Actor Critic Methods

B. Ravindran

Recap

- 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 estimated value function is referred to as the critic. It evaluates actions taken by the actor based on the given policy.

✓ b.

Characteristic Eligibility

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

Recap: One Step Actor Critic

One-step Actor–Critic (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

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 $\hat{v}(S',\mathbf{w}) \doteq 0$)
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S,\mathbf{w})$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla \ln \pi(A|S, \boldsymbol{\theta})$
        $I \leftarrow \gamma I$
        $S \leftarrow S'$

Comparison to REINFORCE

MC Policy Gradient

- Recall the REINFORCE(with baseline) update:

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

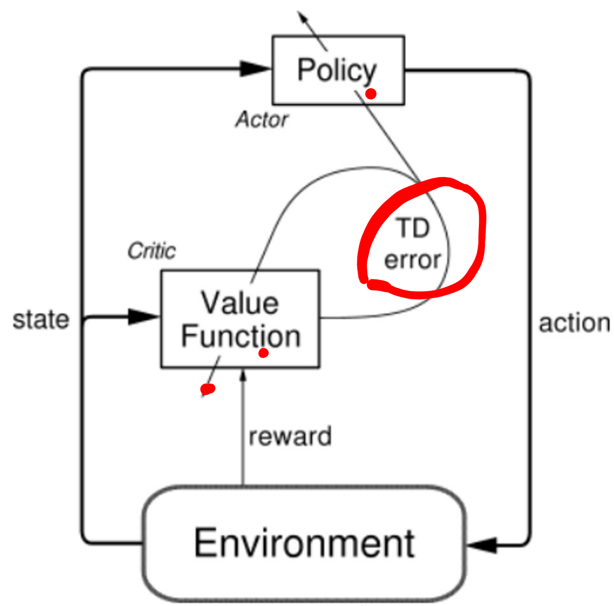
- The G_t term (although unbiased) causes the high variance of the algorithm.
- Recall that $\mathbb{E}_\pi[G_t|S_t, A_t] = q_\pi(S_t, A_t)$.
- If we had an estimate of $q_\pi(S_t, A_t)$ with less variance, then we can use that instead of G_t.

Comparison to REINFORCE

- In the one step AC algorithm, we use $\hat{v}$ for both estimating $q_\pi(S_t, A_t)$ and as the baseline.
- The bootstrapping in the update introduces bias but decreases the variance.
- This reduced variance can accelerate learning.

Common Features of AC Methods

- Actor: Computes the policy π_θ and updates θ.
- Critic: Typically computes an estimate $\hat{v}(\mathbf{s}, \mathbf{w})$ of the state value function . Updates the parameter $\mathbf{w}$.



$\pi_\theta \to \pi^*$

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 $(\mathbf{s}, \underbrace{r + \gamma \hat{v}(\mathbf{s}', \mathbf{w})})$
3. Compute $\hat{\delta}(\mathbf{s}, \mathbf{a}) = r + \gamma \hat{v}(\mathbf{s}', \mathbf{w}) - \hat{v}(\mathbf{s}, \mathbf{w})$ $\langle x, f(x) \rangle$
4. $\theta \leftarrow \theta + \alpha \cdot \hat{\delta}(\mathbf{s}, \mathbf{a}) \cdot \nabla_\theta \log \pi_\theta(\mathbf{a} \mid \mathbf{s})$ $\leftarrow$ $\theta + \alpha \cdot A(s, a) \cdot \nabla_\theta \log \pi$
5. Go back to step 1

