# Week 2: Foundations of RL

Your Name

Your Institution

July 19, 2025

# Introduction to Foundations of Reinforcement Learning

## Overview

Reinforcement Learning (RL) is a branch of machine learning focused on how agents take actions in an environment to maximize cumulative rewards.

- Importance of understanding key concepts:
  - Agents
  - Environments
  - Rewards
  - Policies

# Key Concepts in Reinforcement Learning

1. **Agents**
   - **Definition:** An entity that interacts with the environment.
   - **Example:** A self-driving car navigating roads.
2. **Environments**
   - **Definition:** The context in which the agent operates.
   - **Example:** The traffic system with vehicles and pedestrians.

# Key Concepts Continuation

3. **Rewards**
   - **Definition:** Feedback signals that guide the agent's learning.
   - **Example:** Positive reward for successfully navigating a turn, negative penalty for running a red light.
4. **Policies**
   - **Definition:** A strategy used by an agent to determine actions based on the environment's state.
   - **Example:** Policies governing stopping for red lights and yielding to pedestrians.

# Key Points and Example

## Key Points to Emphasize

- The **agent-environment framework** is fundamental in RL.
- Understanding **rewards and policies** is crucial for effective learning.
- The interplay of these elements enables agents to adapt and optimize their behavior.

## Illustrative Example: Chess

- **Agent:** The player or chess algorithm.
- **Environment:** The chessboard and pieces.
- **Rewards:** Points for winning, losing, or capturing pieces.
- **Policy:** Strategy for selecting moves based on the board configuration.

# Overview of Agents in RL - Definition of Agents

In Reinforcement Learning (RL), an **Agent** is an entity that makes decisions by interacting with an environment to achieve a specific goal. The agent observes the state of the environment, takes actions based on these observations, and receives feedback in the form of rewards or penalties.

- **Autonomy:** Operates without direct human oversight.
- **Adaptability:** Learns from experiences to improve decision-making.
- **Goal-oriented:** Aims to maximize cumulative rewards over time.

# Overview of Agents in RL - Roles of Agents

The roles of agents in the RL process can be summarized as follows:

1. **Observation:** The agent observes the current state of the environment.
2. **Action Selection:** Selecting an action based on the observation, guided by a policy.
3. **Interacting with Environment:** Performing the chosen action affects the environment.
4. **Receiving Feedback:** Gaining rewards or penalties post-action.
5. **Updating Knowledge:** Using feedback to adjust the policy and improve future decisions.

# Overview of Agents in RL - Example and Key Points

**Example of an Agent in a Grid World:**

- **State (S):** Position on the grid.
- **Action (A):** Possible moves (up, down, left, right).
- **Reward (R):** +10 points for reaching a destination; -5 points for falling into a trap.

**Key Points to Emphasize:**

- Learning and adaptability of agents are central to RL.
- Balance between exploration (trying new actions) and exploitation (maximizing known rewards).
- Design of an agent's policy influences its performance significantly.

**Conclusion:** Agents are crucial in RL as they learn from their environment through actions and rewards.

# Understanding Environments

## What is an Environment in RL?

In reinforcement learning, the **environment** is the context in which an agent operates. It includes everything the agent interacts with to learn and make decisions and provides feedback based on the agent's actions.

# Key Components of the Environment

1. **State (s)**: Represents the current situation of the environment.
   - Example: In a chess game, the layout of pieces.
2. **Action (a)**: Set of possible actions the agent can take.
   - Example: In driving simulation, turning, accelerating, braking.
3. **Transition Probability (P)**: Likelihood of moving from one state to another given an action.
   - Example: In a grid world, moving east has an 80% chance of success.
4. **Reward (r)**: Feedback value received after action execution.
   - Example: Winning a round gives a +1 reward.

# Agent-Environment Interaction Cycle

- **Observation**: The agent observes the current state of the environment.
- **Action Selection**: The agent selects an action based on the observed state.
- **Feedback**: The agent receives a reward and transitions to a new state.
- **Learning**: The agent updates its strategy based on the received feedback.

