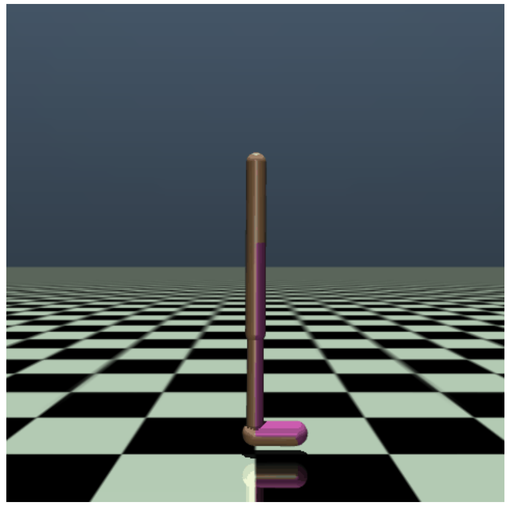
7. **Fit value function by regression on mean-squared error:**

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t\right)^2,$$

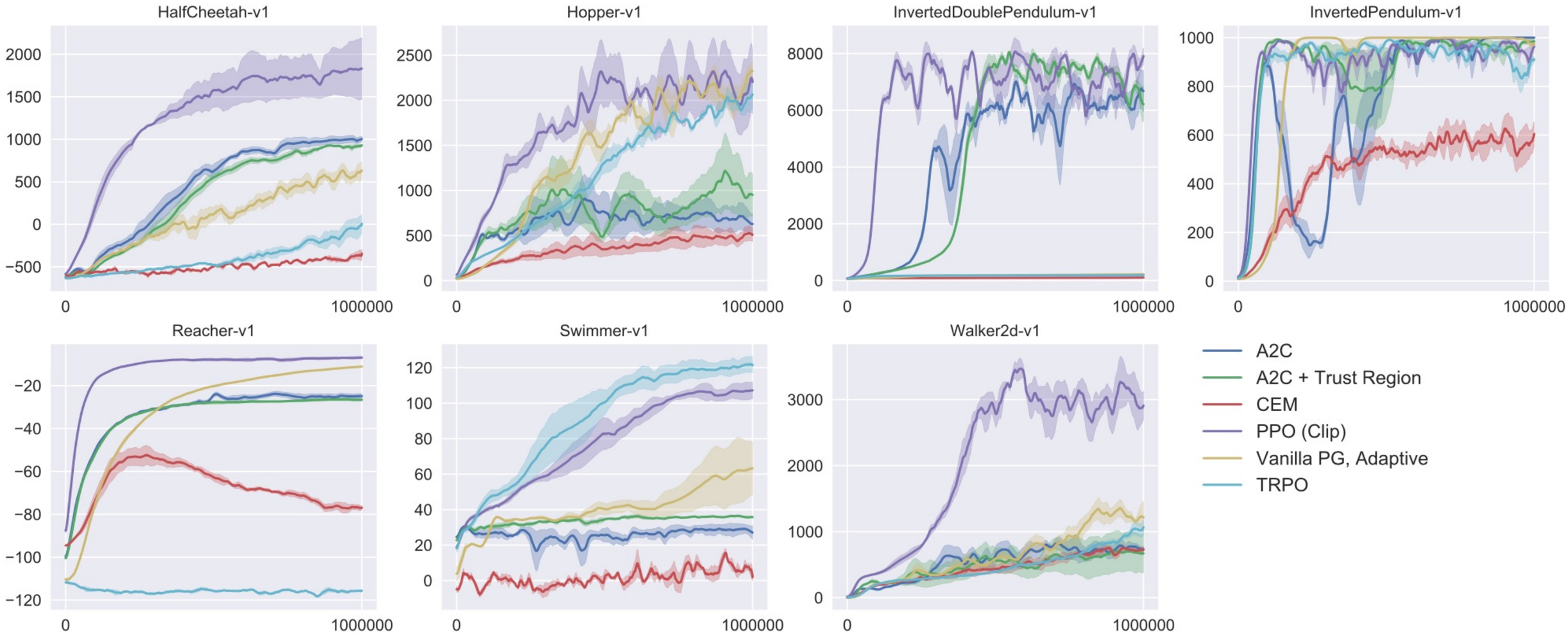
typically via some gradient descent algorithm.

