Lecture 9:
Actor Critic Methods

B. Ravindran

Recall - REINFORCE: MC PG

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta})$$

$$= \mathbb{E}_\pi\left[\sum_a q_\pi(S_t,a) \nabla \pi(a|S_t,\boldsymbol{\theta})\right]$$

Expectation over the visited states while following Π

$$= \mathbb{E}_\pi\left[q_\pi(S_t,A_t) \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}\right]$$

Multiply and divide by Π(a|s,θ) +

Expectation over the actions taken while following Π

$$= \mathbb{E}_\pi\left[G_t \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}\right]$$

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha G_t \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)}$$

The gradient update is

- proportional to the return
- inversely proportional to the action probability

REINFORCE w/ Baseline

- The policy gradient theorem can be generalized to include a comparison of the action value to an arbitrary baseline *b(s)*

$$\boxed{\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a \Big(q_\pi(s,a) - b(s)\Big) \nabla \pi(a|s, \boldsymbol{\theta})}$$

- Baseline - should be a function that is independent of the action *a*

$$\sum_a b(s) \nabla \pi(a|s, \boldsymbol{\theta}) \;=\; b(s) \nabla \sum_a \pi(a|s, \boldsymbol{\theta}) \;=\; b(s) \nabla 1 \;=\; 0$$

- Update rule of REINFORCE with baseline

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha \Big(G_t - b(S_t)\Big) \frac{\nabla \pi(A_t|S_t, \boldsymbol{\theta}_t)}{\pi(A_t|S_t, \boldsymbol{\theta}_t)}$$

(Doesn't change the expected value, but has an effect on the variance)

Recall - Actor-Critic Methods

- Actor-Critic methods learn both a policy and a state-value function simultaneously
- The policy is referred to as the actor that suggests actions given a state
- The value function is referred to as the critic. It evaluates actions taken by the actor based on the given policy

$$
\begin{aligned}
\boldsymbol{\theta}_{t+1} &\doteq \boldsymbol{\theta}_t + \alpha \left(G_{t:t+1} - \hat{v}\left(S_t, \mathbf{w}\right) \right) \frac{\nabla \pi\left(A_t \mid S_t, \boldsymbol{\theta}_t\right)}{\pi\left(A_t \mid S_t, \boldsymbol{\theta}_t\right)} \\
&= \boldsymbol{\theta}_t + \alpha \left(R_{t+1} + \gamma \hat{v}\left(S_{t+1}, \mathbf{w}\right) - \hat{v}\left(S_t, \mathbf{w}\right) \right) \frac{\nabla \pi\left(A_t \mid S_t, \boldsymbol{\theta}_t\right)}{\pi\left(A_t \mid S_t, \boldsymbol{\theta}_t\right)} \\
&= \boldsymbol{\theta}_t + \alpha \delta_t \frac{\nabla \pi\left(A_t \mid S_t, \boldsymbol{\theta}_t\right)}{\pi\left(A_t \mid S_t, \boldsymbol{\theta}_t\right)}
\end{aligned}
$$

Recall - One-step Actor-Critic

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)
Loop forever (for each episode):
    Initialize S (first state of episode)
    $I \leftarrow 1$
    Loop while S is not terminal (for each time step):
        $A \sim \pi(\cdot|S, \boldsymbol{\theta})$
        Take action A, observe S', R
        $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S,\mathbf{w})$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla \ln \pi(A|S, \boldsymbol{\theta})$
        $I \leftarrow \gamma I$
        $S \leftarrow S'$

(This is a fully online, incremental algorithm, with states, actions, and rewards processed as they occur and then never revisited again)

Comparison to REINFORCE

REINFORCE w/ baseline

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha\Big(G_t - b(S_t)\Big)\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)}$$

- ❑ G_t - unbiased estimate but high variance
- ❑ Recall $\mathbb{E}_\pi[G_t \mid S_t, A_t] = q_\pi(S_t, A_t)$
- ❑ Need a good estimate of $q_\pi(S_t, A_t)$ with less variance than G_t
- ❑ In 1-step Actor-Critic, we use $\hat{v}$ for both estimating $q_\pi(S_t, A_t)$ and as the baseline
- ❑ This introduces bias, but lesser variance - Leads to better performance

