Understanding Deep Q-Networks (DQN)

Aayush Borkar, Aditya Neeraje, Balaji Karedla, Nirav Bhattad

IIT Bombay

Finsearch Project Midterm Presentation

July 15, 2024

Introduction to Reinforcement Learning

- Reinforcement Learning (RL) is a type of machine learning where an agent learns to make decisions by taking actions in an environment to maximize cumulative reward.

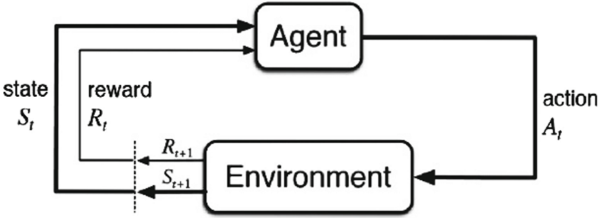

Figure: How an RL Algorithm works

Key Concepts of RL

- **Policy:** The strategy an agent employs to determine the next action based on the current state.

Key Concepts of RL

- **Policy:** The strategy an agent employs to determine the next action based on the current state.
- **Value Function:** Measures the expected reward of a state.

Key Concepts of RL

- **Policy:** The strategy an agent employs to determine the next action based on the current state.
- **Value Function:** Measures the expected reward of a state.
- **Q-Function:** Measures the expected reward of taking a certain action in a given state.

What is Q-Learning?

- A model-free RL algorithm that seeks to find the best action to take given the current state.
- **Bellman Equations:** Explain the relationship between the value of a state and the values of its successor states.

$$v(s) = \max_a \left[\sum s' P(s' \mid s, a) \left(R(s, a, s') + \gamma V(s') \right) \right]$$

$$Q(s, a) = \sum_{s'} P(s' \mid s, a) \left(R(s, a, s') + \gamma \max_{a'} Q(s', a') \right)$$

Introduction to Deep Q-Networks (DQN)

- **Motivation:** Combining Q-Learning with deep learning to handle high-dimensional state spaces.
- **Architecture:** Uses neural networks to approximate the Q-Function.

DQN Algorithm Overview

- **Experience Replay:** Storing agent's experiences and randomly sampling them to break correlation between consecutive samples.
- **Target Network:** Using a separate target network to stabilize training.

DQN Workflow

1. Initialize replay memory.
2. Initialize the Q-network with random weights.
3. For each episode:
    - Select action using epsilon-greedy policy.
    - Execute action and observe reward and next state.
    - Store experience in replay memory.
    - Sample random batch from replay memory.
    - Perform gradient descent step on loss between Q-values and target Q-values.

Epsilon-Greedy Policy

- **Exploration vs. Exploitation:** Balancing exploration of new actions and exploitation of known rewarding actions.
- **Epsilon Decay:** Gradually reducing the exploration rate over time.

Experience Replay

- **Purpose:** Improves sample efficiency and breaks temporal correlations.
- **Method:** Stores past experiences and samples them randomly during training.

Target Network

- **Purpose:** Stabilizes training by reducing oscillations in Q-value updates.
- **Method:** Periodically copies weights from the main Q-network to the target network.

Target Process

- **Loss Function:** Mean Squared Error (MSE) between predicted Q-values and target Q-values.
- **Optimization:** Use of gradient descent or variants like Adam.

Applications of DQN

- **Gaming:** Achieving superhuman performance in Atari games.
- **Robotics:** Learning control policies for complex tasks.
- **Finance:** Optimize trading strategies.

Challenges and Limitations

- **Stability and Convergence:** Difficulties in training stability and convergence.
- **High Computational Cost:** Requires substantial computational resources.
- **Sample Efficiency:** Large amount of data required for effective training.

Advances and Variants

- **Double DQN:** Addresses overestimation bias in Q-learning.
- **Dueling DQN:** Separates value and advantage functions for better learning.
- **Prioritized Experience Replay:** Improves learning by sampling important experiences more frequently.

Conclusion

DQN combines Q-Learning with deep neural networks to handle complex, high-dimensional state spaces effectively.