8. **end for**

Environments

PPO Results


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

Soft Actor Critic

One cause for the poor sample efficiency of deep RL methods is on-policy learning: some of the most commonly used deep RL algorithms, such as TRPO (Schulman et al., 2015), PPO (Schulman et al., 2017b) or A3C (Mnih et al., 2016), require new samples to be collected for each gradient step.

ture parameter (Haarnoja et al., 2017). More importantly, the maximum entropy formulation provides a substantial improvement in exploration and robustness: as discussed by Ziebart (2010), maximum entropy policies are robust in the face of model and estimation errors, and as demonstrated by (Haarnoja et al., 2017), they improve exploration by acquiring diverse behaviors. Prior work has proposed

Soft Policies

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(\mathbf{s_t}, \mathbf{a_t}) \sim \rho_\pi} \left[r(\mathbf{s_t}, \mathbf{a_t}) + \alpha \mathcal{H}\left(\pi(\cdot\,|\mathbf{s_t})\right) \right].$$

✦ How does the Bellman equation change?

$$\mathcal{T}^\pi Q(\mathbf{s_t}, \mathbf{a_t}) \triangleq r(\mathbf{s_t}, \mathbf{a_t}) + \gamma \mathbb{E}_{\mathbf{s_{t+1}} \sim \mathbb{p}}[V(\mathbf{s_{t+1}})]$$

Where

$$V(\mathbf{s_t}) = \mathbb{E}_{\mathbf{a_t} \sim \pi}[Q(\mathbf{s_t}, \mathbf{a_t}) - \log \pi(\mathbf{a_t}|\mathbf{s_t})]$$

Soft Policies (cont.)

Lemma 1

(Soft Policy Evaluation). Consider the soft Bellman backup operator $\mathcal{T}^{\pi}$ in Equation 2 and a mapping $Q^0: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ with $|\mathcal{A}| < \infty$, and define $Q^{k+1} = \mathcal{T}^{\pi} Q^k$. Then the sequence Q^k will converge to the soft Q-value of π as $k \rightarrow \infty$.}

Optimal Soft Policy

- The optimal soft policy (optimizing entropy augment objective) is:

$$\pi^*(a|s) = \frac{\exp Q(s,\ a)}{\sum_{a'} \exp Q(s,\ a')}$$

Soft actor-critic

1. Q-Function update

update Q-Function to evaluate policy

$$Q(\mathbf{s},\mathbf{a}) \leftarrow r(\mathbf{s},\mathbf{a}) + \mathbb{E}_{\mathbf{s}'\sim\mathbb{p}_{\mathbb{s}},\mathbf{a}'\sim\pi}[Q(\mathbf{s}',\mathbf{a}') - \log\pi(\mathbf{a}'|\mathbf{s}')]$$

This converges to Q^{π}

2. Update policy

Update the policy with gradient of information projection

$$\pi_{\text{new}} = \arg\min_{\pi'} \mathrm{D}_{\mathrm{KL}}\left(\pi'(\cdot\,|\mathbf{s}) \,\|\, \frac{1}{Z}\exp Q^{\pi}\text{old}\,(\mathbf{s},\cdot)\right)$$

In practice only take one gradient step on this objective

3. Interact with the world, collect more data

Haarnoja, et al. **Soft Actor-Critic Algorithms and Applications**. ‘18

Soft actor-critic (cont.)

Soft Actor-Critic algorithm:

Inputs: The learning rates, λ_π, λ_Q, and λ_V for functions π_θ, Q_w, and V_ψ respectively; the weighting factor $\tau$for exponential moving average.

