Bryce Jensen
CS-370-11333-M01
6-2 Assignment: Cartpole Revisited
6/13/2023

Explain how the cartpole problem can be solved using the REINFORCE algorithm. Consider using pseudocode, UML, diagrams, or flowcharts to help illustrate your solution.

1. The agent looks at the current state of the cart and pole (cart position and velocity, and pole angle and angular velocity), then decides what action to take ("Should I move the cart left or right?")
2. The environment (the cart and pole) responds to the agent's action by changing its state (the pole might wobble a bit).
3. If the pole is still balanced, the agent gets a reward! If it falls, no reward.
4. The algorithm uses this information to update the agent's policy (i.e., what actions to take in similar situations). It says, "Hey, when I made that choice last time, it worked out well! Let me make a similar choice next time too!"

The agent repeats steps 1-4 hundreds or even thousands of times, trying different actions and learning from the results. Over time, it gets better at balancing the pole.

I will explain how the cartpole problem can be solved using the REINFORCE algorithm using some pseudocode. The main idea is to collect a trajectory (episode) using the current policy, calculate the total expected reward for that episode, and then update the agent's policy parameters using the policy loss. This is based on this implementation of REINFORCE on Cartpole. Hopefully this explains it well enough:

```
# Main loop
for each episode within n_episodes:
    # Initialize variables for this episode, saved log and reward arrays
    # Reset the environment

    # Collect a trajectory (episode) using the current policy loop
    for each trajectory within max_trajectory:
        # Sample an action from the current policy
        # save the current log to the saved logs
        # Take the action and get the next state, reward, and done flag
        # Store the reward for this step
        # If done, break out of the loop

    # Calculate the total expected reward for this episode
    # Save it to the scores array

    # this part is COMPLICATED because of how he implemented it, I hope
    I get this right.
    # Recalc the reward with the discount factor
    Discount list = discount factor to the power of its index
    REWARD = the sum of all the discounts * rewards
```

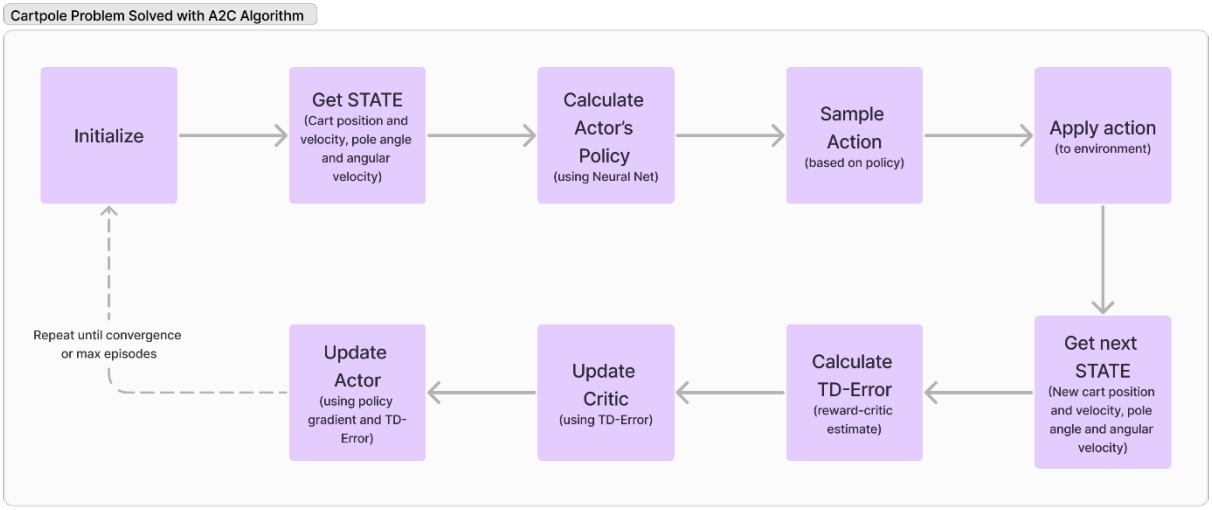
```
        # Calculate the loss
        for each log in saved log:
            save to a list, the -log * REWARD

        # I believe we add each of the values from the loss list above to
        create a single value that is used for updating policy parameters

        # Print some progress information
```

(Yoon, 2018; Kang, 2021)

Explain how the cartpole problem can be solved using the A2C algorithm. Consider using pseudocode, UML, diagrams, or flowcharts to help illustrate your solution.



