****

Reinforcement learning Assignment

Artificial Intelligence

|  |  |
| --- | --- |
| **Team Members:** | |
| **7818** | **Mohamed Hussein** |
| **7925** | **Fares Amin** |
| **7721** | **Alaa Elkhouly** |

### What is a Markov Decision Process (MDP)?

An MDP is a mathematical framework for modeling decision-making in situations where outcomes are partly random and partly controlled by a decision-maker. Its key components are:

1. **States (S):** Possible situations in the system, e.g., positions in a 3x3 grid.
2. **Actions (A):** Choices available at each state, like moving Up, Down, Left, or Right.
3. **Transition Model (P):** Probabilities of moving to new states after actions. For instance, moving in the intended direction 80% of the time.
4. **Rewards (R):** Immediate payoff or penalty for being in a state or taking an action, such as +10 in a top-right cell.
5. **Discount Factor (γ):** Determines the importance of future rewards. A higher γ values future rewards more.

The goal of an MDP is to find a **policy** (set of actions) that maximizes long-term rewards.

### What is a Policy?

A policy, denoted as π(s), provides the agent with the action to take in each state to maximize rewards. It acts as a guide for the agent.

#### Types of Policies

1. **Random Policy:** Initial policy where actions are chosen randomly.
2. **Deterministic Policy:** Fixed rules that suggest specific actions for each state.
3. **Optimal Policy:** The best policy to maximize total rewards, computed through Value Iteration or Policy Iteration.

#### How Policies are Derived

Policies are based on state values, which indicate how rewarding each state is. The agent selects actions that lead to the highest-value states.

1. **Value Iteration:** Calculates state values first, then derives the optimal policy.
2. **Policy Iteration:** Starts with a random policy, refines it by evaluating state values and improving actions until it stabilizes.

#### Importance of Policies

1. **Efficiency:** Once computed, the agent can follow the policy without recalculating decisions.
2. **Optimality:** Ensures consistent reward maximization.
3. **Adaptability:** Policies can adjust if the environment changes.

### Solving the MDP: Value Iteration and Policy Iteration

Both algorithms determine the optimal policy for the agent.

#### 1. Value Iteration

Value Iteration iteratively updates state values until they stabilize:

1. **Initialize Values:** Set all state values to zero.
2. **Update Values:**
   * For each state, calculate the expected value of all possible actions:
     + Account for transition probabilities, rewards, and discounted future values.
   * Assign the highest action value to the state.
3. **Repeat:** Continue until state values converge (stop changing significantly).
4. **Extract Policy:** Once stable, determine the optimal action for each state.

#### 2. Policy Iteration

Policy Iteration alternates between refining a policy and evaluating it:

1. **Start with a Random Policy:** Assign random actions to states.
2. **Policy Evaluation:**
   * Compute the value of each state under the current policy by updating state values repeatedly.
3. **Policy Improvement:**
   * Compare actions to find better ones for each state.
   * Update the policy with actions that maximize value.
4. **Repeat:** Continue until the policy stabilizes and becomes optimal.

### Value Iteration vs. Policy Iteration

#### Key Similarities

* Both solve MDPs to find the optimal policy.
* Both rely on the Bellman Optimality Equation.

#### Key Difference

* **Value Iteration:** Focuses on calculating state values first, then derives the policy.
* **Policy Iteration:** Starts with a policy, alternates between evaluating and improving it until convergence.

#### Step-by-Step Comparison

| **Aspect** | **Value Iteration** | **Policy Iteration** |
| --- | --- | --- |
| **Focus** | State values first, policy second | Policy refinement through iterations |
| **Initial Step** | All state values set to zero | Start with a random policy |
| **Iterations** | Update state values to convergence | Alternate between evaluation and improvement |
| **Output** | Optimal policy derived from stable values | Optimal policy directly |

#### Advantages

1. **Value Iteration:**
   * Simpler implementation.
   * Faster for small problems.
2. **Policy Iteration:**
   * Structured and robust.
   * Often faster for large problems in terms of iterations.