1. Initialize parameters $\theta, w, \psi, and \overline{\psi}$.
2. **for** each iteration **do**
3. (In practice, a combination of a single environment step and multiple gradient steps is found to work best).
4. **for** each environment setup **do**
5. $a_t \sim \pi_\theta(a_t|s_t)$
6. $s_{t+1} \sim \rho_\pi(s_{t+1}|s_t, a_t)$
7. $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\}$
8. **for** each gradient update step **do**
9. $\psi \leftarrow \psi - \lambda_V \nabla_\psi J_V(\psi)$
10. $w \leftarrow w - \lambda_Q \nabla_w J_Q(w)$
11. $\theta \leftarrow \theta - \lambda_\pi \nabla_\theta J_\pi(\theta)$
12. $\overline{\psi} \leftarrow \tau\psi + (1-\tau)\overline{\psi}$

Loss functions

$$J_V(\psi) = \mathbb{E}_{\mathbf{s_t}\sim\mathbb{D}}\left[\frac{1}{2}\left(V_\psi(\mathbf{s_t}) - \mathbb{E}_{\mathbf{a_t}\sim\pi_\phi}\left[Q_\theta(\mathbf{s_t},\mathbf{a_t}) - \log\pi_\phi(\mathbf{a_t}|\mathbf{s_t})\right]\right)^2\right]$$

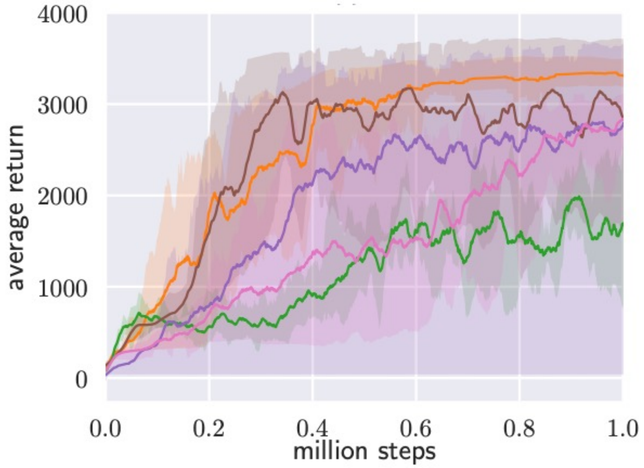
$$J_Q(\theta) = \mathbb{E}_{(\mathbf{s_t},\mathbf{a_t})\sim\mathbb{D}}\left[\frac{1}{2}\left(Q_\theta(\mathbf{s_t},\mathbf{a_t}) - \hat{Q}(\mathbf{s_t},\mathbf{a_t})\right)^2\right]$$

With

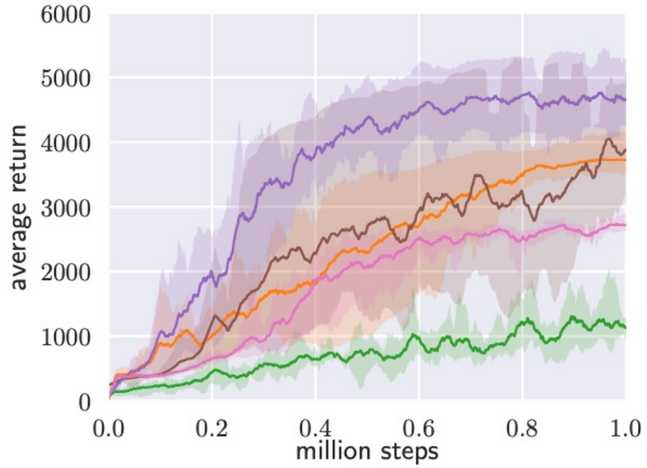
$$\hat{Q}(\mathbf{s_t},\mathbf{a_t}) = r(\mathbf{s_t},\mathbf{a_t}) + \gamma\mathbb{E}_{\mathbf{s_{t+1}}\sim\mathbb{p}}\left[V_{\bar{\psi}}(\mathbf{s_{t+1}})\right]$$

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s_t}\sim\mathbb{D}}\left[\mathrm{D_{KL}}\left(\pi_\phi(\cdot\,|\mathbf{s_t})\middle|\frac{\exp\left(Q_\theta(\mathbf{s_t},\cdot)\right)}{Z_\theta(\mathbf{s_t})}\right)\right]$$

