Carnegie Mellon
School of Computer Science

Deep Reinforcement Learning and Control

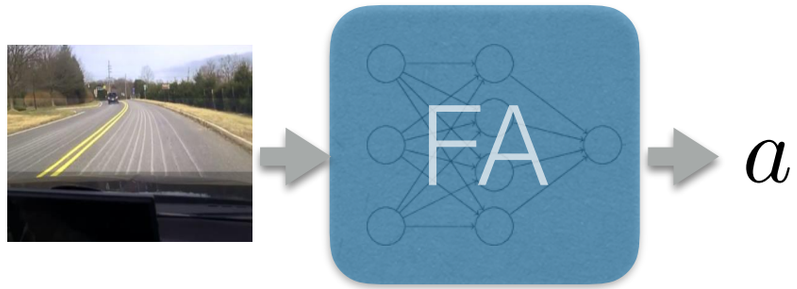
Policy gradients (cont.)

CMU 10-703

Katerina Fragkiadaki

Policy function approximators - this lecture

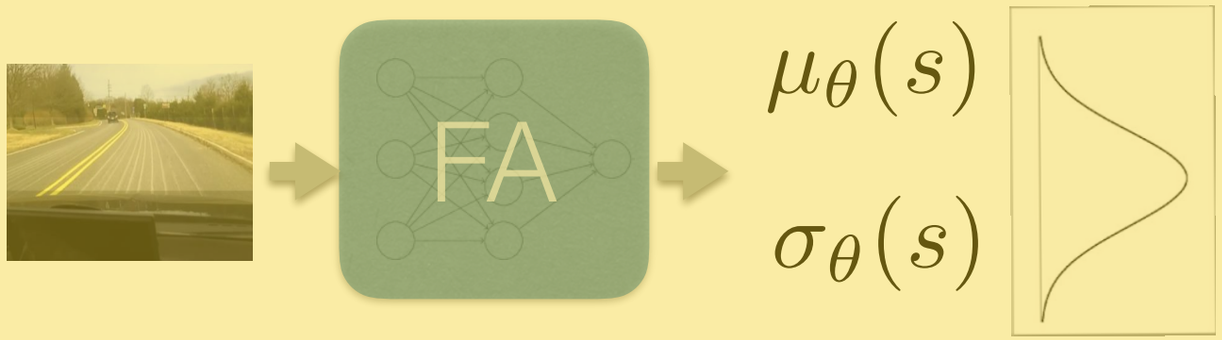
deterministic continuous policy



$$a = \pi_\theta(s)$$

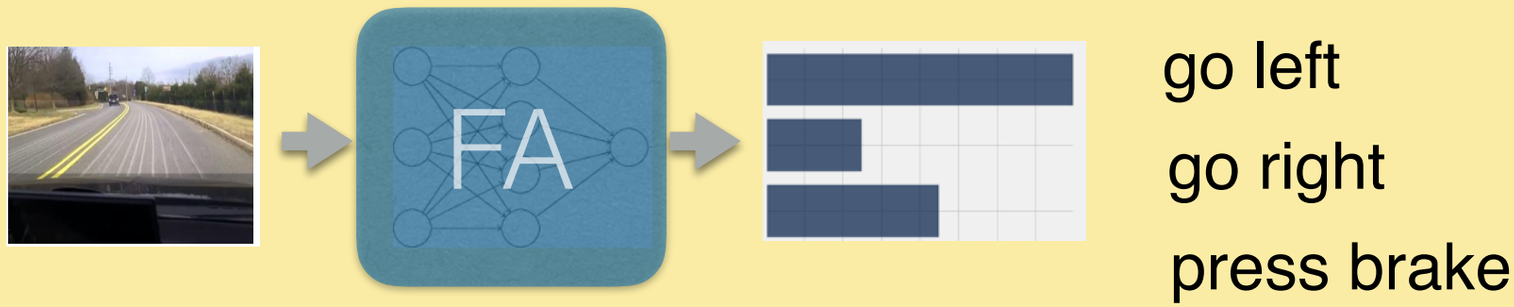
e.g. outputs a steering angle directly

stochastic continuous policy



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

(stochastic) policy over discrete actions



Outputs a distribution over a discrete set of actions

Derivatives of expectations

$$\nabla_\theta \mathbb{E}_x f(x) = \nabla_\theta \mathbb{E}_{x\sim P_\theta(x)}\left[f(x)\right]$$
$$= \nabla_\theta \sum_x P_\theta(x) f(x)$$
$$= \sum_x \nabla_\theta P_\theta(x) f(x)$$
$$= \sum_x P_\theta(x) \frac{\nabla_\theta P_\theta(x)}{P_\theta(x)} f(x)$$
$$= \sum_x P_\theta(x) \nabla_\theta \log P_\theta(x) f(x)$$
$$= \mathbb{E}_{x\sim P_\theta(x)}\left[\nabla_\theta \log P_\theta(x) f(x)\right]$$
$$\approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log P_\theta(x^{(i)}) f(x^{(i)})$$