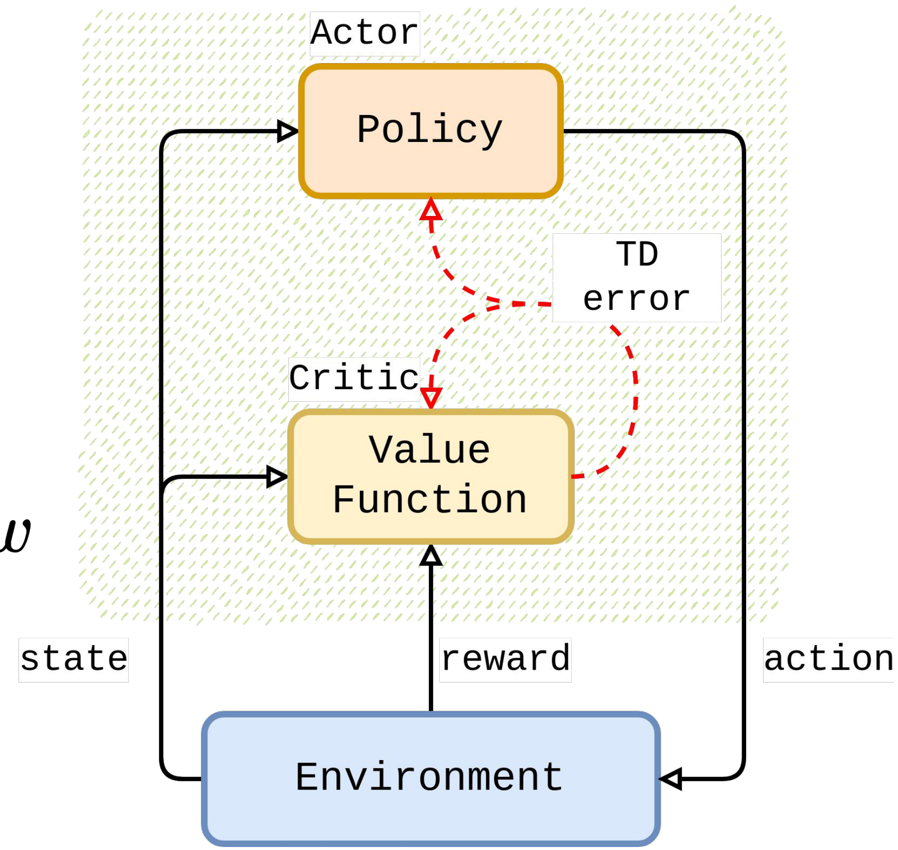
1-step Actor-Critic

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha\Big(R_{t+1} + \gamma\hat{v}(S_{t+1},\mathbf{w}) - \hat{v}(S_t,\mathbf{w})\Big)\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)}$$

Common Features of AC Methods

- Actor: Computes the policy π_θ and updates θ

- Critic: Computes an estimate $\hat{v}(s, w)$ of the state value function. Updates the parameter w



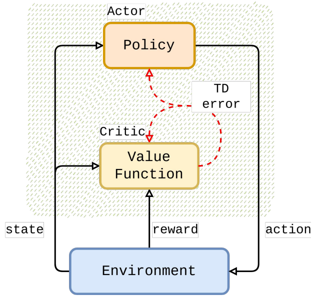
Basic Actor Critic Algorithm

1. Take action $\mathbf{a} \sim \pi_\theta(\mathbf{a} \mid \mathbf{s})$ and receive $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$

2. Update value parameter $\mathbf{w}$ using data $(\mathbf{s}, r + \gamma \hat{v}(\mathbf{s}', \mathbf{w}))$

3. Compute $\hat{\delta}(\mathbf{s}, \mathbf{a}) = r + \gamma \hat{v}(\mathbf{s}', \mathbf{w}) - \hat{v}(\mathbf{s}, \mathbf{w})$