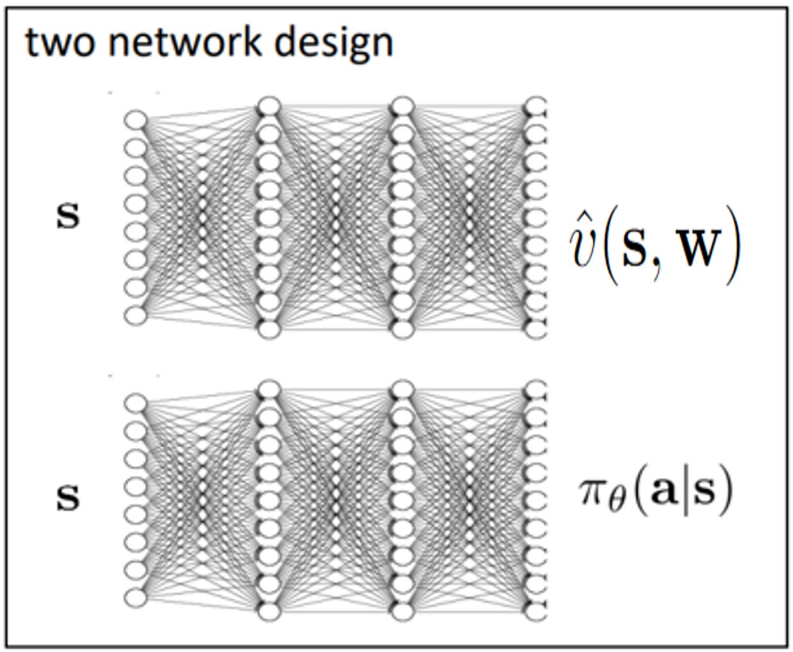
How is the critic updated?

- Step 2 of the previous algorithm usually happens in batches. We get multiple data points of the form $(\mathbf{s}, y)$ from parallel workers.
- Minimize the squared loss:

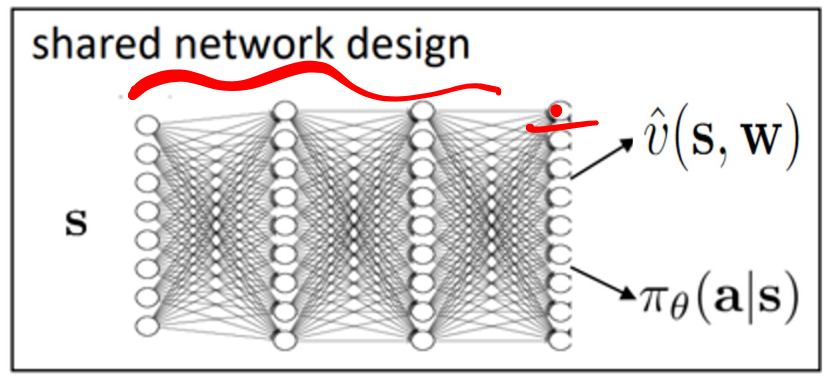
$$L(\mathbf{w}) = \frac{1}{N} \sum_i \|\hat{v}(\mathbf{s}_i, \mathbf{w}) - y_i\|^2$$

- N is the batch size

Design Choices



θ / w



Advantage Function

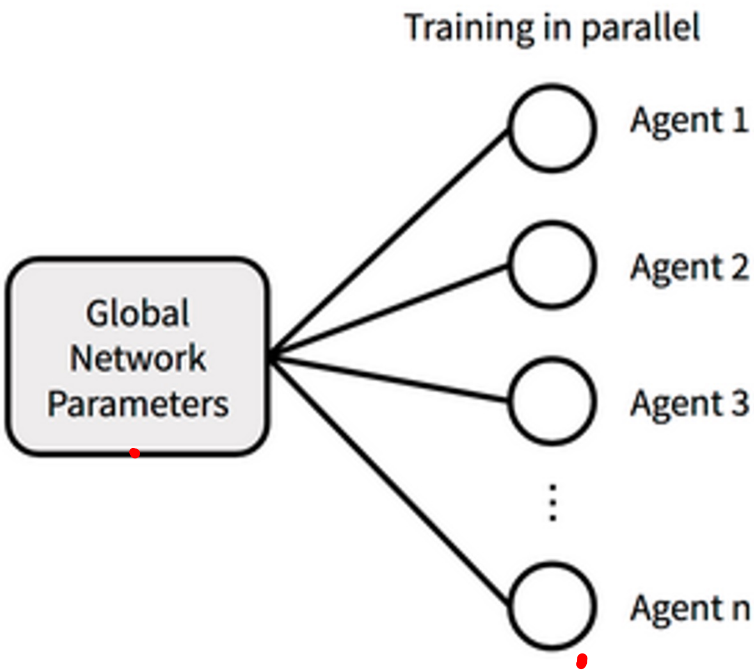
- The advantage function is the difference between the q-value and the value function.
- It can be interpreted as a measure of the advantage of taking action $\mathbf{a}$ in state $\mathbf{s}$ as compared to following policy π

