Reinforcement Learning

Salar Mokhtari Laleh
Email: salarmokhtari0@gmail.com

***Abstract*—Reinforcement Learning (RL) has emerged as a powerful paradigm in machine learning, where an agent learns to make decisions by interacting with an environment to maximize cumulative rewards. Unlike supervised learning, which relies on labeled data, RL involves a trial-and-error approach where the agent's actions are guided by the feedback (rewards or penalties) it receives from the environment. This survey provides a detailed exploration of RL's core concepts, methodologies, algorithms, challenges, and real-world applications. We aim to clarify the foundational principles of RL, examine its mathematical underpinnings, and highlight the critical role RL plays in addressing complex decision-making problems across diverse domains.**

I. Introduction

Reinforcement Learning (RL) is a field of machine learning inspired by behavioral psychology, where an agent learns to perform tasks by interacting with its environment and receiving feedback in the form of rewards or penalties. RL is particularly suited to problems that involve sequential decision-making under uncertainty, where the outcomes of actions are not immediately known but unfold over time. The ultimate goal of the agent is to maximize its expected long-term reward by learning an optimal policy for decision-making.

RL has shown remarkable promise in applications such as robotics, gaming, healthcare, and autonomous systems, where agents must make decisions based on incomplete or dynamic information. Despite its success, RL remains a challenging and evolving area of research, particularly in environments with high-dimensional state spaces and long-term delayed rewards.

This survey aims to provide a comprehensive overview of RL by presenting its foundational concepts, key mathematical models, learning strategies, algorithms, challenges, and its practical applications in real-world problems.

II. The Reinforcement Learning Framework

At its core, RL can be formalized as a **Markov Decision Process (MDP)**, a mathematical model that captures the dynamics of sequential decision-making. An MDP is defined as a tuple (S, A, P, R, γ), where:

- S: The set of states, representing all possible configurations of the environment.
- A: The set of actions the agent can take in each state.
- $P(s'|s, a)$: The transition probability function that defines the probability of transitioning from state s to state s' after taking action a.
- $R(s, a)$: The reward function, which specifies the immediate reward obtained from taking action a in state s.
- γ: The discount factor that determines the relative importance of future rewards compared to immediate rewards.

The objective of the RL agent is to learn a **policy** $\pi(s)$ that maximizes the expected cumulative reward over time, where the cumulative reward is often called the **return**. The agent must make decisions to select actions that maximize the expected return, defined as:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k}$$

where G_t represents the return starting from time step t, and R_{t+k} is the reward at time step $t + k$.

III. Key Components of Reinforcement Learning

RL is based on several fundamental components that define the interaction between the agent and its environment:

- **Agent**: The decision-maker that interacts with the environment by taking actions based on its policy and receives feedback in the form of rewards.
- **Environment**: The external system with which the agent interacts. The environment evolves in response to the agent's actions, changing its state and providing rewards.
- **State (s)**: A representation of the current situation of the environment. The state may include information such as the position of objects, velocities, or any relevant aspect of the environment's configuration.
- **Action (a)**: The move or decision made by the agent, which can either be discrete (e.g., left, right, up, down) or continuous (e.g., speed, rotation).
- **Reward (r)**: A scalar feedback signal that indicates the immediate benefit of the agent's actions. It guides the agent's learning process and provides feedback on how good or bad the action was.
- **Policy ()**: The policy defines the agent's behavior. It maps states to actions and can be deterministic or stochastic.
- **Value Function (V)**: The value function estimates the expected return from a given state under a particular policy. It provides a measure of how good a state is for achieving long-term rewards.
- **Q-Function (Q)**: The action-value function estimates the expected return for a state-action pair. It evaluates the goodness of an action in a particular state.
- **Discount Factor ()**: The discount factor γ (where $0 \leq \gamma < 1$) determines the weight of future rewards compared to immediate rewards. A higher γ places more emphasis on long-term rewards.