4. $\theta \leftarrow \theta + \alpha \cdot \hat{\delta}(\mathbf{s}, \mathbf{a}) \cdot \nabla_\theta \log \pi_\theta(\mathbf{a} \mid \mathbf{s})$



Critic Update

1. Take action $\mathbf{a} \sim \pi_\theta(\mathbf{a} \mid \mathbf{s})$ and receive $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$
2. Update value parameter $\mathbf{w}$ using data $(\mathbf{s}, r + \gamma \hat{v}(\mathbf{s}', \mathbf{w}))$
3. Compute $\hat{\delta}(\mathbf{s}, \mathbf{a}) = r + \gamma \hat{v}(\mathbf{s}', \mathbf{w}) - \hat{v}(\mathbf{s}, \mathbf{w})$
4. $\theta \leftarrow \theta + \alpha \cdot \hat{\delta}(\mathbf{s}, \mathbf{a}) \cdot \nabla_\theta \log \pi_\theta(\mathbf{a} \mid \mathbf{s})$

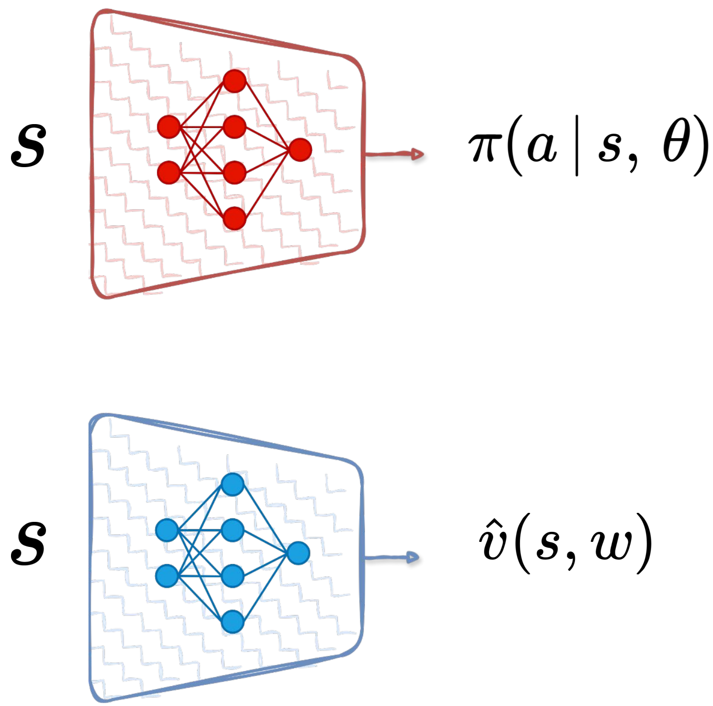
- Step 2 of the previous algorithm usually happens in batches
- Minimize the squared loss:

$$L(\mathbf{w}) = \frac{1}{N} \sum_i \left\| \hat{v}(\mathbf{s}_i, \mathbf{w}) - y_i \right\|^2$$