$$\overset{A}{\cancel{\delta}}_{\pi}(\mathbf{s}, \mathbf{a}) = q_{\pi}(\mathbf{s}, \mathbf{a}) - v_{\pi}(\mathbf{s})$$

||

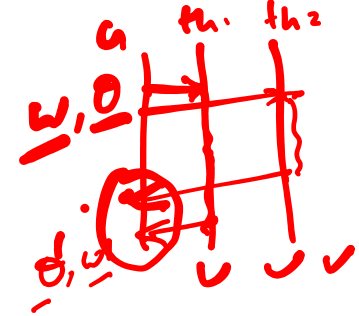
$$\delta(s, a) = r_{t+1} + \gamma v_{\pi}(s') - v_{\pi}(s)$$

A3C – Asynchronous Advantage Actor Critic



A3C (Async)

A3C - Mnih et. al. 2016



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

// Assume global shared parameter vectors θ and w and global shared counter $T = 0$
// Assume thread-specific parameter vectors θ' and w'
Initialize thread step counter $t \leftarrow 1$
repeat
  Reset gradients: $d\theta \leftarrow 0$ and $d\mathrm{w} \leftarrow 0$.
  Synchronize thread-specific parameters $\theta' = \theta$ and $\mathrm{w}' = \mathrm{w}$
  $t_{start} = t$
  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** $t - t_{start} == t_{max}$
  $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t, \mathrm{w}') & \text{for non-terminal } s_t \text{// Bootstrap from last state} \end{cases}$
  for $i \in \{t-1, \ldots, t_{start}\}$ **do**
    $R \leftarrow r_i + \gamma R$
    Accumulate gradients wrt θ': $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i;\theta')(R - V(s_i;\mathrm{w}'))$
    Accumulate gradients wrt w': $d\mathrm{w} \leftarrow d\mathrm{w} + \partial\left(R - V(s_i;\mathrm{w}')\right)^2 / \partial \mathrm{w}'$
  end for
  Perform asynchronous update of θ using $d\theta$ and of w using $d\mathrm{w}$.
until $T > T_{max}$

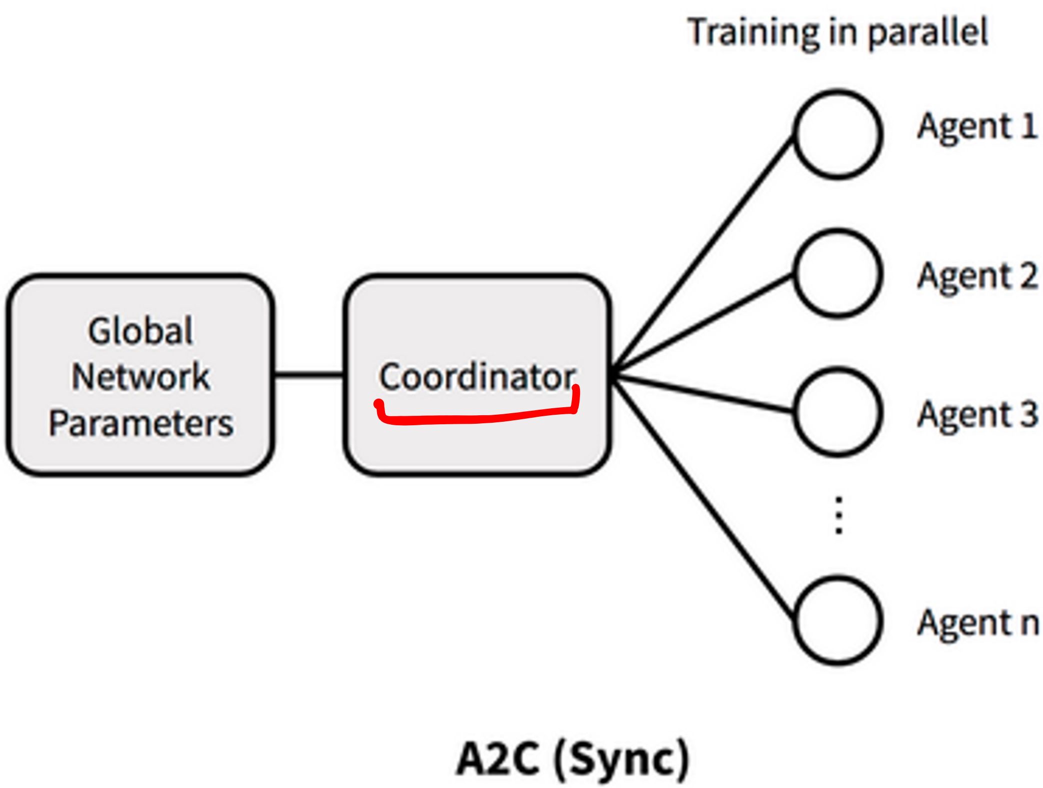
Reset thread params, update local params with global params

