EC332 Machine Learning

Reinforcement Learning: Deep Q-Learning

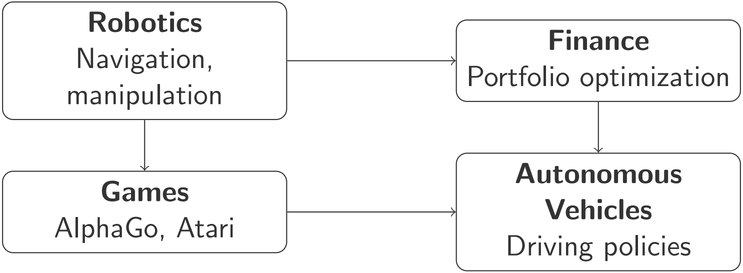
Nazar Khan
Department of Computer Science
University of the Punjab

What is Reinforcement Learning?

- A framework for decision-making problems.
- Agent interacts with an environment to maximize cumulative rewards.
- Key elements:
  - States (s)
  - Actions (a)
  - Rewards (r)
  - Policy (π)

Applications of Reinforcement Learning

- Robotics: Robot navigation, manipulation tasks.
- Games: AlphaGo, Atari games.
- Finance: Portfolio optimization.
- Autonomous vehicles: Driving policies.



Q-Learning Overview

- Model-free reinforcement learning algorithm.
- Learns the optimal action-value function $Q(s, a)$:

$$Q(s, a) = \mathbb{E}[r + \gamma \max_{a'} Q(s', a')]$$

- Update rule:

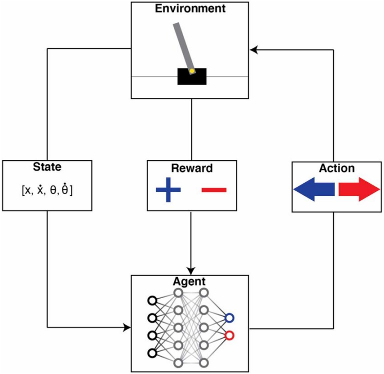
$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$$

Challenges in Q-Learning

- Inefficient for high-dimensional state spaces.
- Requires a large Q-table for discrete states and actions.
- Cannot directly handle continuous state spaces.

What is Deep Q-Learning?

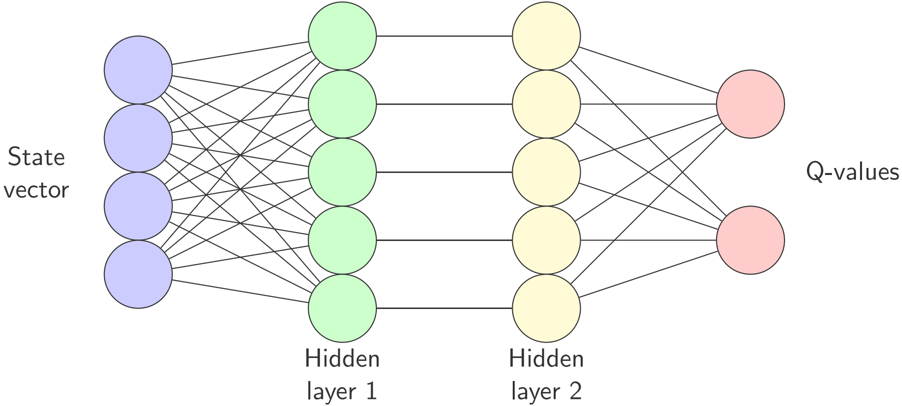
- Combines Q-Learning with deep neural networks.
- Approximates $Q(s, a)$ using a neural network.
- Outputs Q-values for all actions given a state.
- Overcomes limitations of tabular Q-Learning.



From Maffettone et al. (2021)

DQL Network Architecture

- Input: State vector.
- Hidden layers: Fully connected layers with ReLU activation.
- Output: Q-values for all possible actions.



Replay Buffer

- Stores past experiences: $(s, a, r, s', \text{done})$.
- Breaks temporal correlations in training data.
- Enables efficient reuse of experiences.
- Samples mini-batches for training.

Target Network

- Separate network used to calculate target Q-values.
- Stabilizes training by reducing correlations in updates.
- Periodically synchronized with the main Q-network.

DQL Training Steps

1. Initialize Q-network and target network.
2. Initialize replay buffer.
3. For each episode:
    1. Reset the environment.
    2. Select actions using ϵ-greedy policy.
    3. Store transitions in replay buffer.
    4. Train Q-network using mini-batches from buffer.
    5. Periodically update target network.

Loss Function

- Mean Squared Error (MSE):

$$\text{Loss} = \frac{1}{N}\sum_{i=1}^{N}\left(Q(s,a) - \text{target\_q}\right)^2$$

- Target Q-value:

$$\text{target\_q} = r + \gamma \max_{a'} Q_{\text{target}}(s', a')$$

Why Do We Need a Target Network?