Created using Figma

Explanation of the above flowchart:

Initialize: Initialize the A2C agent with random weights for the actor and critic models.

Get State: Get the initial state from the environment (cart position and velocity, pole angle and angular velocity).

Calculate Actor's Policy: Calculate the actor's policy for the current state using a neural network.

Sample Action: Sample an action based on the calculated policy.

Apply Action: Apply the sampled action to the environment

Get next state: Get the next state from the environment (cart position and velocity, pole angle and angular velocity).

Bryce Jensen
CS-370-11333-M01
6-2 Assignment: Cartpole Revisited
6/13/2023

Calculate TD-Error: Calculate the TD-error (reward - critic estimate) using the reward from the environment and the critic's value function estimate.

Update Critic: Update the critic's value function estimate using the calculated TD-error.

Update Actor: Update the actor's policy parameters using the policy gradient and the calculated TD-error.

Repeat: Repeat the process until convergence or a maximum number of episodes.

(Yoon, 2019; Wang, 2021)

Man, that was way easier than the pseudocode. Almost made me want to redo the first section to make it clearer. See if I ever deal with pseudocode ever again. ;)

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

Based on my understanding there are three differences between value-based (VB) and policy gradient (PG) approaches that stand out to me. I think a table makes the most sense here.

Major differences ___	Value-Based (VB)	Policy Gradient (PG)
in Objective	To learn a value function that estimates the expected value of taking a specific action in a scenario. It does this to get the largest reward, which will eventually lead to the goal (OpenAI, 2018).	To optimize the policy for the maximum reward. It does this to learn the best sequence of actions to reach the goal (OpenAI, 2018).
in Algorithm	There are different VB algorithms that can be used. I'll use the one we are using in this class. In the least technical way of explaining it, the Q-learning algorithm updates a value for each action in a situation. VB algorithms focus on learning how good or bad each action is (OpenAI, 2018).	There are also different PG algorithms, the one we used this week is called REINFORCE. This one learns the best actions to take by following the rewards. PG algorithms focus on learning what actions to take in different scenarios (OpenAI, 2018).
in Learning Strategy	VB methods learn the best reward for each action. They update values using rules (like Q-learning). Repeat (OpenAI, 2018). They are less stable, but more efficient in learning (OpenAI, 2018).	PG method is to learn the best actions to take in different situations. They calculate how to improve the policy based on rewards. Then update those rules or policies and repeat. They are more stable and reliable because you are training directly toward the goal (OpenAI, 2018).

Bryce Jensen
CS-370-11333-M01
6-2 Assignment: Cartpole Revisited
6/13/2023

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

As mentioned in the previous sections, there are some benefits and drawbacks to both value-based (VB) and policy Gradient (PG) approaches. Actor-critic approaches are designed to mitigate these by combining the benefits of both VB and PG approaches (Karagiannakos, 2018). Value-based approaches (Q-learning) focus on learning the best value function that estimates the expected return. Policy-based approaches (policy gradients) focus on learning the best policy (or set of rules) that select actions based on states to get the best reward (OpenAI, 2018).

An actor-critic approach combines each of these by splitting the model into two parts and training one part (actor) with a value-based method and the second part (critic) with a value-based method. It then uses both together, the actor uses the critic's value function to update its policy parameters and the critic uses the actor's selected actions to update its value function estimates (Karagiannakos, 2018; Juliani, 2016). This allows it to learn more effectively than either a VB or PG method can do on its own.

References

Juliani, A. (2024, May 14). 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

Kang, C. (2021, May 12). REINFORCE on CartPole-V0. *Chan`s Jupyter*. https://goodboychan.github.io/python/reinforcement_learning/pytorch/udacity/2021/05/12/REINFORCE-CartPole.html

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

Part 2: Kinds of RL Algorithms — Spinning Up documentation. (n.d.). https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html#a-taxonomy-of-rl-algorithms

Wang, M. (2023, June 3). Advantage Actor critic Tutorial: MinA2C - towards Data Science. *Medium*. https://towardsdatascience.com/advantage-actor-critic-tutorial-mina2c-7a3249962fc8

Yoon, C. (2021a, December 7). Deriving policy gradients and implementing REINFORCE. *Medium*. https://medium.com/@thechrisyoon/deriving-policy-gradients-and-implementing-reinforce-f887949bd63

Yoon, C. (2021b, December 7). Understanding actor critic methods and A2C - towards data science. *Medium*. https://towardsdatascience.com/understanding-actor-critic-methods-931b97b6df3f