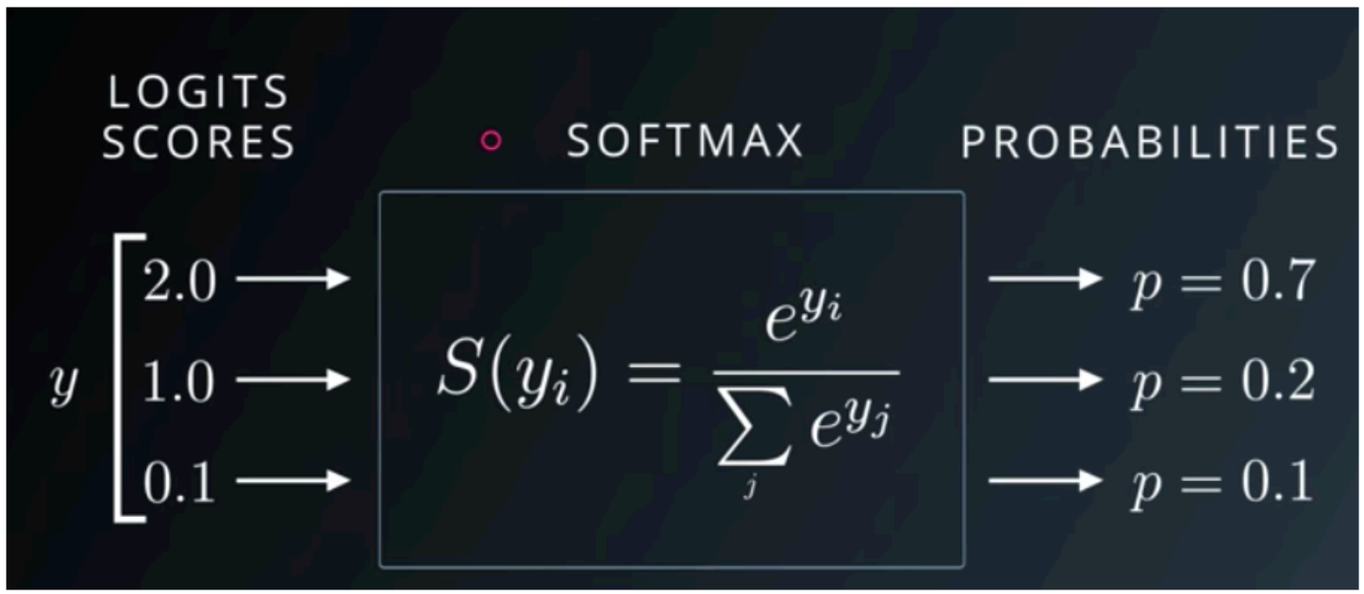
$$\hat{g} = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t^{(i)} \mid s_t^{(i)}) \left(\sum_{k=t}^{H} R(s_k^{(i)}, a_k^{(i)})\right)$$

I can obtain an unbiased estimator for the gradient $\nabla_\theta \mathbb{E}_x f(x)$ by sampling!

$\nabla_\theta \log \pi_\theta(a)$ for Gaussian policy

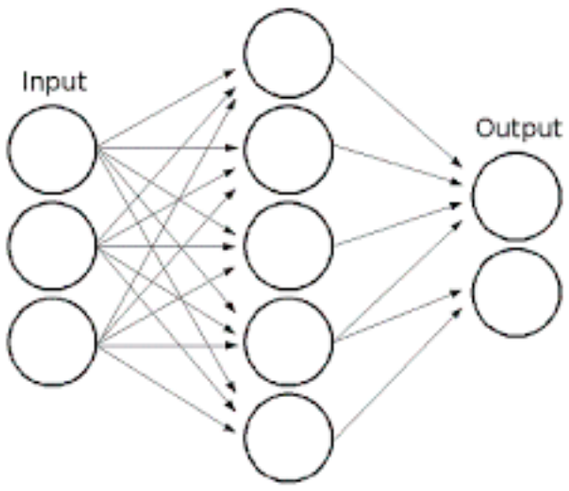
- Policy is Gaussian $a \sim \mathcal{N}(\mu(s,\theta), \sigma^2 I)$
- Variance may be fixed σ^2, or can also parameterized
- Remember: univariete Gaussian $\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(a-\mu)^2}{\sigma^2}}$
- $\nabla_\theta \log \pi_\theta(a \mid s) = \text{const.} \cdot \frac{(\mu_\theta(s) - a)}{\sigma^2} \nabla_\theta \mu_\theta(s)$

Softmax policy



$$\pi(a|s,\boldsymbol{\theta}) \doteq \frac{e^{h(s,a,\boldsymbol{\theta})}}{\sum_b e^{h(s,b,\boldsymbol{\theta})}}$$

discrete actions



s

go left

go right

Output is a distribution over a discrete set of actions

$\nabla_\theta \log \pi_\theta(a)$ for softmax policy

- Policy is the output of a softmax

$$\pi_\theta(a \mid s) = \frac{e^{h_\theta(s,a)}}{\sum_b e^{h_\theta(s,b)}}$$

$$\nabla_\theta \log \pi_\theta(a) =$$

$\nabla_\theta \log \pi_\theta(a)$ for softmax policy

- Policy is the output of a softmax

$$\pi_\theta(a \mid s) = \frac{e^{h_\theta(s,a)}}{\sum_b e^{h_\theta(s,b)}}$$

$$\nabla_\theta \log \pi_\theta(a) = \nabla_\theta h_\theta(s,a) - \nabla_\theta \log \sum_b e^{h_\theta(s,b)}$$

$\nabla_\theta \log \pi_\theta(a)$ for softmax policy

- Policy is the output of a softmax

$$\pi_\theta(a \mid s) = \frac{e^{h_\theta(s,a)}}{\sum_b e^{h_\theta(s,b)}}$$

$$\nabla_\theta \log \pi_\theta(a) = \nabla_\theta h_\theta(s,a) - \nabla_\theta \log \sum_b e^{h_\theta(s,b)}$$

$$\nabla_\theta h_\theta(s,a) - \frac{1}{\sum_b e^{h_\theta(s,b)}} \nabla_\theta \sum_b e^{h_\theta(s,b)}$$

$\nabla_\theta \log \pi_\theta(a)$ for softmax policy

- Policy is the output of a softmax

$$\pi_\theta(a \mid s) = \frac{e^{h_\theta(s,a)}}{\sum_b e^{h_\theta(s,b)}}$$

$$\nabla_\theta \log \pi_\theta(a) = \nabla_\theta h_\theta(s,a) - \nabla_\theta \log \sum_b e^{h_\theta(s,b)}$$

$$\nabla_\theta h_\theta(s,a) - \frac{1}{\sum_b e^{h_\theta(s,b)}} \nabla_\theta \sum_b e^{h_\theta(s,b)}$$

$$\nabla_\theta h_\theta(s,a) - \frac{1}{\sum_b e^{h_\theta(s,b)}} \sum_b \nabla_\theta e^{h_\theta(s,b)}$$

