One approach to solving the cartpole problem is by utilizing the REINFORCE algorithm. This algorithm is a Monte-Carlo, or the taking of random sampling, variant for policy gradients (Yoon, 2018). This means that the weights are initially randomized to maximize the expected reward. The current policy is utilized to collect a trajectory for each episode, which is then used to update policy parameters. In other words, with each small step, the policy's weights are updated, allowing for another, yet more rewarding, step. Furthermore, the discounted rewards are normalized to provide stability in training by controlling the variance of the policy (Joshi, 2022). The goal of this algorithm is to find the optimal policy directly without utilizing the Q-value (Karagiannakos, 2018). The algorithm can be defined by the following flowchart:

1. Use the policy to determine a trajectory.
2. Estimate the reward for the steps within the trajectory.
3. Store policy probabilities and rewards for each step.
4. Use the trajectory to estimate the policy gradient.
5. Update the policy weights.
6. Repeat until convergence.

In applying the REINFORCE algorithm to the cartpole problem, the artificial neural network (ANN) would first initialize a policy with random sampling. This involves the ANN utilizing the state to return to probability for different actions. Next, the agent would play a certain number of steps in the game and record the probabilities and rewards for each step. These steps include moving the cart left or right. Each step would have discounted rewards calculated and normalized through backpropagation to determine the expected rewards. Then, the weights on the policy would be updated to continually increase the expected rewards, slowly enhancing

the performance of the algorithm. This process would then continue until the policy had been tuned adequately for the agent to make precise enough decisions to keep the pole vertical under the cart.

An alternative approach to solving the cartpole problem is to utilize the Advantage Actor Critic (A2C) algorithm. The A2C is broken down into two function approximations: an actor, which will carry out actions and uses a policy function, and a critic, which will influence future actor decisions and uses a value function (Simonini, 2022). In doing so, the actor will decide which action is best to take in the current state based on the policy. This information will be passed to the critic, who will compute the value of the combination of action and state to determine the advantage value. This value is computed by subtracting the mean state value from the Q-value to determine how much more beneficial a certain action is given the current state (Karagiannakos, 2018). This value will then be utilized to update the policy parameters of the actor, while actor decisions will update the value parameters of the critic based on the mean squared error (MSE) loss function. The weights are updated for both actors and critics at each step to continually make and influence more accurately. A2C is designed to incorporate the benefits of policy and weight-based methods to reduce the variance, improve stability, and increase algorithm efficiency. An A2C algorithm can be defined by the following flowchart:

1. Reset gradients.
2. Obtain a new state and observe.
3. The actor samples and acts based on the current policy to receive rewards.
4. Critic calculates the advantage value used to update gradients.
5. Update policy parameters for actors with advantage values.

6. Update value parameters for critic by MSE.
7. Repeat until convergence.

In applying the A2C algorithm to the cartpole problem, the actor would first receive the current state of the game to include the positions of the cart and pole. The actor would then take the best action based on the policy. These actions include moving the cart left or right. The action is then passed through the critic, who computes the advantage value to update the actor's policy parameters. Should an action have a penalty, the critic will negatively push the gradient, influencing future moves against similar actions. In doing so, the actor will be able to make increasingly precise cart movement decisions to balance the pole. The critic will also calculate the MSE to have more accurate influences on the actor. This process would then repeat until the actor's policy parameters have been adjusted enough while considering the influences of the critic to keep the pole vertical under the cart.

Policy gradient approaches differ from value-based approaches in how each algorithm determines optimal actions. Policy gradient algorithms utilize probability distributions for actions given a state to assign actions with higher expected rewards with higher probabilities. After each iteration, the algorithm computes a loss function using the expected return and action probability to train the network, allowing for increasingly precise decisions (Doshi, 2021). Alternatively, a value-based approach obtains the optimal policy indirectly by learning the value function, or Q-value in Q-learning, for each state and action. In doing so, the algorithm requires training through exploration, which can be achieved through an epsilon-greedy policy to build a matrix of action-reward. Based on this matrix, the algorithm will be able to exploit previously gained knowledge to pick the most optimal action. Furthermore, the Bellman equation can be

utilized to estimate future rewards. However, policy gradient approaches are typically better suited for environments with a multitude of variables and actions or when a single best action is available (Doshi, 2021). This is due to the probability distribution nature of a gradient approach, which ensures each action shares a separate importance.

Actor-critic approaches differ from value- and policy-based approaches, as actor-critic approaches utilize both value- and policy-based approaches in tandem. The actor forms the policy function, while the critic performs the value function. During each iteration, the actor's policy parameters are adjusted with policy gradients and advantage values, while the critic's value parameters are modified with the MSE (Yoon, 2019). Throughout training, the actor learns how to make more precise decisions based on the current policy, while the critic serves to influence the actor more accurately by evaluating each action. Alternatively, value- and policy-based approaches utilize matrix tables and probability distributions, respectively. Each individual approach has drawbacks depending on the environment they are situated in, such as when used for continuous or stochastic problems. However, by combining these functions in an actor-critic approach, the variance of a policy gradient can be significantly reduced while stabilizing the model.

References

Doshi, K. (2021, January 9). Reinforcement Learning Explained Visually (Part 6): Policy Gradients, step-by-step. Medium. https://towardsdatascience.com/reinforcement-learning-explained-visually-part-6-policy-gradients-step-by-step-f9f448e73754

Joshi, N. (2022, February 26). Part 2 : Policy Based Reinforcement Learning — OpenAI's Cartpole with REINFORCE algorithm. Medium. https://nandakishorej8.medium.com/part-2-policy-based-reinforcement-learning-openais-cartpole-with-reinforce-algorithm-18de8cb5efa4

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

Simonini, T. (2022, June 22). Advantage Actor Critic (A2C). Hugging Face. https://huggingface.co/blog/deep-rl-a2c

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

Yoon, C. (2019, February 5). Understanding Actor Critic Methods and A2C. Medium. https://towardsdatascience.com/understanding-actor-critic-methods-931b97b6df3f