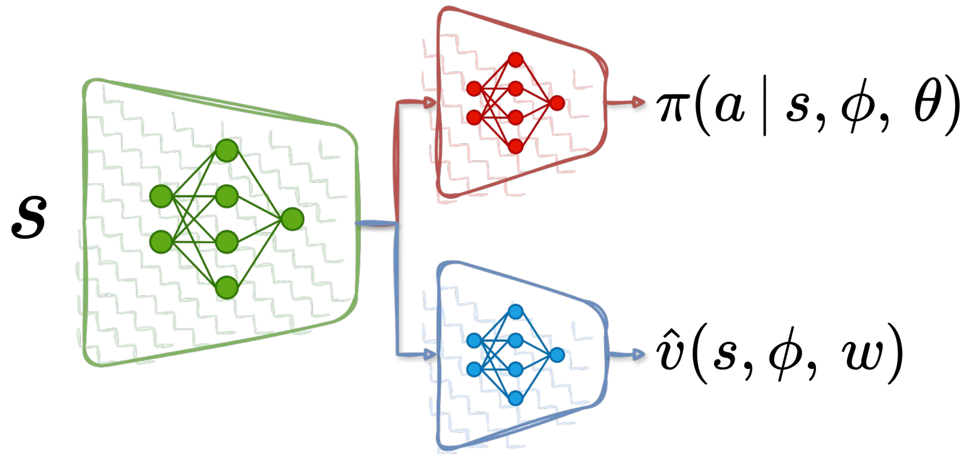
Batch Size

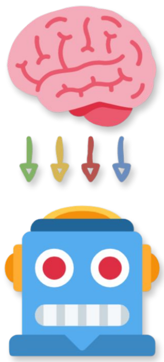
Design Choices

Two Network Design



Shared Network Design





Advantage Actor-Critic Methods

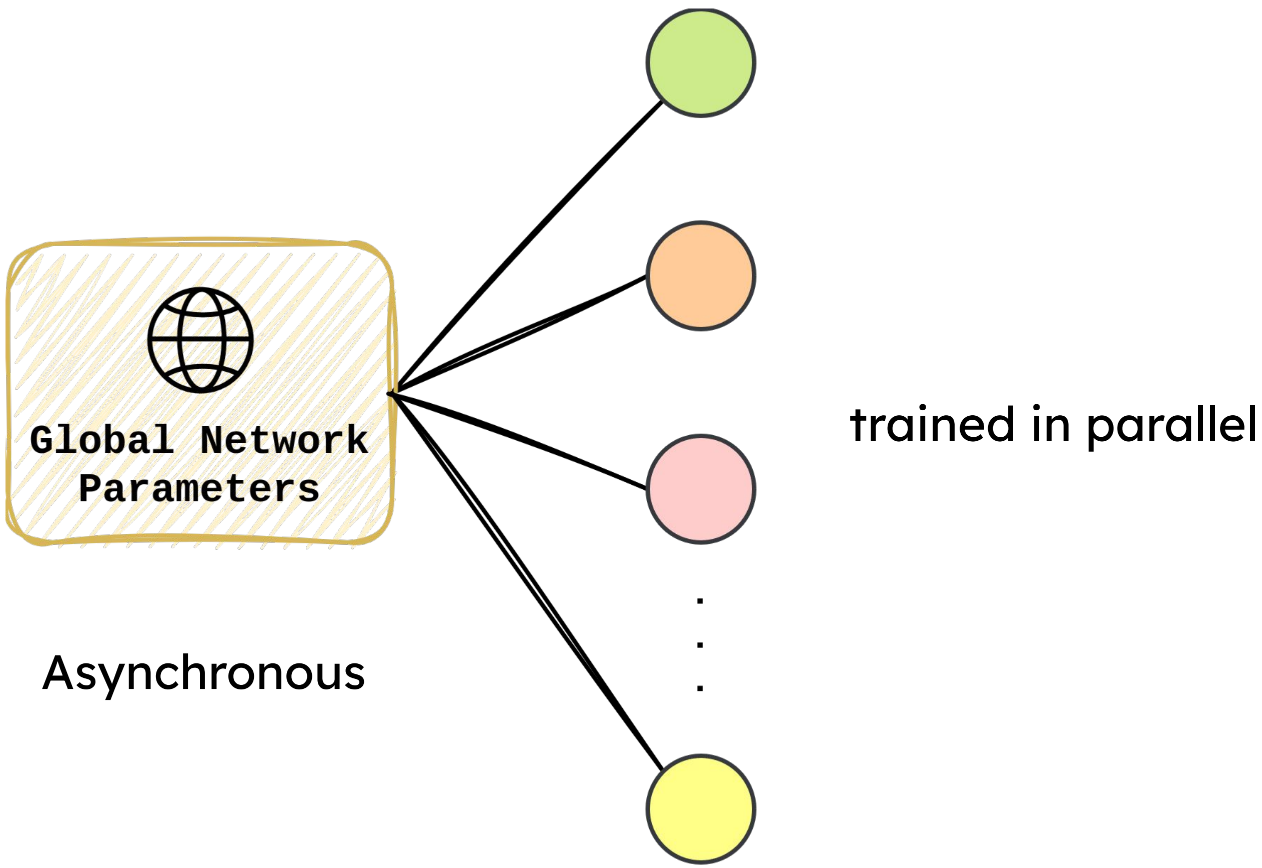
Recall - Advantage Function

- The advantage function is the difference between the q-value and the value function