$\nabla_\theta \log \pi_\theta(a)$ for softmax policy

- Policy is the output of a softmax

$$\pi_\theta(a \mid s) = \frac{e^{h_\theta(s,a)}}{\sum_b e^{h_\theta(s,b)}}$$

$$\begin{aligned}
\nabla_\theta \log \pi_\theta(a) = \; & \nabla_\theta h_\theta(s,a) - \nabla_\theta \log \sum_b e^{h_\theta(s,b)} \\
& \nabla_\theta h_\theta(s,a) - \frac{1}{\sum_b e^{h_\theta(s,b)}} \nabla_\theta \sum_b e^{h_\theta(s,b)} \\
& \nabla_\theta h_\theta(s,a) - \frac{1}{\sum_b e^{h_\theta(s,b)}} \sum_b \nabla_\theta e^{h_\theta(s,b)} \\
& \nabla_\theta h_\theta(s,a) - \frac{1}{\sum_b e^{h_\theta(s,b)}} \sum_b e^{h_\theta(s,b)} \nabla_\theta h_\theta(s,b)
\end{aligned}$$

$\nabla_\theta \log \pi_\theta(a)$ for softmax policy

- Policy is the output of a softmax

$$\pi_\theta(a \mid s) = \frac{e^{h_\theta(s,a)}}{\sum_b e^{h_\theta(s,b)}}$$

$$\begin{aligned}
\nabla_\theta \log \pi_\theta(a) = & \nabla_\theta h_\theta(s,a) - \nabla_\theta \log \sum_b e^{h_\theta(s,b)} \\
& \nabla_\theta h_\theta(s,a) - \frac{1}{\sum_b e^{h_\theta(s,b)}} \nabla_\theta \sum_b e^{h_\theta(s,b)} \\
& \nabla_\theta h_\theta(s,a) - \frac{1}{\sum_b e^{h_\theta(s,b)}} \sum_b \nabla_\theta e^{h_\theta(s,b)} \\
& \nabla_\theta h_\theta(s,a) - \frac{1}{\sum_b e^{h_\theta(s,b)}} \sum_b e^{h_\theta(s,b)} \nabla_\theta h_\theta(s,b) \\
& \nabla_\theta h_\theta(s,a) - \sum_b \frac{e^{h_\theta(s,b)}}{\sum_b e^{h_\theta(s,b)}} \nabla_\theta h_\theta(s,b)
\end{aligned}$$

$\nabla_\theta \log \pi_\theta(a)$ for softmax policy

- Policy is the output of a softmax

$$\pi_\theta(a \mid s) = \frac{e^{h_\theta(s,a)}}{\sum_b e^{h_\theta(s,b)}}$$

$$\begin{aligned}
\nabla_\theta \log \pi_\theta(a) = & \nabla_\theta h_\theta(s,a) - \nabla_\theta \log \sum_b e^{h_\theta(s,b)} \\
& \nabla_\theta h_\theta(s,a) - \frac{1}{\sum_b e^{h_\theta(s,b)}} \nabla_\theta \sum_b e^{h_\theta(s,b)} \\
& \nabla_\theta h_\theta(s,a) - \frac{1}{\sum_b e^{h_\theta(s,b)}} \sum_b \nabla_\theta e^{h_\theta(s,b)} \\
& \nabla_\theta h_\theta(s,a) - \frac{1}{\sum_b e^{h_\theta(s,b)}} \sum_b e^{h_\theta(s,b)} \nabla_\theta h_\theta(s,b) \\
& \nabla_\theta h_\theta(s,a) - \sum_b \frac{e^{h_\theta(s,b)}}{\sum_b e^{h_\theta(s,b)}} \nabla_\theta h_\theta(s,b) \\
& \nabla_\theta h_\theta(s,a) - \sum_b \pi_\theta(s,b) \nabla_\theta h_\theta(s,b)
\end{aligned}$$

Policy gradients VS Evolutionary methods

Considers distribution over actions

$$\max_{\theta} . \; U(\theta) = \mathbb{E}_{\tau \sim P_\theta(\tau)} \left[R(\tau) \right]$$

$$\begin{aligned}
\nabla_\theta U(\theta) &= \nabla_\theta \mathbb{E}_{\tau \sim P_\theta(\tau)} \left[R(\tau) \right] \\
&= \nabla_\theta \sum_\tau P_\theta(\tau) R(\tau) \\
&= \sum_\tau \nabla_\theta P_\theta(\tau) R(\tau) \\
&= \sum_\tau P_\theta(\tau) \frac{\nabla_\theta P_\theta(\tau)}{P_\theta(\tau)} R(\tau) \\
&= \sum_\tau P_\theta(\tau) \nabla_\theta \log P_\theta(\tau) R(\tau) \\
&= \mathbb{E}_{\tau \sim P_\theta(\tau)} \left[\nabla_\theta \log P_\theta(\tau) R(\tau) \right]
\end{aligned}$$

Sample estimate:

$$\nabla_\theta U(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log P_\theta(\tau^{(i)}) R(\tau^{(i)})$$

Considers distribution over policy parameters

$$\max_{\mu} . \; U(\mu) = \mathbb{E}_{\theta \sim P_\mu(\theta)} \left[F(\theta) \right]$$

$$\begin{aligned}
\nabla_\mu U(\mu) &= \nabla_\mu \mathbb{E}_{\theta \sim P_\mu(\theta)} \left[F(\theta) \right] \\
&= \nabla_\mu \int P_\mu(\theta) F(\theta) d\theta \\
&= \int \nabla_\mu P_\mu(\theta) F(\theta) d\theta \\
&= \int P_\mu(\theta) \frac{\nabla_\mu P_\mu(\theta)}{P_\mu(\theta)} F(\theta) d\theta \\
&= \int P_\mu(\theta) \nabla_\mu \log P_\mu(\theta) F(\theta) d\theta \\
&= \mathbb{E}_{\theta \sim P_\mu(\theta)} \left[\nabla_\mu \log P_\mu(\theta) F(\theta) \right]
\end{aligned}$$

