Actor-Critic

Actor-Critic is a policy gradient method, with two competing models - an actor and a critic. The actor learns a policy function, while the critic learns a value function.

Actor-Critic Components

Actor

The goal of the actor is to learn a policy that produces the best action a, for a given state s, more specifically $\pi(a | s, \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ are the parameters of the function approximator.

Critic

The critic on the other hand, learns a value function. This could be a state-value function $\hat{v}(S, \mathbf{w})$ or an action-value function $\hat{Q}(S,A, \mathbf{w})$ [^1], where $\mathbf{w}$ are the parameters of the function approximator.

The critic criticises actions suggested by the actor, by using it's value function $\hat{v}(s, \mathbf{w})$ and calculating the TD (temporarl difference) error, as follows [^1]:

$\delta \leftarrow R+\gamma \hat{v}\left(S^{\prime}, \mathbf{w}\right)-\hat{v}(S, \mathbf{w})$

Which is then used to update the parameters of the actor and the critic [^1]:

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

Actor-critic is different to the reinforce with baseline algorithm, because it makes use of bootstrapping (the value function is an estimate of the true value) and the algorithm is online, i.e. the error is calculatied during the episode as opposed to after the end of the episode.

Architecture

These two models can be represented together and share parameters (example a double headed Neural Network) or they can be implemented as two seperate function approximators.

Deep Deterministic Policy Gradients (DDPG)

Deep Deterministic Policy Gradients (DDPG) is an actor-critic algorithm , which was introduced in this paper [^2]. It is model-free, online and off-policy and it learns a Q-value function and a policy. It uses the "Bellman equation to learn the Q-function, and uses the Q-function to learn the policy." [^3]

It can be seen as a combination of DPG (Deterministic Policy Gradient) and DQN (Deep Q-Networks) and it was made to work in continuous action spaces [^4].

DDPG componenets [^2] :

Q network

An action-value function $Q(S,A,\phi)$, which is also sometimes represented as $Q_{\phi}(s, a)$, where ${\phi}$ represents the parameters of the function approximator, in this case a neural network.

A deterministic policy network

The goal is to learn a deterministic policy $\mu_{\theta}(s)$ which gives the action that maximizes $Q_{\phi}(s,a)$. Since the action space is continuous, the policy is differentiable with the respect to action and can be solved using gradient ascent.[^3]

Target Q network & a target policy network

The term $r+\gamma(1-d) \max *{a^{\prime}} Q*{\phi}\left(s^{\prime}, a^{\prime}\right)$ is referred to as the target, because while minimizing the loss, we are making the Q-network more like this function. [^3] This is problematic because the target, depends on parameters ϕ, which we are trying to train and hence as we update ϕ, the target will change.

To solve this issue, a copy of the main network, known as the target network is used. The weights of the target network are frozen, to keep consistent while updating the weights. The target network is updated once per main network update by polyak averaging [^3] :

$\phi_{\mathrm{targ}} \leftarrow \rho \phi_{\mathrm{targ}}+(1-\rho) \phi$

Where ρ is a hyperparameter between 0 and 1 .

Steps of Algorithm [^5]

1. Use a replay buffer to sample experiences from the past to update Q-network (Experience replay).
2. Update actor and critic networks.
3. Update target networks.
4. Add noise to the continuous action output to encourage exploration.

Sources

[^1]: Page 332, Chapter 13: Policy Gradient Methods, Reinforcement learning: An introduction by Sutton, Richard S and Barto, Andrew G [^2]: Continuous control with deep reinforcement learning [^3]: Deep Deterministic Policy Gradient — Spinning Up documentation [^4]: Policy Gradient Algorithms [^5]: Deep Deterministic Policy Gradients Explained - Towards Data Science