## Example: An RL Agent in a Maze

- **Environment**: A maze with walls and paths.
- **State**: Agent's current position.
- **Actions**: Move up, down, left, right.
- **Rewards**: Reaching the exit gives a reward, hitting a wall incurs a penalty.

# Rewards in Reinforcement Learning - Overview

## What Are Rewards?

In Reinforcement Learning (RL), rewards are signals that provide feedback to an agent based on its actions within an environment.

- Positive feedback for desirable behavior.
- Negative feedback for undesirable behavior.

Mathematically, a reward at time $t$ is represented as $r_t$.

# Significance of Rewards

1. **Guiding Learning:**
   - Rewards direct the agent towards achieving specific goals.
   - They help agents to learn which actions yield the best outcomes, optimizing behavior to maximize cumulative rewards (the return).
2. **Establishing Preferences:**
   - Differentiates helpful actions from harmful ones.
   - In environments with delayed rewards, understanding the sequence of actions leading to a reward is crucial.
3. **Influencing Decisions:**
   - Rewards impact the strategy adopted by the agent, informing whether actions were beneficial or detrimental.
4. **Example:**
   - In a game (e.g., chess):
     - Positive reward (+10) for winning.
     - Negative reward (-10) for losing.
     - Smaller rewards for advantageous positions.

# How Rewards Influence Agent Behavior

- **Exploration vs. Exploitation:**
  - Agents balance exploring new actions and utilizing known rewarding actions. Poor reward design can lead to early favoring of exploitation.
- **Temporal Difference Learning:**

$$Q(s, a) = r + \gamma \max_{a} Q(s', a) \tag{1}$$

  - Here, $Q(s, a)$ is the quality of action $a$ in state $s$, $r$ is the immediate reward, and $\gamma$ is the discount factor.
- **Sparse vs. Dense Rewards:**
  - **Sparse Rewards:** Long sequences without rewards hinder learning (e.g., a maze with a reward only at the exit).
  - **Dense Rewards:** Frequent guidance can speed up learning but may confuse the agent if over-rewarded.

# Key Takeaways and Conclusion

- Rewards drive the learning process in RL.
- Effective reward design influences the agent's strategy and efficiency.
- Exploration strategies must align with the reward structure to optimize learning outcomes.

## Conclusion

Rewards are fundamental in reinforcement learning, guiding agents toward desirable behaviors and impacting decision-making. Understanding and designing reward systems are vital for the success of RL applications.

# Policies: Directives for Action

## Understanding Policies in Reinforcement Learning (RL)

A policy is a strategy employed by an RL agent to determine its actions based on the current state of the environment. It is a mapping from states to actions and can be either deterministic or stochastic.

# Definition of a Policy

- **Deterministic Policy:**

$$\pi(s) = a$$

Where $\pi$ is the policy function, $s$ is a state, and $a$ is the action taken in that state.

- **Stochastic Policy:**

$$\pi(a|s) = P(A = a|S = s)$$

The probability of taking action $a$ in state $s$ defines the variability in the agent's action choice.

## Role of Policies

Policies guide the agent's decisions throughout the learning process, influencing how it perceives and interacts with its environment to maximize cumulative rewards over time.

# Examples to Illustrate Policies

1. **Example 1: Grid World**
   - Deterministic Policy: The agent follows a specific path every time it starts in the same state.
   - Stochastic Policy: The agent might choose to move in various directions based on a probability distribution (e.g., 70% chance to move up, 30% chance to move right).
2. **Example 2: Self-Driving Cars**
   - A car's policy determines how to respond in various traffic situations (e.g., stop at traffic lights, yield to pedestrians).
   - The policy evolves as the car learns from experiences and real-time data.

## Conclusion

The policy is essential in RL, defining how an agent interacts with its environment. Optimizing policies leads to better decision-making and improved performance.

# Exploration vs. Exploitation Dilemma

## Understanding the Dilemma

In reinforcement learning (RL), the agent faces the **exploration vs. exploitation** dilemma when making decisions about how to act in an environment.