Sample estimate:

$$\nabla_\mu U(\mu) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\mu \log P_\mu(\theta^{(i)}) F(\theta^{(i)})$$

Variance

Here is our unbiased gradient estimator:

$$\hat{g} = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\alpha_t^{(i)} \mid s_t^{(i)}) G_t^{(i)}$$

The trace of the covariance matrix:

$$\mathrm{Var}(\hat{g}) = \mathrm{tr}\left(\mathbb{E}\left[(\hat{g} - \mathbb{E}[\hat{g}])(\hat{g} - \mathbb{E}[\hat{g}])^T \right] \right) = \sum_{k=1}^{n} \mathbb{E}\left[\left(\hat{g}_k - \mathbb{E}[\hat{g}_k] \right)^2 \right]$$

Our goal is to minimize the variance of our estimator

Remember that for random variable X: $\mathrm{Var}(aX) = a^2 \mathrm{Var}(x)$

Baseline choices

$$\hat{g} = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t^{(i)} \mid s_t^{(i)})\left(\sum_{k=t}^{H} R(s_k^{(i)}, a_k^{(i)}) - b\right)$$

- Constant baseline: $b = \mathbb{E}[R(\tau)]$
- Time-dependent baseline: $b_t = \sum_{i=1}^{N}\sum_{k=t}^{H} R(s_k^{(i)}, a_k^{(i)})$

Baseline choices

$$\hat{g} = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t^{(i)} \mid s_t^{(i)})\left(\sum_{k=t}^{H} R(s_k^{(i)}, a_k^{(i)}) - b\right)$$

- Constant baseline: $b = \mathbb{E}[R(\tau)]$
- Time-dependent baseline: $b_t = \sum_{i=1}^{N}\sum_{k=t}^{H} R(s_k^{(i)}, a_k^{(i)})$
- State-dependent expected return:

$$b(s_t) = \mathbb{E}\left[r_t + r_{t+1} + r_{t+2} + \ldots + r_{H-1}\right] = V_\pi(s_t)$$

$$\hat{g} = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^{(i)} \mid s_t^{(i)})(G_t^i - b(s_t^{(i)}))$$

Variance

$$\hat{g} = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^{(i)} \mid s_t^{(i)})(G_t^i - b(s_t^{(i)}))$$

$$\mathrm{Var}(\hat{g}) = \mathrm{tr}\left(\mathbb{E}\left[(\hat{g} - \mathbb{E}[\hat{g}])(\hat{g} - \mathbb{E}[\hat{g}])^T\right]\right) = \sum_{k=1}^{n}\mathbb{E}\left[\left(\hat{g}_k - \mathbb{E}[\hat{g}_k]\right)^2\right]$$

- Imagine in some state S_1 the rewards of all actions are ~3000 and in S_2 ~ -4000. The variance is large of this vector.
- Now imagine you have b(S_1)=3000 and b(S_2)=-4000. I have much decreased the variance.
- We want to encourage an action, not when it has high return, but when it have higher return thatn the state expected return b(s), i.e., when making this action MORE probable would allow me to improve over what I have now.

Estimate $V_\pi(s_t)$

$$\hat{g} = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t^{(i)} \mid s_t^{(i)})\left(\sum_{k=t}^{H} R(s_k^{(i)}, a_k^{(i)}) - V^\pi(s_k^{(i)})\right)$$

MC estimation

Initialize ϕ

- Collect trajectories $\tau_1, \ldots \tau_N$
- Regress against empirical return:

$$\phi_{i+1} \leftarrow \arg\min_\phi \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{H-1}\left(V_\phi^\pi(s_t^{(i)}) - \left(\sum_{k=t}^{H-1} R(s_k^{(i)}, u_k^{(i)})\right)\right)^2$$

Estimate $V_\pi(s_t)$

$$\hat{g} = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t^{(i)} \mid s_t^{(i)})\left(\sum_{k=t}^{H} R(s_k^{(i)}, a_k^{(i)}) - V^\pi(s_k^{(i)})\right)$$

TD estimation

Initialize ϕ

- Collect data {s, u, s', r}
- Fitted V iteration:

$$\phi_{i+1} \leftarrow \min_\phi \sum_{(s,u,s',r)} \|r + V_{\phi_i}^\pi(s') - V_\phi(s)\|_2^2 + \lambda\|\phi - \phi_i\|_2^2$$

Bootstrapping!

Better estimates for cumulative future reward

$$\hat{g} = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t^{(i)} \mid s_t^{(i)})\left(\sum_{k=t}^{H} R(s_k^{(i)}, a_k^{(i)}) - V^\pi(s_k^{(i)})\right)$$

We are essentially attempting to estimate Q from a single rollout:

$$Q^\pi(s,a) = \mathbb{E}[R_0 + R_1 + \cdots \mid S_0 = s, A_0 = a]$$

Minimize variance by:

- discounting
- introducing a learnt approximation for the extected return (critic), as opposed to use MC samples

Actor-Critic

Reducing variance using a critic

$$Q^{\pi,\gamma}(s,a) = \mathbb{E}[R_0 + \gamma R_1 + \gamma^2 R_2 \cdots \mid S_0 = s, A_0 = a]$$

Actor-Critic

Reducing variance using a critic

$$
\begin{aligned}
Q^{\pi,\gamma}(s,a) &= \mathbb{E}[R_0 + \gamma R_1 + \gamma^2 R_2 \cdots \,|\, S_0 = s, A_0 = a] \\
&= \mathbb{E}[R_0 + \gamma V^{\pi}(S_1) \,|\, S_0 = s, A_0 = a]
\end{aligned}
$$