- It can be interpreted as a measure of the advantage of taking action *a* in state *s* as compared to following policy π

$$A_\pi(\mathbf{s}, \mathbf{a}) = q_\pi(\mathbf{s}, \mathbf{a}) - v_\pi(\mathbf{s})$$

A3C – Asynchronous Advantage Actor Critic



A3C Algorithm

// *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 $\theta'_v = \theta_v$
    $t_{start} = t$
    Get state s_t
    repeat
        Perform a_t according to policy $\pi(a_t|s_t;\theta')$
        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': $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i;\theta')(R - V(s_i;\theta'_v))$
        Accumulate gradients wrt θ'_v: $d\theta_v \leftarrow d\theta_v + \partial\left(R - V(s_i;\theta'_v)\right)^2 / \partial\theta'_v$
    end for
    Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$.
until $T > T_{max}$

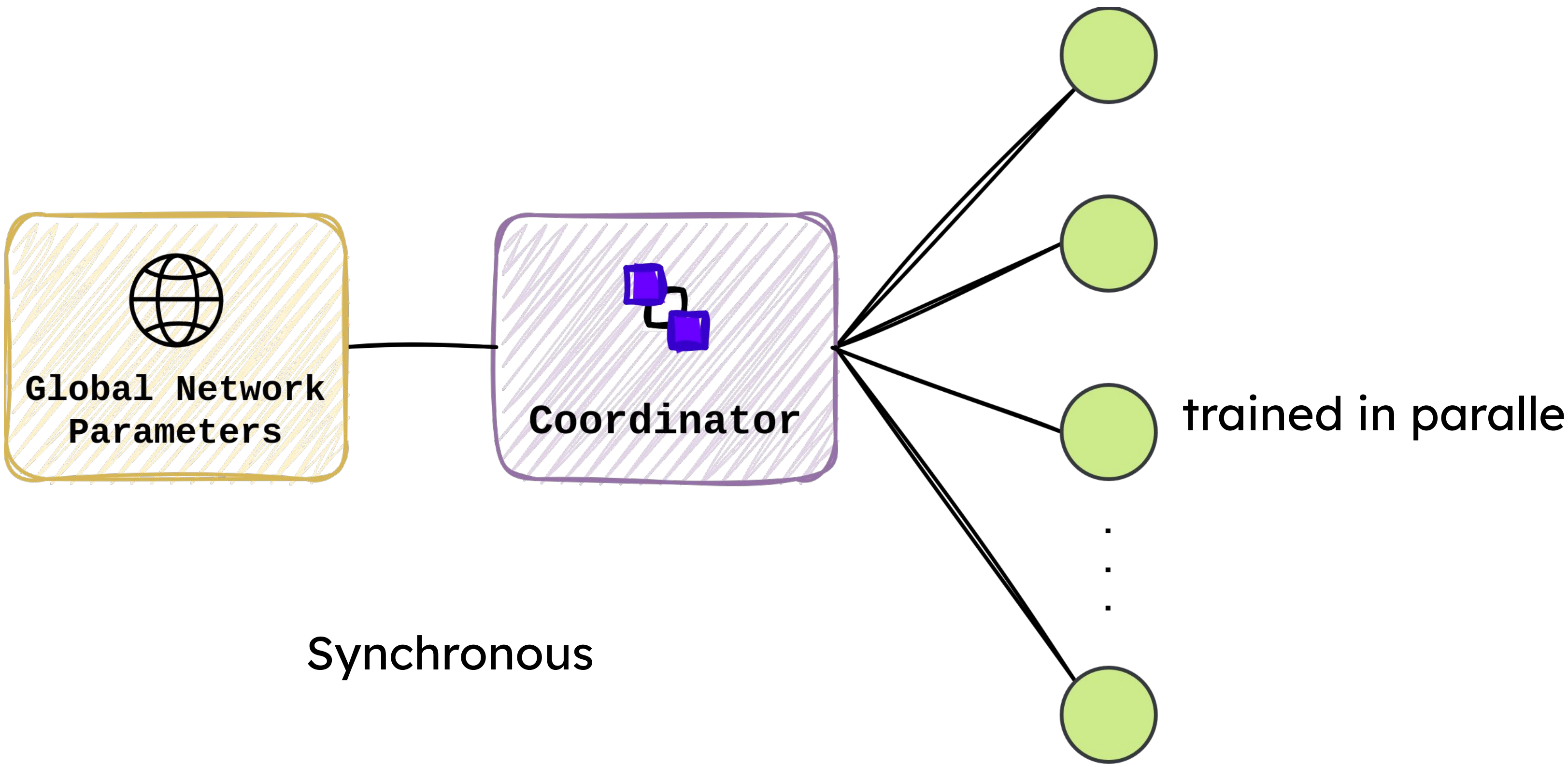
Reset thread params, update local params with global params

