Cairo University

Faculty of Engineering

Computer Engineering Department

CMPS458 Reinforcement Learning Report
Assignment 1 Policy Iteration

Team Name/Number:
Mohamed Ahmed Ibrahim Sobh 1210288

Omar Ahmed Ibrahim 1210020

Supervisor: Ayman AboElhassan

October 23, 2025

Deliverables

Repo link: Github Repository
Video record link: add link here

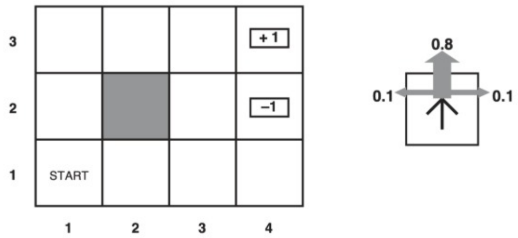

Figure 1: Example grid-world environment used for testing.

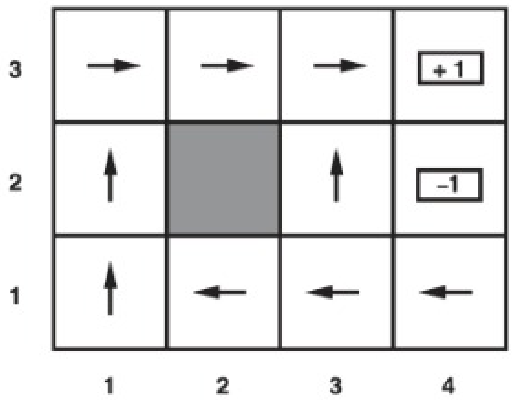

Figure 2: Example of a learned policy after convergence.

Table 1: Core parameters used in the policy iteration experiments.

Parameter	Symbol	Default Value	Description
Discount factor	γ	0.9	Future reward weighting
Convergence threshold	θ	1×10^{-6}	Stopping criterion for evaluation
Maximum iterations	N_{iter}^{max}	1000	Upper limit on policy updates
Step reward	R_{step}	-1	Penalty for each move
Goal reward	R_{goal}	10	Reward for reaching the goal state
Bad cell penalty	R_{bad}	-10	Penalty for entering a bad cell

Discussion

0.1 Experiments

Table 2: Effect of varying the discount factor γ on policy iteration results (seed=40501, maze=5, bad=2, correct hit=0.7 BUT the simulation steps are based on policy only).

γ	Observation	Outcome / Behavior
0.3	Agent focuses on immediate rewards	5 Policy Iterations AND 7 Simulation Steps
0.9	Balances short and long-term rewards	4 Policy Iterations AND 5 Simulation Steps

Table 3: Effect of convergence threshold θ on stability and convergence (maze=100, bad=40).

θ	Observation	Outcome / Behavior
10^{-1}	Fast convergence but less accurate	116 Iters to reach Stable Policy, Time taken = 15.75s
10^{-6}	Slower convergence with high precision	83 Iters to reach Stable Policy, Time taken = 19.57s

Table 4: Effect of changing reward values on agent behavior and learning outcome (maze=40, bads=20).

Reward Configuration	Observation	Expected Outcome
Goal=5, Bad=-1, Step=-0.1	Small penalty difference causes uncertain exploration	25 Policy Steps AND 55 Simulation Steps
Goal=10, Bad=-10, Step=-1	Balanced reward setup encourages stable learning	15 Policy Steps AND 55 Simulation Steps
Goal=20, Bad=-50, Step=-1	Strong contrast between rewards and penalties	20 Policy Steps AND 57 Simulation Steps

0.2 Question Answers

1. **What is the state-space size of the 5x5 Grid Maze problem?**
The state space size depends on how we define the state. In our Grid Maze environment, the state consists of Agent, Goal, Bad cell 1, and Bad cell 2 coordinates. For a 5×5 grid, we have 25 possible positions. Since we have 4 unique positions, we can say that the state space size is $25 \times 24 \times 23 \times 22 =$ **303,600**. If we fix the goal and bad cell positions then the state space size is 25 states only