Actor-Critic

Reducing variance using a critic

$$
\begin{aligned}
Q^{\pi,\gamma}(s,a) &= \mathbb{E}[R_0 + \gamma R_1 + \gamma^2 R_2 \cdots \,|\, S_0 = s, A_0 = a] \\
&= \mathbb{E}[R_0 + \gamma V^{\pi}(S_1) \,|\, S_0 = s, A_0 = a] \\
&= \mathbb{E}[R_0 + \gamma R_1 + \gamma^2 V^{\pi}(S_2) \,|\, S_0 = s, A_0 = a]
\end{aligned}
$$

Actor-Critic

Reducing variance using a critic

$$
\begin{aligned}
Q^{\pi,\gamma}(s,a) &= \mathbb{E}[R_0 + \gamma R_1 + \gamma^2 R_2 \cdots \,|\, S_0 = s, A_0 = a] \\
&= \mathbb{E}[R_0 + \gamma V^{\pi}(S_1) \,|\, S_0 = s, A_0 = a] \\
&= \mathbb{E}[R_0 + \gamma R_1 + \gamma^2 V^{\pi}(S_2) \,|\, S_0 = s, A_0 = a] \\
&= \mathbb{E}[R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 V^{\pi}(S_3) \,|\, S_0 = s, A_0 = a] \\
&= \ldots
\end{aligned}
$$

Actor-Critic

- Monte-Carlo policy gradient still has high variance
- We can use a critic to estimate the action-value function:

$$Q_w(s,a) \approx Q^{\pi_\theta}(s,a)$$

- Actor-critic algorithms maintain two sets of parameters
  - Critic Updates action-value function parameters w
  - Actor Updates policy parameters θ, in direction suggested by critic

REINFORCE/Actor-critic training

- Stability of training neural networks requires the **gradient updates to be de-correlated**
- This is not the case if data arrives **sequentially**
- Gradient updates computed from some part of the space can cause the value (Q) function approximator to **oscillate**
- Our solution so far has been: **Experience buffers** where experience tuples are mixed and sampled from. Resulting sampled batches are more stationary that the ones encountered online (without buffer)
- This limits deep RL to **off-policy** methods, since data from an older policy are used to update the weights of the value approximator (critic) (except if we take care and weight such data under our current stochastic policy->importance sampling)

Asynchronous Deep RL for on policy learning

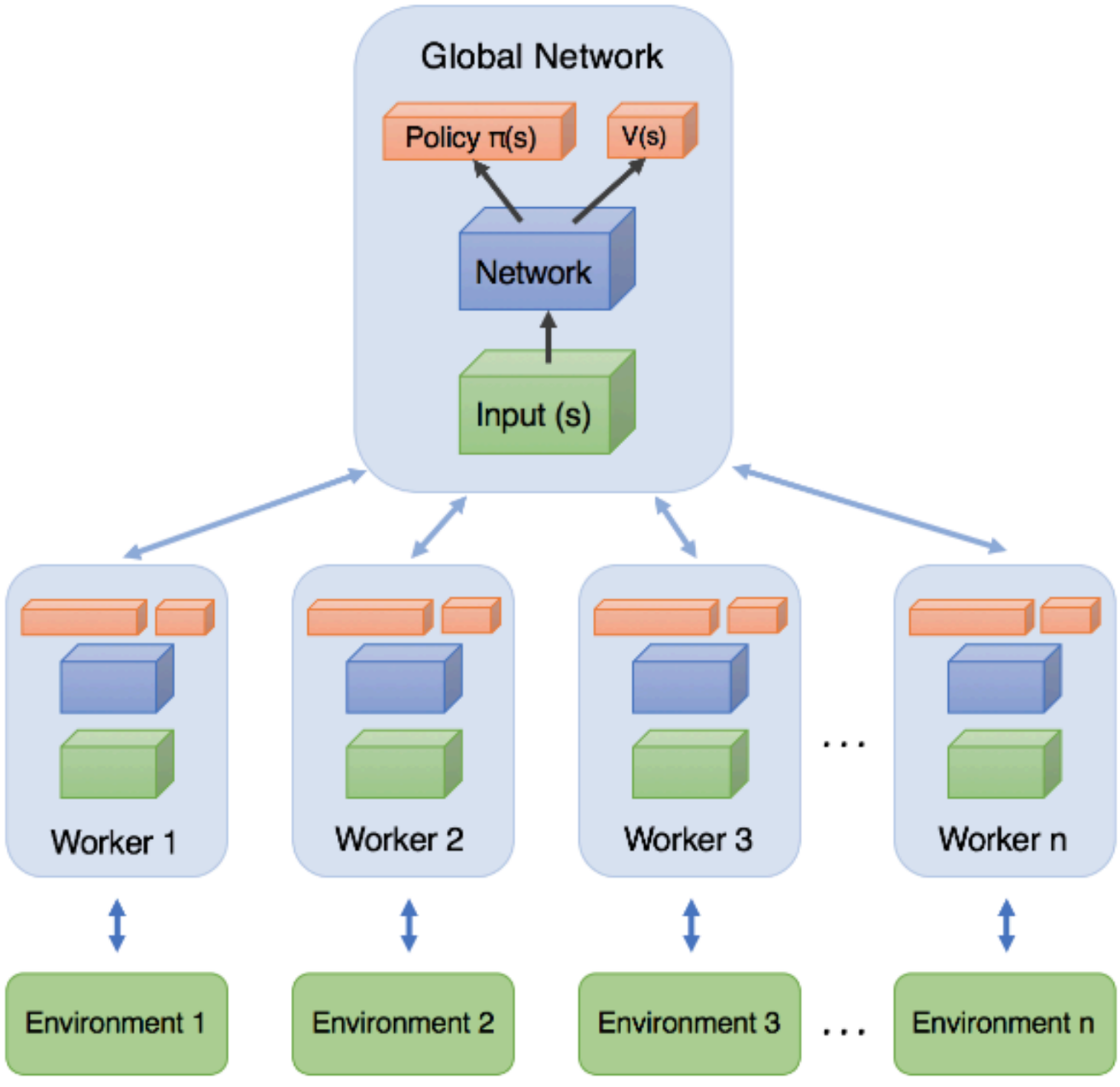
Asynchronous Methods for Deep Reinforcement Learning