### Why Both Lead to the Same Result

Both algorithms aim to solve for the same optimal policy, π\*, and rely on the Bellman Optimality Equation:

This equation ensures that state values and actions are optimized for long-term rewards.

#### Path to Convergence

1. **Value Iteration:** Iteratively refines state values and derives the policy after stabilization.
2. **Policy Iteration:** Alternates between policy evaluation and improvement, refining both state values and the policy step by step.

#### Conditions for Convergence

Both algorithms converge to the same result if:

* The discount factor γ<1, ensuring finite rewards.
* The state and action spaces are finite.

### Summary

1. An MDP models decision-making under uncertainty using states, actions, rewards, and probabilities.
2. The goal is to find an optimal policy that maximizes long-term rewards.
3. **Value Iteration** and **Policy Iteration** are two algorithms to compute the optimal policy:
   * Value Iteration focuses on refining state values first.
   * Policy Iteration alternates between evaluating and improving policies.
4. Both algorithms converge to the same optimal policy if properly implemented.

**Outputs**

**Value and Policy Iteration for r = 100**

* **Values:** The high values represent the substantial reward of 100 for reaching the top-left cell. As the agent moves farther from this reward, the values decrease due to the discounting of future rewards.
* **Policy:** The optimal policy directs the agent upwards and towards the left, aligning with the goal of maximizing the reward at the top-left cell. Minor differences between value iteration and policy iteration policies may occur due to rounding or tie-breaking.
* **Incentive:** The terminal state (state 0) with its high positive reward incentivizes the agent to prioritize this state.
* **Behavior:** The agent avoids distractions or long paths, creating a direct policy focused on reaching the high-reward state.

**Value and Policy Iteration for r = 3**

* **Values:** Lower than those for r = 100, reflecting a moderate reward of 3. The values still propagate across the grid but with less magnitude.
* **Policy:** The optimal policy now directs the agent towards the top-right terminal reward (10). The moderate reward at the top-left does not dominate decision-making, allowing the agent to focus on the larger terminal reward.
* **Incentive:** While state 0 is desirable, its reward does not overshadow others, leading to a balanced policy.
* **Behavior:** The agent moves toward state 0 but less deterministically, considering alternative paths.

**Value and Policy Iteration for r = 0**

* **Values:** Slightly lower, as state rewards contribute less. The terminal reward of 10 becomes the primary driver of values.
* **Policy:** Without a significant incentive for the top-left cell, the policy guides the agent toward the top-right reward (10).
* **Incentive:** With no advantage at state 0, the agent’s decisions are influenced mainly by the grid structure and discount factor.
* **Behavior:** The agent takes paths without urgency, as all non-terminal states hold equal value.

**Value and Policy Iteration for r = -3**

* **Values:** These further decrease due to penalties for each step, highlighting the importance of shorter paths to terminal rewards.
* **Policy:** The agent minimizes time spent in the grid by moving directly to the nearest reward (top-right), avoiding unnecessary detours.
* **Incentive:** A negative reward at state 0 makes it less desirable, prompting the agent to prioritize the alternative terminal state (10).
* **Behavior:** The agent actively avoids paths leading to state 0, resulting in a more circuitous but efficient strategy.

**Observations**

* **Convergence:** Both value iteration and policy iteration converge closely, ensuring the algorithms work correctly.
* **Effect of r:**
  + Higher r: Encourages the agent to maximize rewards by reaching state 0 faster.
  + Moderate r: Balances exploration, allowing the agent to weigh rewards more evenly.
  + Negative r: Discourages reaching state 0, with the agent prioritizing the higher reward at the alternative terminal state.
* **Behavioral Adaptation:** As r decreases or becomes negative, the agent adapts its policy to minimize penalties, reflecting a shift in priorities based on rewards.