CS 370: Module Six Assignment

Jason Kremhelmer

Southern New Hampshire University

CS 370: Current/Emerging Trends in CS

Roland Morales

10/13/2024

# Explain how the carpole problem can be solved using REINFORCE algorithm.

As the cartpole problem is a classic control problem where the pole is attached to the cart via hinge, with the goal being balancing the pole upright by movement of the cart right or left. The use of DQN was used in previous attempts at a solution, however this is not the only viable solution. The REINFORCE approach along with others may be used to solve this problem.

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

Pseudocode example:

*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 A2C algorithm.

Advantage Actor-Critic, or A2C, is a policy-based reinforcement algorithm. The difference with A2C is that it not only learns from the actor function, but also the critic function as well. The critic evaluates the actor and how well the actor did, and how valid the action was. A key point of the A2C 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 example:

*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, being a value-based approach, attempts to tune the action-value function. It does this by computing the expected cumulative reward for a state/action pair. An action is chosen by the agent that has the highest expected cumulative reward in the current state. The selection is based on prior learned action-value pairs and is calculated by the action-value function. The algorithm gains the optimal action-value function by iteratively updating the estimates of the action-value function using the Bellman equation, as well as using an exploration strategy to visit different states and actions. With Q-learning and other value-based approaches being model-free, that is they do not model the environment’s dynamics and do not learn a policy explicitly.

Policy gradient approach aims to learn the optimal policy directly. This is done without explicitly learning the action-value function. The function maps a state based on probability distribution over actions, while the agent selects actions by sampling from this distribution. This allows the algorithm to learn the optimal policy by iteratively updating the policy parameters using the gradient of the expected cumulative reward with respect to the policy parameters. Depending on whether the environment’s dynamics are modeled or not, the policy gradient can either be model-free or model-based.

A key difference between value-based and policy gradient approaches is the policy gradient can handle continuous actions, while value-based are limited to discrete action for the most part.

Asynchronous Actor-Critic Agents (A3C) “…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 stated above, value-based approaches have the agent learn to estimate the expected cumulative reward for each state-action pair. Where in policy-based approaches, the agent learns to directly optimize the policy, which in turn maps state to actions. With the actor-critic approach, it combines elements of both value- and policy-based approaches by learning both the policy and the value functions simultaneously.

Advantage Actor-Critic (A2C) is another type of policy-based reinforcement algorithm. A difference is that A2C learns from both the actor and critic functions. The critic evaluates if the actions were valid and how well the actor did.

# References

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

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

*Part 2: Kinds of RL Algorithms*. (n.d.). Retrieved from OpenAI Spinning Up: 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*. Retrieved from Medium: https://medium.com/@thechrisyoon/deriving-policy-gradients-and-implementing-reinforce-f887949bd63