2. **How to optimize the policy iteration for the Grid Maze problem?**

   (a) **Modified Policy Iteration**
   - Do not run policy evaluation to full convergence - Use a fixed number of evaluation sweeps (e.g., k = 3) before improvement - This balances evaluation accuracy with computational speed

   (b) **Early Termination**
   - Set a practical threshold (theta) for policy evaluation convergence - Stop evaluation when changes are negligible (e.g., θ=1e-4)

   (c) **Exploiting Problem Structure**
   - Use Manhattan distance heuristics to initialize value function - Initialize policy to point toward the goal - Skip evaluation for terminal states (goal and bad cells)

   (d) **Asynchronous Updates**
   - Update states in-place during policy evaluation instead of using a separate copy - Prioritize updates for states more likely to be visited - Focus computation on the "critical path" from start to goal

   (e) **Parallel Processing**
   - Evaluate multiple states simultaneously using vectorization - NumPy operations can process the entire grid at once

3. **How many iterations did it take to converge on a stable policy for 5x5 maze?**

   The exact number depends on:

   (a) **Initial policy**: Random initialization may take longer

   (b) **Maze configuration**: Distance from start to goal affects convergence

   (c) **Discount factor (γ)**: Higher values (closer to 1.0) may require more iterations

   (d) **Convergence threshold (θ)**: Stricter thresholds require more iterations

   Using seed 40501, 10, -1, -10 reward system, 0.9 discount factor, and convergence factor 1e-6. Convergence was achieved in **4 iterations** of policy evaluation and improvement cycles.

4. **Explain, with an example, how policy iteration behaves with multiple goal cells.**
Policy iteration can handle multiple goal cells by forming a **multi-objective optimization problem**, where the agent learns to reach the **nearest** or **most valuable** goal.

   Scenario: 5×5 grid with two goal cells (G_1 and G_2):

```
G1  .   .   .   .
 .  X   .   X   .
 .  .   S   .   .
 .  .   .   .   .
 .  .   .   .  G2
```

1. Modified Reward Function:

```
def _get_reward(self, pos):
    if pos in self.goal_positions:  # instead of pos == self.goal_pos
        return 100.0
    elif pos in self.bad_cells:
        return -100.0
    else:
        return -1.0
```

2. Value Function Behavior:

- The value function develops **multiple peaks** (one at each goal).
- States closer to any goal have higher values.
- The optimal policy routes agents to the **nearest goal**.

3. Policy Convergence:

- The grid divides into **regions of attraction**.
- Each region's policy points toward the nearest goal.
- The boundary between regions depends on distance, obstacles, and transition probabilities.

Example (Equal Rewards): Both G_1 and G_2 give $+100$ reward.

```
G1  →   →   ↓   ↓
↑   →   →   ↓   ↓
↑   ↑   S   ↓   ↓
↑   ↑   ←   ↓   ↓
↑   ←   ←   ←  G2
```

Observations:

- Top-left states move toward G_1, bottom-right toward G_2.
- A diagonal boundary separates the two regions.
- The agent always takes the shortest path.

Example (Different Rewards): If $G_1 = +100$ and $G_2 = +200$:

- More states route toward G_2.
- The boundary shifts toward G_1.
- Only nearby states prefer G_1.

Mathematical Explanation:

For each state s, policy improvement follows:

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

With multiple goals, $V(s')$ reflects the best reachable goal's return, so the policy naturally guides the agent to the **optimal goal**.

5. **Can policy iteration work on a 10x10 maze? Explain why.**
Yes, policy iteration can work on a 10×10 maze, but the computational cost grows quadratically with the number of states (100 states $\rightarrow$ 400 state-action pairs). It remains feasible, but slower.

6. **Can policy iteration work on a continuous-space maze? Explain why.**
No, standard policy iteration assumes a discrete and finite state space. In a continuous environment, the number of possible states is infinite, making it impossible to represent the value function as a lookup table.

7. **Can policy iteration work with moving bad cells (like Pac-Man moving ghosts)? Explain why.**
Not directly. Policy iteration assumes a stationary environment (fixed transition probabilities). If bad cells move, the environment becomes *non-stationary*.