# Exploration vs. Exploitation Dilemma - Exploration

- **Exploration**: Trying new actions to discover potential rewards.
- **Goal**: Gather information to improve future decision-making.
- **Example**: In chess, trying a new opening for potential long-term benefits.

# Exploration vs. Exploitation Dilemma - Exploitation

- **Exploitation**: Choosing known actions believed to yield the highest reward.
- **Goal**: Leverage known information to maximize short-term benefits.
- **Example**: Continuing with a successful chess opening rather than risking a new strategy.

# Implications of the Dilemma

- Balancing exploration and exploitation is critical in RL.
- **Key Considerations**:
  1. Long-Term vs. Short-Term Rewards
  2. Learning Rate
  3. Environment Complexity

# Key Strategies to Address the Dilemma

- **Epsilon-Greedy Strategy**:
  - Best-known action most of the time, random action with probability $\epsilon$.
  - **Example**: $\epsilon = 0.1 \implies$ 10% chance to explore.
- **Upper Confidence Bound (UCB)**:

$$UCB(a) = \overline{X}_a + c\sqrt{\frac{\ln(n)}{n_a}} \tag{2}$$

  - $\overline{X}_a$: average reward of action $a$
  - $n$: total actions taken
  - $n_a$: number of times action $a$ has been taken
  - $c$: exploration parameter
- **Thompson Sampling**:
  - Selecting actions based on probability distributions over expected rewards.

# Conclusion

## Summary

The exploration vs. exploitation dilemma is critical for RL success. Finding the right balance enhances learning efficiency and improves decision-making capabilities, leading to better performance in complex environments.

# Value Functions Overview

## Understanding Value Functions

Value functions are fundamental components of reinforcement learning (RL) that help estimate the expected future rewards an agent can receive from being in a particular state or taking a specific action. They are essential for guiding the agent's decision-making process.

# Types of Value Functions

1. **State Value Function (V(s))**:
   - Represents the expected return from a given state following a certain policy.
   - **Formula**:
   $$V^{\pi}(s) = \mathbb{E}_{\pi}[R_t | S_t = s] \tag{3}$$
   - **Explanation**: Where $V^{\pi}(s)$ is the value of state $s$ and $R_t$ is the return received after time $t$.
2. **Action Value Function (Q(s, a))**:
   - Represents the expected return from taking a specific action $a$ in state $s$ and then following a particular policy.
   - **Formula**:
   $$Q^{\pi}(s, a) = \mathbb{E}_{\pi}[R_t | S_t = s, A_t = a] \tag{4}$$
   - **Explanation**: Here, $Q^{\pi}(s, a)$ gives the value of taking action $a$ in state $s$.

# Importance of Value Functions

- **Decision-Making**: They guide the agent's actions by estimating long-term benefits.
- **Efficient Learning**: Aid in updating knowledge of actions and states for faster convergence on optimal policies.
- **Handling Uncertainty**: Allow the agent to express uncertainty in its environment and adapt accordingly.

# Example in Context

## Chess Game Scenario

- **State**: The current arrangement of pieces on the board.
- **Action**: Moving a piece, for example, moving a knight.
- **Value Function**: Assists the AI in evaluating immediate gains as well as potential future scenarios resulting from that move.

# Key Points to Emphasize

- Value functions simplify decision-making by quantifying potential rewards.
- They are critical for exploring new strategies and exploiting known successful actions.
- The relationship between state value and action value functions enhances comprehension of the environment.

## Conclusion

By understanding value functions, students will gain insights into how reinforcement learning algorithms enhance decision-making processes. This concept is closely tied to the upcoming topic on Markov Decision Processes.

# Markov Decision Processes (MDPs)

## Overview of MDPs

A Markov Decision Process (MDP) is a mathematical framework used for modeling decision making in environments. MDPs formalize making a sequence of decisions under uncertainty, applicable in various fields including robotics, finance, and game playing.

# Components of MDP

An MDP is defined by the following components:

1. **States (S)**: A finite set of all possible situations.
2. **Actions (A)**: A finite set of actions available in each state.
3. **Transition Function (P)**: Defines the probability of transitioning from one state to another given an action ($P(s'|s, a)$).
4. **Reward Function (R)**: Defines the immediate reward after transitioning from state $s$ to $s'$ upon taking action $a$ ($R(s, a, s')$).
5. **Discount Factor ($\gamma$)**: Between 0 and 1, it governs the importance of future rewards.