- The target network is introduced to improve stability and convergence in Deep Q-Learning.
- Without it, the learning process can become unstable and diverge.

Stabilizing Updates

- Q-learning aims to minimize the difference between current Q-values and target Q-values.
- The target Q-value typically comes from the Bellman equation:

$$Q(s_t, a_t) = r_t + \gamma \max_{a'} Q(s_{t+1}, a')$$

- Using the same Q-network to generate both the predictions and targets can lead to instability.

Decoupling Target and Learning Network

- The target network is a separate copy of the Q-network.
- The main Q-network generates predictions, while the target network generates the target Q-values.
- The target network is updated less frequently (e.g., every few thousand steps).
- This decoupling improves stability in learning.

Slow Target Network Updates

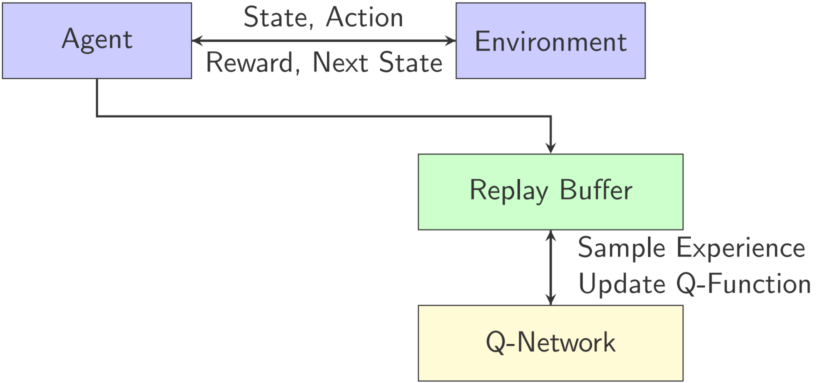
- The target network is updated at regular intervals, not after every training step.
- This slow update helps avoid instability from rapidly changing targets.
- It allows the main Q-network to adjust to a more stable target.

Addressing Temporal Difference (TD) Error

- Q-Learning is based on *Temporal Difference (TD) learning*.
- TD learning updates use the difference between current Q-values and the target.
- Since the target comes from some previous version of the Q-network, this difference is called *TD Error*.
- If the target is rapidly changing, the TD error may become large and destabilize learning.
- The target network helps reduce this by providing a stable target for a longer period.

Why Do We Need a Replay Buffer?

- The replay buffer is a key component of Deep Q-Learning (DQN).
- It helps improve the learning efficiency and stability of the agent.
- Without the replay buffer, training could be less stable and slower.



Stabilizing Training with Experience Replay

- In Deep Q-Learning, an agent learns from experiences: state, action, reward, next state.
- Directly updating the Q-network from each experience may introduce correlations between consecutive samples.
- These correlations can harm the learning process and lead to unstable updates.

Experience Replay

- The replay buffer stores past experiences (state, action, reward, next state) in a buffer.
- Randomly sampling experiences from this buffer helps break correlations between consecutive samples.
- This improves the stability of the updates by ensuring that the training data is more diverse.

Reducing Temporal Correlation

- Consecutive experiences in reinforcement learning are temporally correlated.
- If we use these correlated samples directly, the Q-network might overfit to recent experiences.
- By sampling randomly from the replay buffer, we reduce the impact of temporal correlation, leading to better convergence.

Stabilizing Q-Function Updates

- The Q-function is updated based on Temporal Difference (TD) errors.
- If the agent updates its Q-values using correlated data, the TD error can become large, making learning unstable.
- The replay buffer mitigates this by providing a diverse set of experiences, leading to more stable TD errors and smoother learning.

Efficiency in Learning

- The replay buffer allows the agent to reuse past experiences, improving sample efficiency.
- Without the buffer, each experience could be used only once, limiting the amount of learning from each sample.
- By storing experiences and sampling them multiple times, the agent can learn more efficiently.

How It Works in Practice

- The agent collects experiences and stores them in a buffer.
- At each training step, a mini-batch of experiences is sampled randomly from the buffer.
- This mini-batch is used to update the Q-network, helping the agent improve its policy.

Replay Buffer Summary

- The replay buffer stabilizes training by breaking correlations between consecutive experiences.
- It improves the efficiency of learning by allowing for the reuse of past experiences.
- By smoothing out updates, it helps the agent converge more reliably and quickly.

Enhancements to DQL

- Double DQN: Reduces overestimation bias.
- Dueling DQN: Separates state value and action advantage.
- Prioritized Experience Replay: Samples important transitions more frequently.

Summary

- Deep Q-Learning integrates deep learning with reinforcement learning.
- Uses neural networks to approximate Q-values.
- Replay buffers and target networks stabilize training.
- Foundation for advanced RL algorithms like Double DQN and DDPG.