Using Q-learning to Navigate in a Grid-World Environment

Introduction

Q-learning is a model-free reinforcement learning algorithm used to find the optimal action-selection policy for a given environment. In the context of navigation in a grid-world environment, Q-learning aims to train an agent to navigate from the starting position to the goal position while maximizing cumulative rewards obtained along the way.

Q-learning

Q-learning is a form of temporal difference learning, where the agent learns from its actions and the resulting rewards. At each step, the agent selects an action based on the current state and updates its Q-values to reflect the expected future rewards for taking that action in that state.

Q-table

The Q-table is a data structure used by the agent to store Q-values for each state-action pair. It represents the learned knowledge of the agent about the environment. The Q-values indicate the expected cumulative rewards for taking a specific action from a given state.

How Q-learning Works

Q-learning works by iteratively updating Q-values using the following formula:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \cdot \left[r + \gamma \cdot \max_{a'} Q(s',a') - Q(s,a) \right]$$

Where:

- $Q(s,a)$ is the Q-value for taking action a in state s.
- α is the learning rate, controlling the extent to which new information overrides old information.
- r is the immediate reward received after taking action a in state s.
- γ is the discount factor, representing the importance of future rewards.
- s' is the resulting state after taking action a in state s.

The agent updates the Q-value for each state-action pair based on observed rewards and transitions, gradually learning the optimal action-selection policy.

Q-learning Approach

Q-learning can be used to solve this problem efficiently by following these steps:

1. **Initialize Q-table**: Create a Q-table to store action values for each state-action pair. Initialize the Q-values arbitrarily or to zeros.
2. **Exploration and Exploitation**: Balance exploration and exploitation by selecting actions using an ϵ-greedy policy. Initially, explore more to discover the environment, and gradually shift towards exploitation to exploit learned knowledge.
3. **Update Q-values**: Use the Q-learning update rule to iteratively update Q-values based on observed rewards and transitions. The update rule adjusts the Q-value towards the observed reward, incorporating learning rate and discount factor.
4. **Convergence**: Repeat the exploration-exploitation and Q-value update steps until convergence, i.e., until the Q-values stabilize or converge to optimal values.

Result

By applying Q-learning, the agent can effectively learn to navigate from the starting position to the goal position while maximizing cumulative rewards. The learned Q-values represent optimal action-selection policies, enabling the agent to choose actions that lead to higher rewards and avoid obstacles. Experimental results demonstrate the effectiveness of the Q-learning approach in solving this navigation problem in grid-world environments.