Dennis Ward

10/07/2024

CS370

Module 6-2

Algorithms

The Cartpole problem involves balancing a pole on a moving cart by applying forces to the left or right. The REINFORCE algorithm, a Monte Carlo policy-gradient method, can effectively solve this problem by optimizing a policy that selects actions based on the current state of the environment. The core idea behind REINFORCE is to adjust the policy in a way that actions leading to high cumulative rewards are more likely to be taken in the future (Yoon, 2019).

In the Cartpole scenario, the state consists of variables such as the position and velocity of the cart and pole. The action could be either pushing the cart left or right. The REINFORCE algorithm collects these state-action pairs over entire episodes, calculates the total reward at the end, and updates the policy to increase the likelihood of actions that resulted in positive outcomes.

Pseudocode for the REINFORCE algorithm in Cartpole can be outlined as:

Initialize policy parameters θ

Repeat for each episode:

Initialize state s₀

Repeat for each time step:

Select action aₜ from policy π(a|sₜ, θ)

Execute action aₜ, observe reward rₜ and new state sₜ₊₁

Store (sₜ, aₜ, rₜ)

End episode

Compute return G for each time step

Update policy parameters θ ← θ + α ∑ G ∇θ log π(a|sₜ, θ)

The key concept is that the policy is updated at the end of each episode, with each action’s likelihood adjusted based on the cumulative reward for the episode (Yoon, 2019). One of the challenges of the REINFORCE algorithm is its high variance in updates, as it depends solely on rewards collected over entire episodes, making learning slower and more unstable in environments with delayed rewards.

The A2C (Advantage Actor-Critic) algorithm improves upon REINFORCE by combining both policy-based and value-based approaches. In A2C, there are two key components: the actor, which selects actions according to a policy, and the critic, which evaluates the quality of the actions taken using a value function (Juliani, 2017).

In the Cartpole problem, the actor chooses whether to move the cart left or right based on the current state. The critic evaluates this action by comparing the predicted value of the current state to the actual reward obtained. If the action performs better than expected, the policy is updated to reinforce that action. The advantage function used by the critic is crucial in determining how much better or worse the action was compared to the expected value.

A simplified pseudocode for A2C in Cartpole:

Initialize actor and critic parameters (θₐ, θᵥ)

Repeat for each episode:

Initialize state s₀

Repeat for each time step:

Select action aₜ from actor policy π(a|sₜ, θₐ)

Execute action aₜ, observe reward rₜ and new state sₜ₊₁

Compute advantage Aₜ = rₜ + γ V(sₜ₊₁, θᵥ) - V(sₜ, θᵥ)

Update actor θₐ ← θₐ + α Aₜ ∇θ log π(aₜ|sₜ, θₐ)

Update critic θᵥ ← θᵥ + β ∇θᵥ (Aₜ)²

In A2C, the policy (actor) is updated every time step, rather than at the end of the episode like in REINFORCE. Additionally, the critic helps reduce variance by providing a stable estimate of the advantage, allowing the actor to update more efficiently (AI Summer, 2023). A2C is more sample efficient and stable than pure policy-gradient methods because of this dual feedback system between the actor and critic.

Policy gradient methods like REINFORCE directly optimize the policy by adjusting the probability of actions based on rewards. They are well-suited for environments with continuous action spaces because they can handle a wide range of actions without discretizing them. In contrast, value-based approaches like Q-learning estimate the expected cumulative reward (Q-values) for each action in a given state and select actions by choosing the one with the highest estimated value (Sutton & Barto, 2018).

A significant difference is that policy gradient methods learn stochastic policies, meaning they directly model the probability distribution over actions. In contrast, Q-learning is deterministic—actions are chosen based on maximum Q-values. Policy gradient methods can inherently handle exploration by sampling from the policy distribution, while Q-learning requires explicit exploration mechanisms (e.g., epsilon-greedy) to avoid getting stuck in local optima (Yoon, 2019).

Another distinction is that policy gradient methods often have higher variance but are unbiased, as they rely on actual observed returns rather than bootstrapped estimates. Q-learning, on the other hand, typically has lower variance but can be biased due to its reliance on estimated values (Spinning Up, 2023).

Actor-critic approaches, such as A2C, combine the benefits of both value-based and policy-based methods. The actor, which selects actions, operates like a policy-gradient method, while the critic estimates the value of the chosen actions, similar to value-based methods (Yoon, 2019). This combination reduces the high variance found in policy-gradient methods by leveraging the critic’s value estimates to stabilize updates, making learning more efficient and stable.

Compared to pure policy-based methods (like REINFORCE), actor-critic methods are more sample-efficient, as the critic provides feedback on each action's quality during every time step, rather than waiting until the end of an episode to adjust the policy. Additionally, actor-critic methods can handle environments with continuous and discrete action spaces, making them more versatile than value-based methods like Q-learning, which perform better in discrete settings (AI Summer, 2023).

In summary, policy gradient methods optimize actions directly but suffer from high variance, while value-based methods estimate future rewards but struggle with continuous actions. Actor-critic approaches like A2C combine both strategies, leading to more efficient and stable learning.

**References:**

AI Summer. (2023). *The idea behind Actor-Critics and how A2C and A3C improve them*. AI Summer. <https://theaisummer.com/Actor_critics/>

Juliani, A. (2017). *Simple reinforcement learning with TensorFlow Part 8: Asynchronous actor-critic agents (A3C)*. Medium. <https://awjuliani.medium.com/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2>

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*.

Spinning Up. (2023). *Part 2: Kinds of RL Algorithms — Spinning Up documentation*. OpenAI. <https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html>

Yoon, C. (2019). *Deriving policy gradients and implementing REINFORCE*. Medium. <https://medium.com/@thechrisyoon/deriving-policy-gradients-and-implementing-reinforce-f887949bd63>

Yoon, C. (2019). *Understanding Actor Critic Methods and A2C*. Towards Data Science. <https://towardsdatascience.com/understanding-actor-critic-methods-931b97b6df3f>