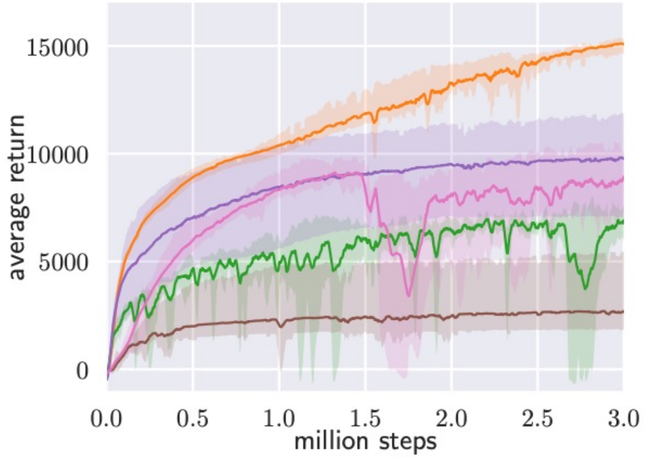
SAC Results



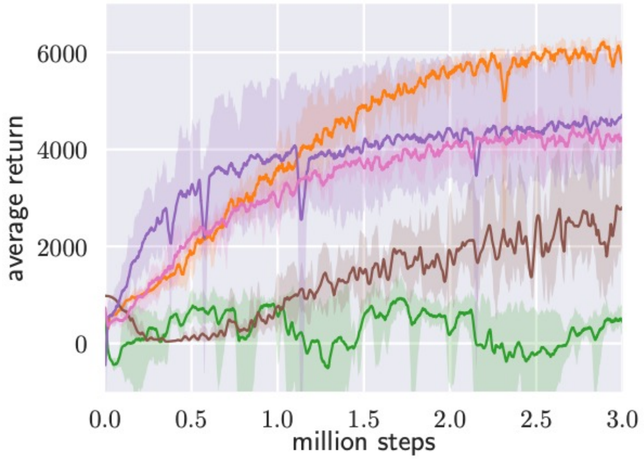
(a) Hopper-v1



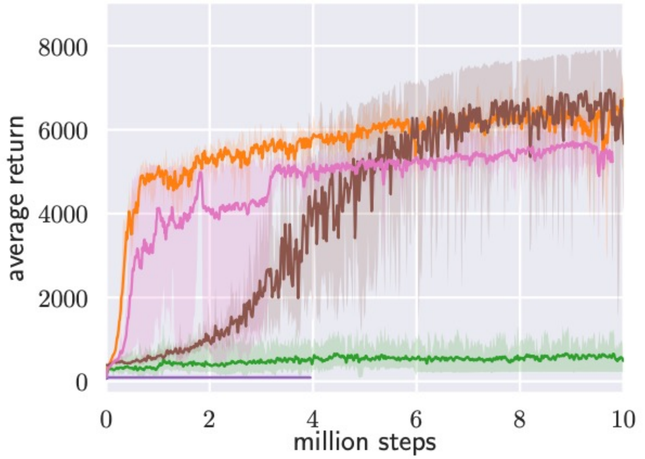
(b) Walker2d-v1



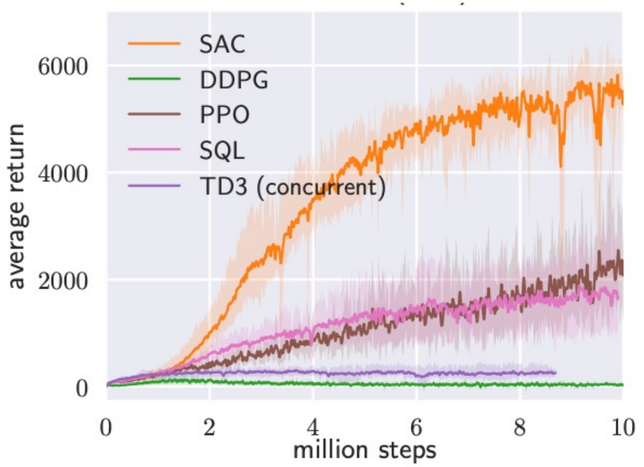
(c) HalfCheetah-v1



(d) Ant-v1

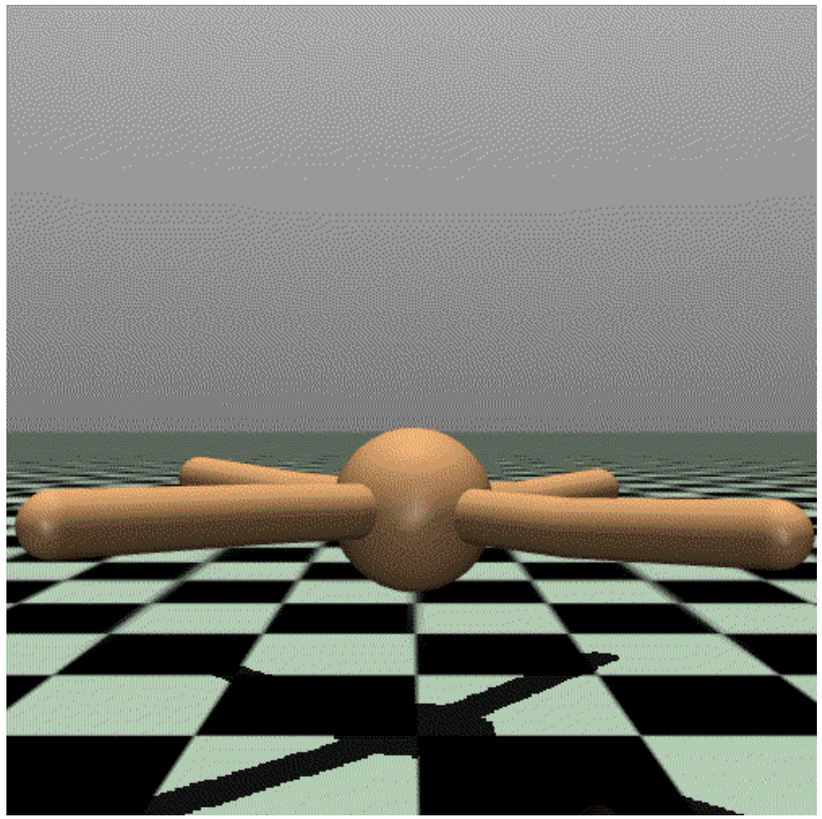