Gather experience

Compute the gradients for this thread

Update global params

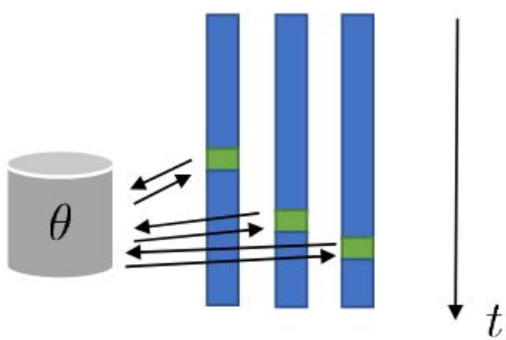
A2C – Synchronous Advantage Actor Critic



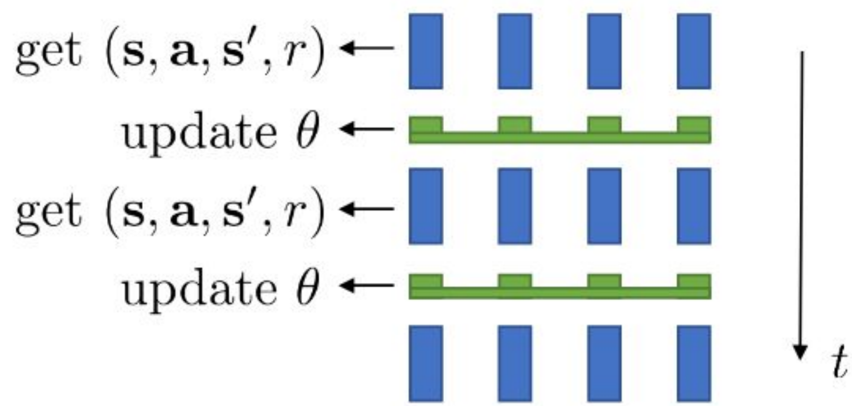
A3C vs A2C

- We remove the "asynchronous" part of A3C
- The updates to the global parameters are executed only after all the threads have finished their computation

asynchronous parallel actor-critic



synchronized parallel actor-critic



Compatible Parametrization

- ❑ Substituting the approximation $\hat{q}(s, a, w)$ instead of the true value of $q_\pi(s, a)$ may introduce bias
- ❑ It can be proved that there is no bias if the function approximator has a "compatible" parametrization with the policy parametrization

- ❑ Condition 1: projection of characteristic eligibility

$$\hat{q}(\mathbf{s}, \mathbf{a}, \mathbf{w}) = \nabla_\theta \log \pi_\theta(\mathbf{a} \mid \mathbf{s})^T \mathbf{w}$$

- ❑ Condition 2: minimize mean squared error

$$\mathbf{w} = \arg\min \mathbb{E}_{\mathbf{s} \sim \rho^{\pi_\theta}, \mathbf{a} \sim \pi_\theta} \left[\left(\hat{q}(\mathbf{s}, \mathbf{a}, \mathbf{w}) - q_{\pi_\theta}(\mathbf{s}, \mathbf{a}) \right)^2 \right]$$