Volodymyr Mnih[1] VMNIH@GOOGLE.COM
Adrià Puigdomènech Badia[1] ADRIAP@GOOGLE.COM
Mehdi Mirza[1,2] MIRZAMOM@IRO.UMONTREAL.CA
Alex Graves[1] GRAVESA@GOOGLE.COM
Tim Harley[1] THARLEY@GOOGLE.COM
Timothy P. Lillicrap[1] COUNTZERO@GOOGLE.COM
David Silver[1] DAVIDSILVER@GOOGLE.COM
Koray Kavukcuoglu [1] KORAYK@GOOGLE.COM

[1] Google DeepMind
[2] Montreal Institute for Learning Algorithms (MILA), University of Montreal

- Alternative: parallelize the collection of experience and stabilize training **without experience buffers**
- **Multiple threads of experience**, one per agent, each exploring in different part of the environment contributing experience tuples
- **Different exploration strategies** (e.g., various ϵ values) in different threads increase diversity
- Now you can train on-policy using any of our policy gradient methods

Distributed RL



Algorithm S3 Asynchronous advantage actor-critic - pseudocode for each actor-learner thread.

// Assume global shared parameter vectors θ *and* θ_v *and global shared counter* $T = 0$
// Assume thread-specific parameter vectors θ' *and* θ'_v
Initialize thread step counter $t \leftarrow 1$
repeat
  Reset gradients: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.
  Synchronize thread-specific parameters $\theta' = \theta$ and $\theta'_v = \theta_v$ — Copying the weights
  $t_{start} = t$
  Get state s_t
  repeat
    Perform a_t according to policy $\pi(a_t|s_t;\theta')$ — Rollout
    Receive reward r_t and new state s_{t+1}
    $t \leftarrow t + 1$
    $T \leftarrow T + 1$
  until terminal s_t **or** $t - t_{start} == t_{max}$
  $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t, \theta'_v) & \text{for non-terminal } s_t \text{// Bootstrap from last state} \end{cases}$
  for $i \in \{t-1, \ldots, t_{start}\}$ **do**
    $R \leftarrow r_i + \gamma R$
    Accumulate gradients wrt θ': $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i;\theta')(R - V(s_i;\theta'_v))$ — Advantage
    Accumulate gradients wrt θ'_v: $d\theta_v \leftarrow d\theta_v + \partial (R - V(s_i;\theta'_v))^2 / \partial \theta'_v$ — Learning the critic
  end for
  Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$.
until $T > T_{max}$

s_1, s_2, s_3, s_4

r_1, r_2, r_3

What is the approximation used for the advantage?

$R_3 - V(s_3)$

$R_3 = r_3 + \gamma V(s_4, \theta'_v)$ $\quad A_3 = R_3 - V(s_3; \theta'_v)$

$R_2 = r_2 + \gamma r_3 + \gamma^2 V(s_4, \theta'_v)$ $\quad A_2 = R_2 - V(s_2; \theta'_v)$

Algorithm S3 Asynchronous advantage actor-critic - pseudocode for each actor-learner thread.

// Assume global shared parameter vectors θ *and* θ_v *and global shared counter* $T = 0$
// Assume thread-specific parameter vectors θ' *and* θ'_v
Initialize thread step counter $t \leftarrow 1$
repeat
Reset gradients: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.
Synchronize thread-specific parameters $\theta' = \theta$ and $\theta'_v = \theta_v$

``We also found that adding the entropy of the policy π to the objective function improved exploration by discouraging premature convergence to suboptimal deterministic policies." So you need to add to the policy gradient: $+\beta \nabla_\theta \mathrm{H}(\pi_\theta(a_t \mid s_t; \theta))$