# How MDPs Work

Using MDPs, an agent can analyze its environment and choose actions to maximize cumulative rewards over time. The Markov property indicates that future states depend only on the current state and action, not on prior histories.

## Key Points to Emphasize

- MDPs model complex decision-making environments effectively.
- The interrelationship among states, actions, transitions, rewards, and discount factors is essential for solving MDPs.
- The Markov property simplifies decision-making, allowing efficient algorithm development for optimal policies.

# Example of an MDP

**Scenario: A robot navigating through a maze.**

- **States (S)**: Each position in the maze.
- **Actions (A)**: Move up, down, left, right.
- **Transition Function (P)**: Likelihood of moving to intended cell vs. hitting a wall.
- **Reward Function (R)**: +10 for reaching the exit, -1 for each move, -5 for hitting an obstacle.
- **Discount Factor ($\gamma$)**: 0.95 to prioritize earlier rewards.

# Conclusion

MDPs are foundational in understanding reinforcement learning. They establish the structure for problem formulation that enables the development of algorithms to discover optimal decision-making strategies under uncertainty. Moving forward, we will explore concepts like the Bellman equations critical for solving MDPs.

# Bellman Equations Fundamentals

## Overview

Bellman Equations are foundational principles in Reinforcement Learning (RL) that describe the relationship between the value of a state and the values of its successor states.

# Key Concepts

1. **Value Function (V)**

$$V(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t \mid S_0 = s\right]$$

2. **Action Value Function (Q)**

$$Q(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t \mid S_0 = s, A_0 = a\right]$$

3. **Bellman Equation for Value Functions**

$$V(s) = \sum_{a \in A} \pi(a|s) \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma V(s')\right]$$

4. **Bellman Optimality Equation**

$$V^*(s) = \max_a \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma V^*(s')\right]$$

# Example: Simplified MDP

Consider a grid world where an agent moves right or down to reach a goal. The grid cells represent states, and the allowed actions determine transitions and rewards. Using the Bellman equation, we calculate:

$$V(0,0) = \frac{1}{2}\left[R(0,1) + \gamma V(0,1)\right] + \frac{1}{2}\left[R(1,0) + \gamma V(1,0)\right]$$

# Key Points to Emphasize

- **Importance:** Bellman equations are crucial for deriving optimal policies and value functions in RL.
- **Recursive Structure:** They utilize recursion to define the value at a state concerning future values.
- **Connection to Policy:** They underpin critical RL algorithms (e.g., Value Iteration, Policy Iteration).

# Conclusion

Understanding Bellman Equations is essential for mastering RL. They form the theoretical backbone for various algorithms used to find optimal strategies in MDPs, enabling agents to learn effectively in complex environments.

# Conclusion and Key Takeaways - Overview

- Summary of foundational concepts discussed in Reinforcement Learning (RL).
- Key takeaways encapsulating our discussions through the week.

# Key Concepts in Reinforcement Learning

1. **Fundamentals of RL**
   - Definition: An agent learns to make decisions by interacting with an environment to maximize cumulative rewards.
   - Key components: Agent, Environment, Action, Reward.
2. **Trial and Error**
   - Exploration vs. Exploitation: The balance between trying new actions and leveraging known actions is crucial.

# Key Concepts Continues

3. **Bellman Equations**
   - Define the value function, critical for determining future rewards.
   - Formula:

$$V(s) = \max_a \left( R(s,a) + \gamma \sum_{s'} P(s'|s,a) V(s') \right) \tag{5}$$

4. **Applications of RL**
   - Fields: Robotics, Game Playing, Recommendation Systems, Finance.

# Key Points to Remember

- Importance of the environment for effective learning.
- Focus on long-term goals over immediate rewards.
- Robustness of RL techniques: Policy Gradient methods, Q-learning, and DQNs.
- Mastering these principles is essential for advanced RL techniques.