Victor Udeh

CS370 Module 6

July 25th, 2024

6-2 Assignment: Cartpole Revisited

Solving the CartPole Problem Using REINFORCE and A2C Algorithms

**Introduction**

In reinforcement learning, various approaches can be used to solve problems such as the CartPole problem. This paper explores two different methods: the policy-based REINFORCE algorithm and the actor-critic-based A2C (Advantage Actor-Critic) algorithm. Additionally, it explains how these approaches differ from the value-based methods like Q-learning and highlights the unique aspects of actor-critic methods.

**Solving CartPole with REINFORCE Algorithm**

The REINFORCE algorithm is a policy gradient method that optimizes the policy directly. It uses the return (sum of future rewards) to adjust the policy parameters, moving them in a direction that increases the expected reward.

**Pseudocode for REINFORCE Algorithm**

Initialize policy network with random weights

Set learning rate alpha

For each episode:

Initialize state s

Repeat until termination:

Select action a according to policy π(a|s, θ)

Take action a, observe reward r and new state s'

Store (s, a, r)

s = s'

End Repeat

Compute return G for each time step t

For each time step t:

Update policy parameters θ using gradient ascent:

θ = θ + alpha \* G \* ∇θ log π(a|s, θ)

End For

End For

**Solving CartPole with A2C Algorithm**

The A2C algorithm combines both value and policy-based approaches. It uses two neural networks: one for the policy (actor) and another for the value function (critic). The actor updates the policy direction while the critic evaluates how good the action taken is, by calculating the advantage.

**Pseudocode for A2C Algorithm**

Initialize actor network (policy) with random weights

Initialize critic network (value function) with random weights

Set learning rates alpha\_actor and alpha\_critic

For each episode:

Initialize state s

Repeat until termination:

Select action a according to policy π(a|s, θ)

Take action a, observe reward r and new state s'

Compute advantage A = r + γ \* V(s', w) - V(s, w)

Update critic network weights w using gradient descent:

w = w - alpha\_critic \* ∇w A^2

Update actor network weights θ using gradient ascent:

θ = θ + alpha\_actor \* A \* ∇θ log π(a|s, θ)

s = s'

End Repeat

End For

**Policy Gradient vs. Value-Based Approaches**

Policy gradient methods, such as REINFORCE, optimize the policy directly by adjusting the parameters in the direction that increases the expected reward. These methods work well in continuous action spaces and can learn stochastic policies. In contrast, value-based methods like Q-learning estimate the value of state-action pairs and derive a policy by acting greedily with respect to the value function. Value-based methods struggle with continuous action spaces and require discretization.

**Actor-Critic Approaches**

Actor-critic methods combine the strengths of both policy and value-based approaches. The actor (policy) selects actions, and the critic (value function) evaluates them. The critic provides feedback to the actor, allowing it to improve the policy based on the estimated value of actions. This approach stabilizes training and often results in better performance and faster convergence compared to using pure policy or value-based methods alone.

**Conclusion**

The CartPole problem can be effectively solved using both the REINFORCE and A2C algorithms. While REINFORCE directly optimizes the policy, A2C leverages the advantages of both policy and value-based methods to provide a more stable and efficient learning process. Understanding the differences between these approaches and their implementation helps in selecting the appropriate method for various reinforcement learning tasks.

**References**

* Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction. MIT Press.
* OpenAI. (2018). Deriving Policy Gradients and Implementing REINFORCE. Retrieved from [Spinning Up](https://spinningup.openai.com/en/latest/spinningup/rl_intro3.html)
* OpenAI. (2018). Understanding Actor Critic Methods and A2C. Retrieved from [Spinning Up](https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html)