Module 6: CARTPOLE Revisited

CS-370

Brandon Hobbs

April 5, 2023

**Explain how the cartpole problem can be solved using the REINFORCE algorithm.**

The cartpole problem is a classic control problem in which a pole is attached to a cart via a hinge, and the goal is to balance the pole upright by moving the cart left or right. In previous modules I solved this using a DQN solution, however, it can be also solved with REINFORCE or other approaches.

The REINFORCE algorithm works by iteratively updating the parameters of a policy function to maximize the expected cumulative reward over multiple episodes.

Pseudocode might look like this:

*Call policy function with a random parameter*

*Get current state*

*Calculate the probability distribution of actions*

*Loop*

*Generate trajectories*

*Calculate the expected reward*

*For each time\_step*

*Reward\_current = Reward\_current + Reward\_previous*

*Calculate the gradient*

*Update the policy parameters*

*Exit upon policy convergence*

*End loop*

**Explain how the cartpole problem can be solved using the A2C algorithm.**

A2C or Advantage Actor-Critic is another policy-based reinforcement algorithm. The difference is that A2C learns not only the actor function but also the critic function – the critic evaluates how well the actor did or how valid its actions were. One key difference between other reinforcement learning is the Advantage function. The advantage function is the difference between the expected cumulative reward of taking a specific action in the current state and the expected cumulative reward of being in the current state.

Pseudocode might look something like this:

*Define the actor function*

*Define the critic function*

*Loop*

*Generate trajectories*

*Calculate the Advantage function*

*Calculate the policy gradients*

*Calculate the critic loss*

*Update the actor parameters*

*Update the critic parameters*

*Exit upon policy convergence*

*End loop*

**Explain how policy gradient approaches differ from value-based approaches, such as Q-learning.**

Q-learning, a value-based approach, tries to tune the action-value function by computing the expected cumulative reward for a state/action pair. The agent chooses an action by selecting the action with the highest expected cumulative reward in its current state. This selection is based on prior learned action-value pairs and is calculated by the action-value function. The algorithm learns the optimal action-value function by iteratively updating the estimates of the action-value function using the Bellman equation and using an exploration strategy to visit different states and actions. Q-learning and other value-based approaches are model-free, meaning that they do not model the environment's dynamics and do not learn a policy explicitly.

The policy gradient approach aims to learn the optimal policy directly, without explicitly learning the action-value function. The policy function maps a state to a probability distribution over actions, and the agent selects actions by sampling from this distribution. The algorithm learns the optimal policy by iteratively updating the policy parameters using the gradient of the expected cumulative reward with respect to the policy parameters. Policy gradient approaches can be model-free or model-based, depending on whether they model the environment's dynamics.

One key difference between policy gradient and value-based approaches is that policy gradient approaches can handle continuous actions, while value-based approaches are typically limited to discrete actions.

A3C (a policy-based algorithm) “…could work in continuous as well as discrete action spaces” (Juliani, 2016).

**Explain how actor-critic approaches differ from value- and policy-based approaches.**

As mentioned previously, value-based approaches have the agent learn to estimate the expected cumulative reward for each state-action pair; policy-based approaches have the agent learn to directly optimize the policy, which maps states to actions. Actor-critic approaches combine elements of both value-based and policy-based and learn both a policy function and a value function simultaneously.

A2C or Advantage Actor-Critic is another policy-based reinforcement algorithm. The difference is that A2C learns not only the actor function but also the critic function – the critic evaluates how well the actor did or how valid its actions were.

**Citations**:

Juliani, A. (2016, December 16). *Simple reinforcement learning with Tensorflow part 8: Asynchronous Actor-Critic Agents (A3C)*. Medium. https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2

Karagiannakos, S. (2018, November 17). *The idea behind Actor-Critics and how A2C and A3C improve them*. AI Summer. https://theaisummer.com/Actor\_critics

OpenAI. (n.d.). *Part 2: kinds of RL algorithms*. https://spinningup.openai.com/en/latest/spinningup/rl\_intro2.html#a-taxonomy-of-rl-algorithms

Yoon, C. (2018, December 29). *Deriving policy gradients and implementing REINFORCE*. Medium. https://medium.com/@thechrisyoon/deriving-policy-gradients-and-implementing-reinforce-f887949bd63