Advantage Actor Critic

Policy gradient is combined with Temporal Difference where the state values are estimated by bootstrapping. TD-error formula is applied in Advantage function to find the excess return of choosing action a instead of following the [expected] policy:

$$\begin{aligned}\mathrm{Adv}_\pi(s,a) &= q_\pi(s,a) - v_\pi(s) \\ &= q_\pi(s,a) - \sum_a \pi(a \mid s) \cdot q_\pi(s,a).\end{aligned} \tag{1}$$

If the $\mathrm{Adv}_\pi(s,a) > 0$, taking action a is better than following the policy; if the $\mathrm{Adv}_\pi(s,a) < 0$, taking action a is worse than following the policy.

The Advantage of an action can be estimated from experience samples such that:

$$\begin{aligned}\hat{\mathrm{A}}\mathrm{dv}_\pi(s,a) &= \hat{q}_\pi(s,a) - \hat{v}_\pi(s) \\ &= r(s,a) + \gamma\hat{v}(s') - \hat{v}_\pi(s)\end{aligned} \tag{2}$$

where $r(s,a)$ is the reward of taking action a at state s, γ is the discount factor, $\hat{v}(s')$ is the estimate of the value of the reached state. Substracting state value $\hat{v}_\pi(s)$ acts as the *baseline* that helps eliminate the immediate effects of the state we are in and allows us to evaluate only the merits of the current state, whereas vanilla REINFORCE does not have baseline for its returns and thus the *nudges* for the parameters are fully weighted by the full returns.

Update Rule for A2C

The parameters for the neural networks are updated just like those in Policy Gradient methods, but replacing the Returns with Bootstrapped values minus Baseline:

$$\theta_{t+1} = \theta_t + \alpha\gamma^t \hat{\mathrm{A}}\mathrm{dv}_t \cdot \nabla \ln \pi(A_t \mid S_t, \theta_t) \tag{3}$$

where the Advantage is calculated using Temporal-Difference error at state t:

$$\hat{\mathrm{A}}\mathrm{dv}_t = R_{t+1} + \gamma\hat{v}(S_{t+1} \mid \omega) - \hat{v}_\pi(S_t \mid \omega). \tag{4}$$

where θ is the weights of the Actor (policy) network, and ω is the weights of the Critic (value) network.

> The expression looks like TD-error because it **is** TD-error, which includes bootstrapping the next state to speed up the learning process (and reduce variance while adding some bias), while the baseline is the old Q value of the current state to be updated.

> Just like REINFORCE, you can do the learning in parallel and also implement Entropy Regularization to encourage exploration.