DQN for CartPole - Project Documentation

Project Overview

The goal of this project is to train an AI agent using Deep Q-Learning (DQN) to solve the CartPole environment in OpenAI's Gym.

The project involves building a neural network that will learn to balance a pole on a cart by interacting with the environment, receiving rewards,

and improving its performance over time through training.

Key Concepts:

- Reinforcement Learning (RL): An area of machine learning where an agent learns to make decisions by interacting with the environment and receiving rewards based on its actions.
- Q-Learning: A reinforcement learning algorithm where the agent learns a value (Q-value) for each action it takes in each state.
- Deep Q-Learning (DQN): An extension of Q-Learning that uses a deep neural network to approximate the Q-values.

Steps Taken in the Project

1. Set Up Environment:
   - OpenAI Gym is used to create the CartPole environment.
   - Python libraries: gym, torch, numpy.

2. Define the DQN Agent:
   - Neural Network: 2 fully connected layers to approximate Q-values.
   - Replay Memory: Stores experiences and samples randomly to update Q-values.
   - Action Selection: Uses epsilon-greedy strategy to choose actions.

3. Training the Agent:

- The agent interacts with the environment and learns using the Bellman Equation to update Q-values.

- The agent decays epsilon over time to shift from exploration to exploitation.

4. Testing the Trained Model:

- After training, the model is loaded and the agent is tested by running it through the environment without exploration (epsilon = 0).

Code Breakdown

Neural Network Architecture:

- Input Layer: 4 neurons (Cart state).

- Hidden Layer: 64 neurons (can vary).

- Output Layer: 2 neurons (possible actions: left or right).

Training Loop:

1. Initialize state, select action using epsilon-greedy strategy.

2. Interact with the environment, store the experience in memory.

3. Sample random experiences, compute target Q-values, and update the model.

4. Decay epsilon to focus more on exploitation.

Testing Loop:

- Run the trained agent in the environment with epsilon set to 0 for no exploration.

Project Flowchart

1. Initialize Environment -> 2. Create DQN Agent -> 3. Train Agent (loop through episodes) ->

4. Test Agent (after training) -> 5. Save Trained Model.

- Train Agent:
    - Select Action (epsilon-greedy) -> Take Action in Environment ->
    - Store Experience (state, action, reward, next state) ->
    - Sample from Replay Memory -> Update Neural Network.

Neural Network Overview

- Input: State (4 values: cart position, cart velocity, pole angle, pole velocity).
- Output: Q-values for each action (2 actions: move left, move right).
- The network learns to approximate the Q-values and improve as the agent interacts with the environment.

Conclusion

- Goal: The agent learns to balance the CartPole by maximizing cumulative rewards.
- Achievements: The agent improves over time and successfully balances the pole.
- Next Steps: Explore more complex environments, use other reinforcement learning algorithms.