Gather experience (s_t, a_t, s_{t+1}, r_t)

Compute the gradients for this thread

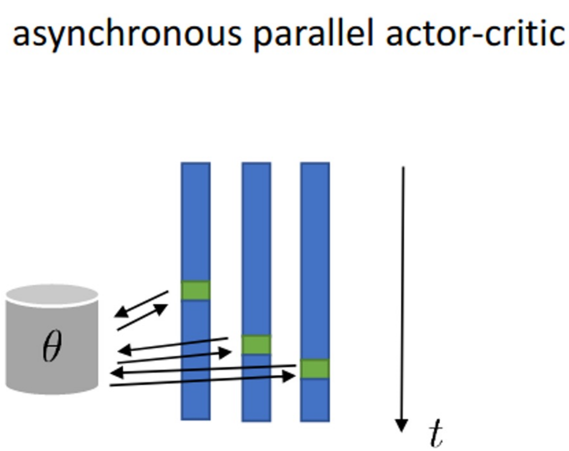
Update global params

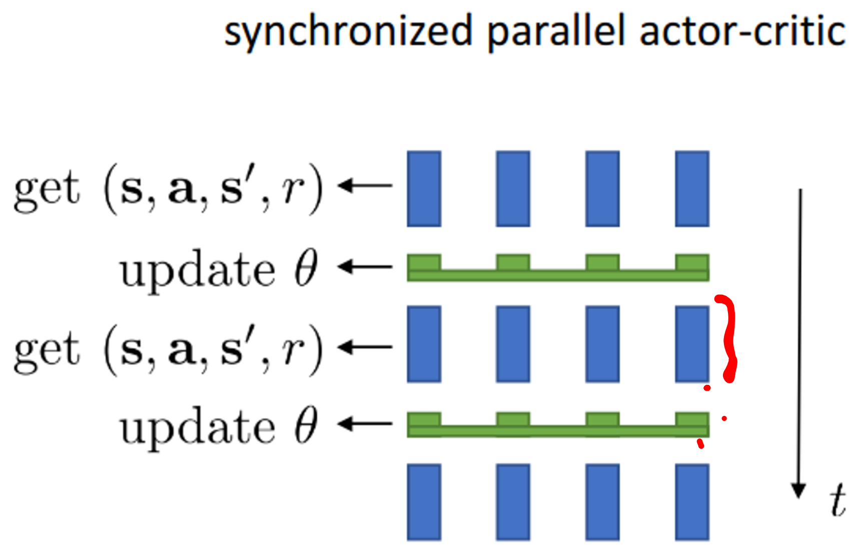
A2C – Synchronous Advantage Actor Critic



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.





PG Theorem
MC P.G.
A.C.
A3C
A2C

Compatible Parametrization

- Substituting the approximation $\hat{q}(\mathbf{s}, \mathbf{a}, \mathbf{w})$ instead of the true value of $q_\pi(\mathbf{s}, \mathbf{a})$ 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: $\hat{q}(\mathbf{s}, \mathbf{a}, \mathbf{w}) = \nabla_\theta \log \pi_\theta(\mathbf{a} \mid \mathbf{s})^T \mathbf{w}$
- Condition 2: $\mathbf{w}$ minimizes mean squared error:

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