CSPB 3202: Final Project

Ashley Kastler

GitHub Final Project Demo.mp4

Overview

In my final project, I wanted to explore the effectiveness of the Q-Learning algorithm in solving the discrete Mountain Car problem, a classic control reinforcement learning challenge provided by the OpenAI gym. The mountain car environment represents a scenario where a car that has limited power is in between two hills and needs to rock back and forth to gain momentum to drive up the right hill and towards the flag which is the goal state. The goal of this project was to develop, train and test a Q-Learning model to successfully drive the car to the flag at the top of the hill.

Approach

Environment

The Mountain Car environment contains a state space defined by position and velocity and an action space that consists of three discrete actions such as accelerate to the left, accelerate to the right or do nothing. The car's position ranges between -1.2 and 0.6, and its velocity ranges between -0.07 and 0.07. The car starts at the bottom of the valley and the goal is to drive the car to a position that is equal or greater than 0.5.

Model Selection

I chose the Q-learning algorithm for this project due to its simplicity and effectiveness in discrete action spaces. Q-learning is an off-policy RL algorithm that learns the value of actions in states by iteratively updating a Q-table based on the agent's experiences. I found that this algorithm was a good approach to this problem as it can effectively learn the optimal policy by exploring different states and actions that lead to higher rewards and eventually the goal.

Methods

The parameters that made major impacts to my algorithm were:

- Learning rate: Controlled how much new information overrides the old information. After testing the algorithm with different values, I found 0.1 to be the most effective.
- Discount factor: Determined the importance of future rewards. I set the default to 0.99 to favor long -term rewards. I also found during testing that this combination of values enabled the agent to be successful before 100 episodes.
- Epsilon: Determined the probability of taking a random action. I used an epsilon greedy policy starting with epsilon = 1 so the agent would explore and take completely random actions. Over time I had an epsilon decay value that adjusted based on the number of episodes the agent was being trained for that comes to a minimum of 0.
- Episodes: The more episodes the agent was given to train for allowed for more time to explore and find a successful policy.

CSPB 3202: Final Project

Ashley Kastler

GitHub Final Project Demo.mp4

The training process involved updating the Q-table after each action, based on the reward received and the maximum expected future reward.

Troubleshooting

Throughout the project, I encountered several challenges, such as balancing exploration and exploitation, working with the continuous state space, and managing the Q-tables convergence. To solve these issues, I played around with the epsilon and epsilon decay values as well as introducing more randomness by using a random number generator to determine whether to take a random action or not. I discretized the state space by creating 20 bins that span the range of values so I could use an even state space to store values for in the q table.

Results

The trained Q-learning model successfully learned to drive the car to the goal at least once before 100 episodes in an average of 750 steps. However, as the algorithm completes more episodes the number of steps it takes to get to the goal decreases and successes are more consistent. After only 1000 episodes the Q-table starts to converge, and the policy becomes stable with the car consistently reaching the goal state. The results demonstrated that the Q-learning algorithm could effectively solve the Mountain Car problem after 1000 episodes.

Iterative Improvements

During the training process, I iteratively adjusted the hyperparameters to improve the model's performance. For example, reducing the learning rate slightly after observing instability in early episodes helped in stabilizing the Q-value updates. Additionally, experimenting with different epsilon rate decay schedules led to faster convergence.

Conclusion

The Q-learning algorithm proved to be an effective solution for the Mountain Car problem, successfully learning an optimal policy through trial and error. The primary challenge was managing the exploration-exploitation trade-off, which was addressed by fine-tuning the epsilon-greedy policy. This project highlighted the importance of hyperparameter tuning and exploration strategies in reinforcement learning. While Q-learning worked well for this action discrete environment, it may face challenges in more complex or continuous action spaces. Future work could involve experimenting with other RL algorithms, such as Deep Q-Networks (DQNs) or policy gradient methods, to address these challenges.

References

"The BEST Q-Learning example! | The Mountain Car problem" *YouTube,* uploaded by Marcus Koseck, 6 Jan 2024, https://www.youtube.com/watch?v=u6DJvaOkRS8

CSPB 3202: Final Project

Ashley Kastler

GitHub Final Project Demo.mp4