IV. The Learning Process in Reinforcement Learning

The RL learning process is fundamentally characterized by the agent's exploration of the environment and its exploitation of knowledge gained over time. Two major components of this process include:

- **Exploration vs. Exploitation**: The agent faces a trade-off between exploration (trying new actions to gather information about the environment) and exploitation (using the best-known actions to maximize rewards). A well-balanced exploration-exploitation strategy is crucial for successful learning.
- **Learning from Interaction**: The agent improves its policy by interacting with the environment and receiving feedback. This iterative process involves continuously updating the agent's understanding of the environment to refine its decision-making policy.

The RL agent is often trained using iterative methods that update the agent's value functions or policy. Key methods include:

- **Value-Based Methods**: These methods focus on learning value functions, such as the state-value function $V(s)$ or the action-value function $Q(s,a)$, which help the agent estimate the best actions to take in each state.
- **Policy-Based Methods**: In contrast to value-based methods, policy-based methods focus on directly learning the policy $\pi(s)$ that maps states to actions.
- **Actor-Critic Methods**: These methods combine value-based and policy-based approaches. The **actor** learns the policy, while the **critic** evaluates the actions by learning the value function.

V. Bellman Equation and Optimal Policy

The Bellman equation forms the foundation of dynamic programming techniques in RL. It provides a recursive relationship for computing the value function and is critical for deriving optimal policies.

For the state-value function $V(s)$, the Bellman equation is given by:

$$V(s) = \mathbb{E}[R(s,a) + \gamma V(s')]$$

For the action-value function $Q(s,a)$, the equation becomes:

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

The optimal policy $\pi^*(s)$ can be derived by selecting the action that maximizes the value function or action-value function at each state:

$$\pi^*(s) = \arg\max_a Q^*(s,a)$$

where $Q^*(s,a)$ is the optimal action-value function.

VI. Categories of Reinforcement Learning Algorithms

Reinforcement learning algorithms can be broadly classified into the following categories based on their learning paradigms and methodologies:

- **Model-Free vs. Model-Based Methods**:
  - **Model-Free Methods**: These methods do not build or use a model of the environment. They focus on learning optimal policies or value functions directly from experience. Algorithms such as Q-Learning and Policy Gradient methods fall under this category.
  - **Model-Based Methods**: These algorithms construct a model of the environment (i.e., the transition probabilities and reward function) and use it to plan and make decisions. This approach can significantly reduce the sample complexity by leveraging the learned model for planning.
- **On-Policy vs. Off-Policy Methods**:
  - **On-Policy Methods**: These methods learn the value of the policy being executed by the agent. The policy is improved based on the agent's own experience. Examples include SARSA and Policy Gradient methods.
  - **Off-Policy Methods**: These methods learn from experiences generated by a different policy, which could be exploratory or based on a different behavior. They do not require the agent's current policy for learning. Examples include Q-Learning and Deep Q-Networks (DQN).
- **Value-Based vs. Policy-Based vs. Actor-Critic Methods**:
  - **Value-Based Methods**: These algorithms focus on estimating the value of states or state-action pairs. Q-learning and Deep Q-Networks (DQN) are examples.
  - **Policy-Based Methods**: These directly optimize the policy function without requiring a value function. Examples include REINFORCE and various policy gradient methods.
  - **Actor-Critic Methods**: These combine both value-based and policy-based approaches, where the actor updates the policy and the critic evaluates actions using a value function. Examples include A3C and Advantage Actor-Critic (A2C).
- **Offline vs. Online Learning**:
  - **Offline Learning**: The agent learns from a fixed dataset of experiences, which is beneficial in scenarios where real-time interaction with the environment is costly or impractical.
  - **Online Learning**: The agent continuously learns and updates its knowledge while interacting with the environment in real time.
- **Continuous vs. Discrete Action Spaces**:
  - **Discrete Action Spaces**: In these environments, the set of possible actions is finite and typically small.

Q-Learning and DQN are often used for problems with discrete actions.

- **Continuous Action Spaces**: These methods deal with environments where actions are continuous (e.g., robot arm movement). Policy gradient methods and Actor-Critic methods are commonly used in these settings.