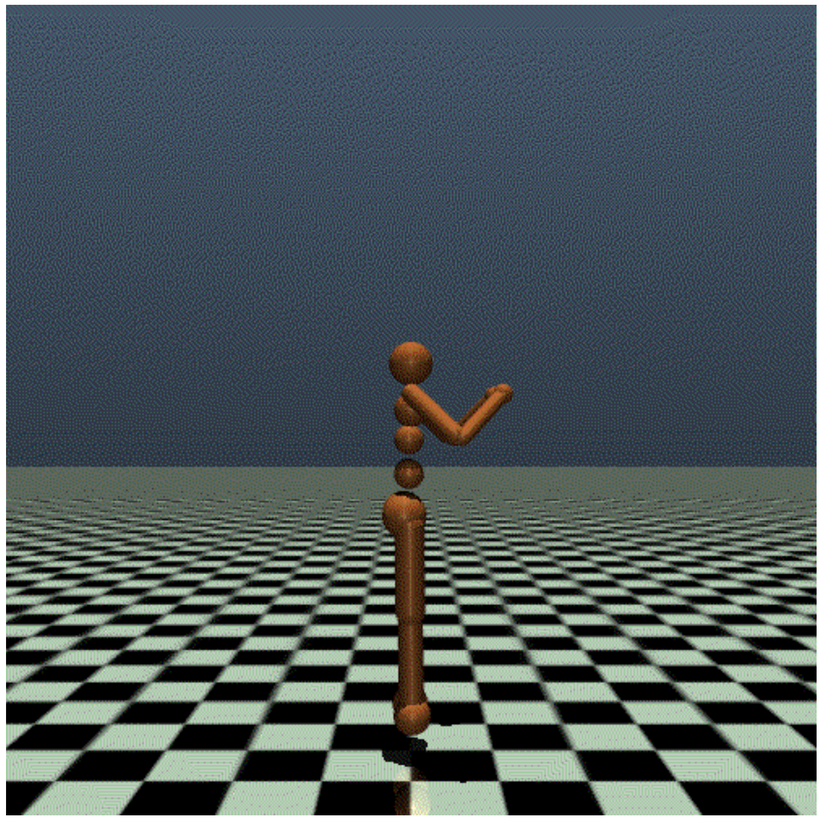
(e) Humanoid-v1



(f) Humanoid (rllab)

Environments





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
- 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

Related Papers (cont.)

- 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