We will look into the entropy as part of the reward in later lecture

Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$.
until $T > T_{max}$

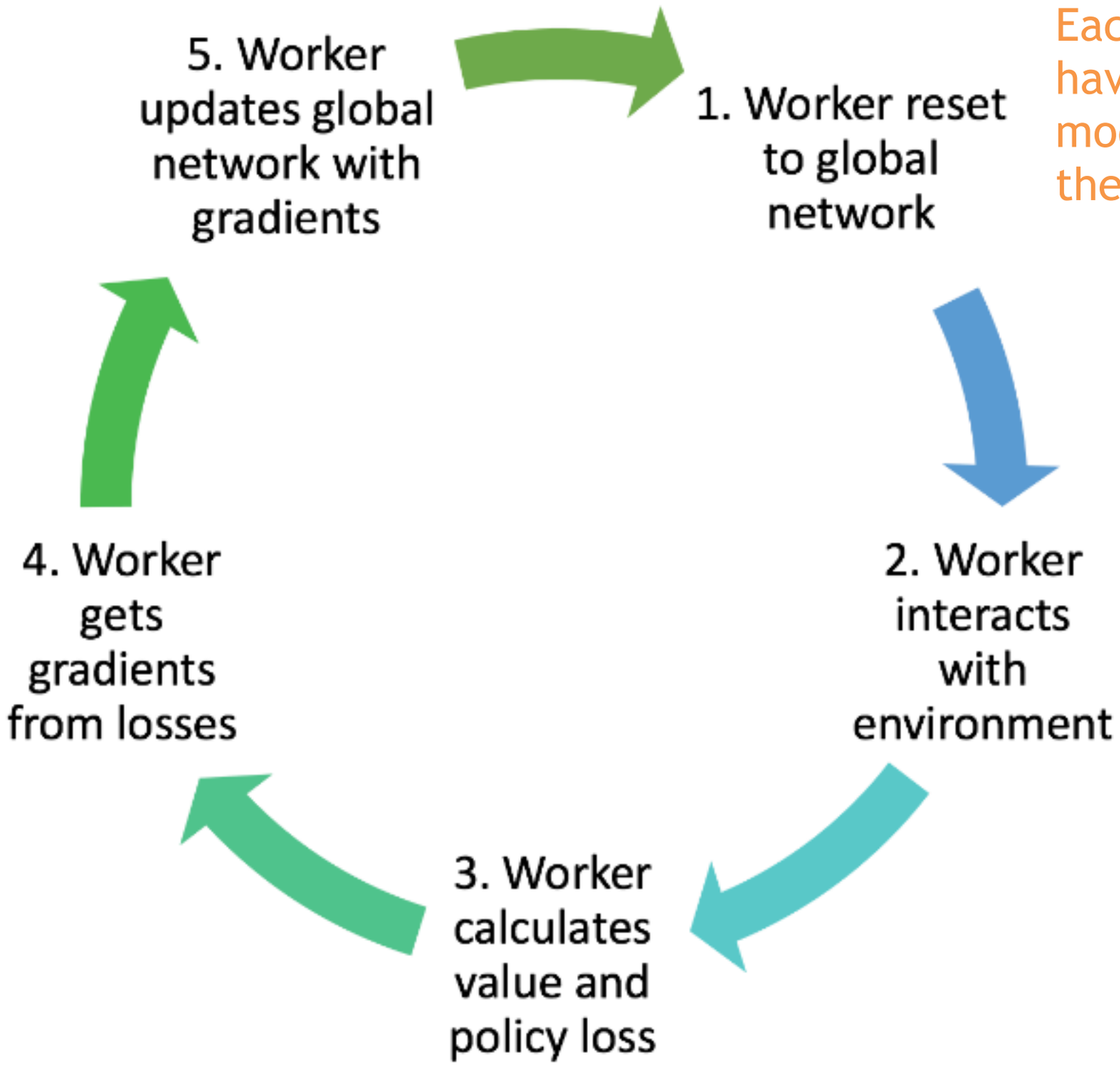
What is the approximation used for the advantage?

$R_3 = r_3 + \gamma V(s_4, \theta'_v)$ $\quad A_3 = R_3 - V(s_3; \theta'_v)$

$R_2 = r_2 + \gamma r_3 + \gamma^2 V(s_4, \theta'_v)$ $\quad A_2 = R_2 - V(s_2; \theta'_v)$

