# Reinforcement Learning

By : Jamuna S Murthy

# Overview

Define Reinforcement Learning. Explain the components of a reinforcement learning problem.
What is Reinforcement Learning and explain the Reinforcement learning problem with a neat diagram?

**Definition**: Learning process where an agent learns to make optimal decisions by interacting with its environment.

- **Goal**: Maximize cumulative rewards over time.

**2. Applications**

- Controlling **mobile robots**.
- Optimizing **factory operations**.
- Playing **board games**.

**3. Reward Mechanism**

- **Positive reward**: For desirable outcomes (e.g., winning a game).
- **Negative reward**: For undesirable outcomes (e.g., losing a game).
- **Zero reward**: For neutral states.

# Overview

## 4. Challenges

- **Delayed Rewards**:
  - Rewards are not immediate; the agent must infer long-term benefits of actions.
- **Indirect Feedback**:
  - Feedback is sparse and indirect, requiring exploration.

## 5. Q-Learning

- **Key Algorithm**: Acquires optimal control strategies through delayed rewards.
- **Features**:
  - No prior knowledge of environment required.

# Introduction

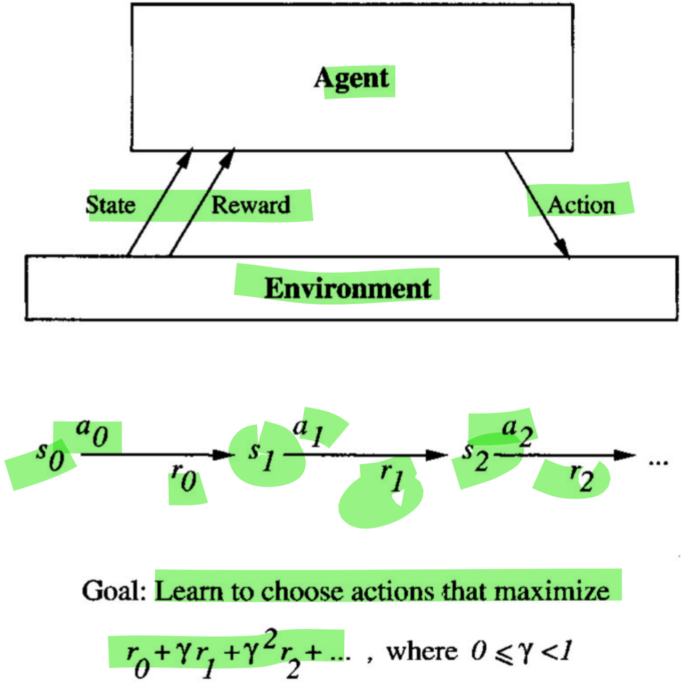


**FIGURE 13.1**
An agent interacting with its environment. The agent exists in an environment described by some set of possible states $S$. It can perform any of a set of possible actions $A$. Each time it performs an action $a_t$ in some state $s_t$ the agent receives a real-valued reward $r_t$ that indicates the immediate value of this state-action transition. This produces a sequence of states $s_i$, actions $a_i$, and immediate rewards $r_i$ as shown in the figure. The agent's task is to learn a control policy, $\pi : S \rightarrow A$, that maximizes the expected sum of these rewards, with future rewards discounted exponentially by their delay.

# Learning Task

### 1. Overview of the Learning Task

- **Objective**: Learn an optimal policy ($\pi^*$) that maximizes cumulative rewards over time.
- **Scenarios**: Agent may face deterministic or non-deterministic outcomes and may or may not predict the next state.

### 2. Markov Decision Process (MDP)

- **MDP Definition**: A mathematical framework to model the learning task.
- **Key Components**:
  1. **States ( S )**: Set of all possible states the agent can perceive.
  2. **Actions ( A )**: Set of all actions the agent can perform.
  3. **Reward Function ( $r(s, a)$ )**: Immediate reward for performing action $a$ in state $s$.
  4. **State Transition Function ( $\delta(s, a)$ )**: Defines the next state $s_{t+1}$ after taking action $a$ in state $s$.

# Learning Task

### 3. Agent's Task

- Learn a **policy** $\pi(s) = a$, which specifies the action $a$ to take in state $s$.
- **Optimal Policy**: Maximizes cumulative rewards starting from any state $s$.

### 4. Cumulative Reward (Discounted)

- **Formula**:
  - V^π(s) : Cumulative reward following policy $\pi$ from state $s$.
  - $\gamma$ : Discount factor $(0 \leq \gamma < 1)$ , determines the importance of future rewards.
  - r(st, at) : Reward at time $t$ for state $s_t$ and action $a_t$.
- **Interpretation**:
  - **$\gamma = 0$**: Considers only immediate rewards.
  - **$\gamma \to 1$**: Future rewards have nearly equal weight as immediate rewards

# Learning Task

## 5. Optimal Policy and Value Function

- **Optimal Policy**:
  - Maximizes $V^\pi(s)$ for all states s.
- **Optimal Value Function**:
  - Represents the maximum cumulative reward achievable from state s.

# Q Learning

Explain the Q function and Q Learning Algorithm assuming deterministic rewards and actions with example.
Define Q-learning algorithm. Explain how Q-learning helps an agent make decisions.

## Q-Function

- **Definition**: Measures the expected utility of performing action $a$ in state $s$ and following the optimal policy thereafter.
- **Formula**:
  - $Q^*(s, a)$ : Optimal Q-value for state-action pair.
  - $r(s, a)$ : Immediate reward for action $a$ in state $s$.
  - $P(s' \mid s, a)$ : Transition probability to state $s'$ from state $s$ after action $a$.
  - $\gamma$ : Discount factor.

# Q Learning

## Q-Learning Algorithm

- **Objective**: Iteratively learn Q-values to approximate Q^*(s, a) without needing a model of the environment.
- **Update Rule**:
  - α : Learning rate ($0 < \alpha \leq 1$).
  - r : Immediate reward.
  - γ : Discount factor.
  - max_{a'} Q(s', a') : Maximum Q-value for the next state.

# Q Learning

**Steps in the Q-Learning Algorithm**

1. **Initialize**: Q-values arbitrarily (e.g., zero) for all state-action pairs.
2. **Observe State**: Start at an initial state s.
3. **Choose Action**: Select an action a using an exploration strategy (e.g., ε-greedy).
4. **Take Action**: Perform the action a and observe the reward r and next state s'.
5. **Update Q-Value**: Apply the Q-learning update rule.
6. **Repeat**: Iterate until convergence or a stopping criterion is met.

# Q Learning

**Example: Grid-World Q-Learning**

- **Scenario**: Same grid-world environment as described earlier.
- **Process**:
  1. Agent starts in a random state.
  2. Selects an action using an ε-greedy policy.
  3. Updates Q-values based on observed rewards and transitions.
  4. Over time, learns the optimal policy leading to state G.

# Questions ?