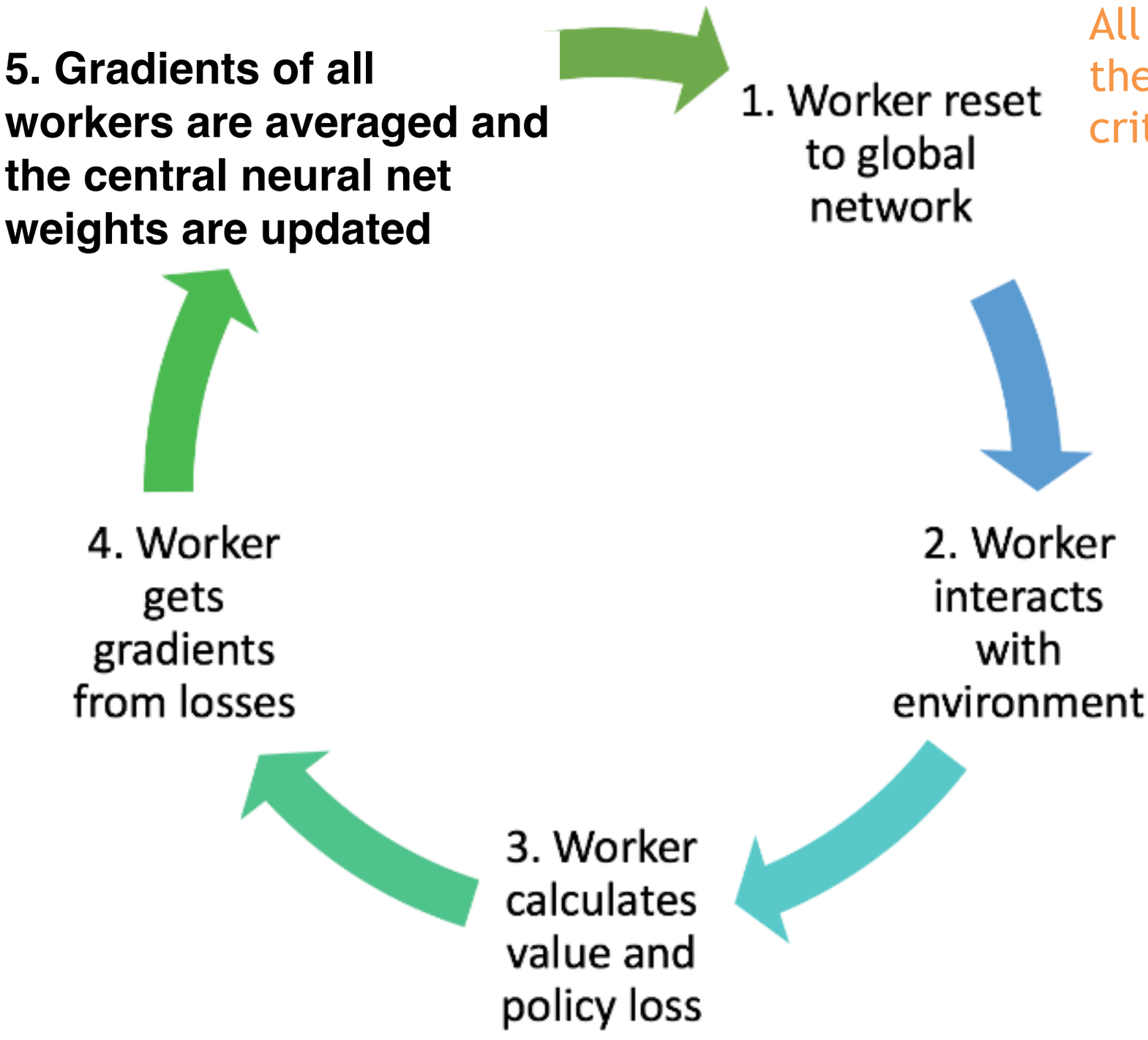
Distributed Asynchronous RL-A3C

No locking

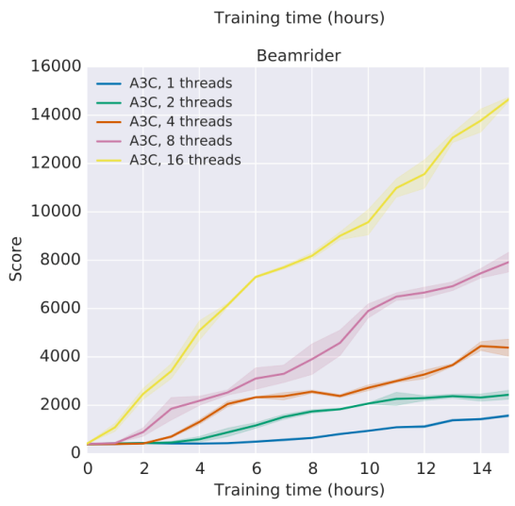


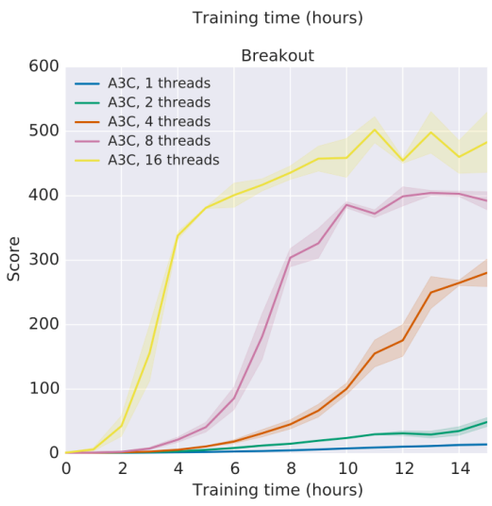
Each worker may have slightly modified version of the policy/critic

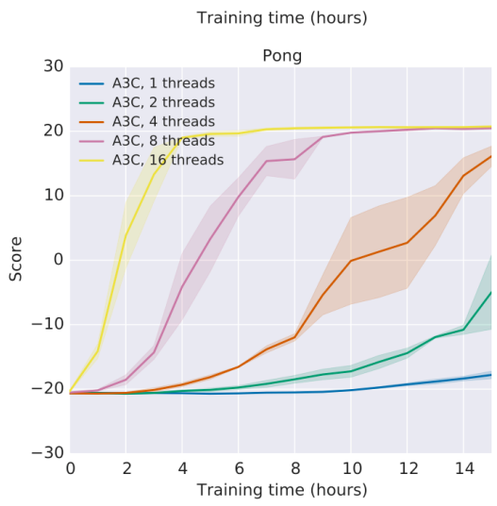
Distributed Synchronous RL-A2C

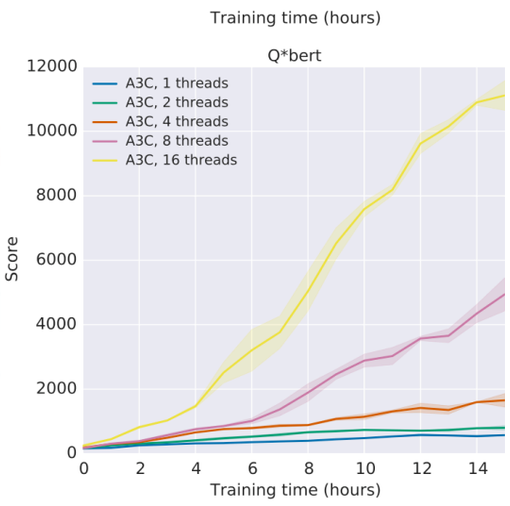


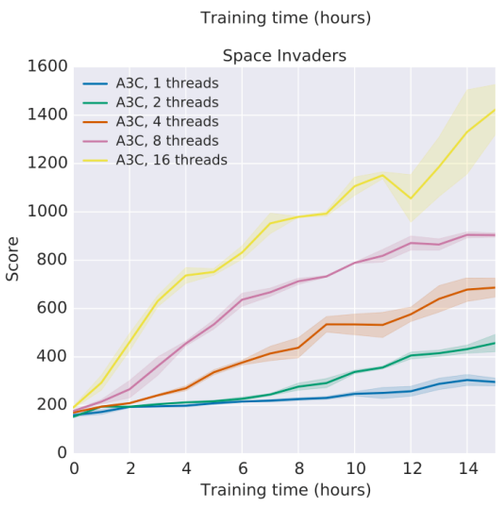
Advantages of Asynchronous (multi-threaded) RL











Summary of policy gradients so far

- The policy gradient has many equivalent forms

$$\begin{aligned}
\nabla_\theta U(\theta) &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ v_t\right] && \text{REINFORCE} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ Q^w(s,a)\right] && \text{Q Actor-Critic} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ A^w(s,a)\right] && \text{Advantage Actor-Critic}
\end{aligned}$$

- Each leads a stochastic gradient ascent algorithm

- Critic uses policy evaluation (e.g. MC or TD learning) to estimate

$$Q^\pi(s,a),\ A^\pi(s,a) \text{ or } V^\pi(s)$$