Reinforcement learning and Optimal control for Robotics (ROB-GY 6323)

New York University, Fall 2024

Due on 11/14, 11:59 PM

Name: Raman Kumar Jha
NYU ID: N13866145

Exercise series 3

For questions requesting a written answer, please provide a detailed explanation and typeset answers (e.g. using LaTeX[1]). Include plots where requested in the answers (or in a Jupyter notebook where relevant). For questions requesting a software implementation, please provide your code in runnable Jupyter Notebook. Include comments explaining how the functions work and how the code should be run if necessary. Code that does not run out of the box will be considered invalid.

Exercise 1[40 points]

a) Consider the following dynamical system:

$$x_{n+1} = \begin{cases} -x_n + 1 + u_n & \text{if } -2 \leq -x_n + 1 + u_n \leq 2 \\ 2 & \text{if } -x_n + 1 + u_n > 2 \\ -2 & \text{otherwise} \end{cases}$$

where $x_n \in \{-2, -1, 0, 1, 2\}$ and $u_n \in \{-1, 0, 1\}$, and the cost function is:

$$J = \sum_{k=0}^{2} 2|x_k| + |u_k| + x_3^2$$

Use the dynamic programming algorithm to solve the finite-horizon optimal control problem that minimizes J. Show the different steps of the algorithms and present the results in a table including the cost-to-go and the optimal control at every stage.

[1] https://en.wikibooks.org/wiki/LaTeX, NYU provides access to Overleaf to all the community https://www.overleaf.com/edu/nyu

b) What is the sequence of control actions, states, and the optimal cost if $x_0 = 0$, if $x_0 = -2$, and if $x_0 = 2$?

c) Assume now that the constant term 1 in the previous dynamics can sometimes be 0 with probability 0.4. We can now write the dynamics as:

$$x_{n+1} = \begin{cases} -x_n + w_n + u_n & \text{if } -2 \leq (-x_n + w_n + u_n) \leq 2 \\ -2 & \text{otherwise} \end{cases}$$

where $w_n \in \{0, 1\}$ is a random variable with probability distribution $p(w_n = 0) = 0.4, p(w_n = 1) = 0.6$. The cost function to minimize becomes:

$$J_E = E\left[\sum_{k=0}^{T-1}(2|x_k| + |u_k|)\right] + E[x_T^2]$$

Use the dynamic programming algorithm to solve the finite-horizon optimal control problem that minimizes J_E. Show the different steps of the algorithms and present the results in a table including the cost-to-go and optimal control at every stage.

d) Compare the costs and optimal control from the deterministic and probabilistic models and explain where, in your opinion, the differences come from.

Solution (a): Deterministic Model

The dynamic programming solution for the deterministic model involves solving:

$$x_{n+1} = \begin{cases} -x_n + 1 + u_n & \text{if } -2 \leq -x_n + 1 + u_n \leq 2 \\ 2 & \text{if } -x_n + 1 + u_n > 2 \\ -2 & \text{otherwise} \end{cases}$$

with cost function:

$$J = \sum_{k=0}^{2}(2|x_k| + |u_k|) + x_3^2$$

Dynamic Programming Steps:

1. **Stage N (t = 3):** Terminal cost = x_3^2
2. **Stage N-1 (t = 2):** Minimize $2|x_2| + |u_2| + x_3^2$
3. **Stage N-2 (t = 1):** Minimize $2|x_1| + |u_1| + V_2(x_2)$
4. **Stage N-3 (t = 0):** Minimize $2|x_0| + |u_0| + V_1(x_1)$

Time	State	Optimal Control	Cost
0	-2	-1.0	10.0
0	-1	-1.0	5.0
0	0	-1.0	2.0
0	1	0.0	3.0
0	2	1.0	6.0
1	-2	0.0	8.0
1	-1	-1.0	5.0
1	0	-1.0	1.0
1	1	0.0	2.0
1	2	1.0	5.0
2	-2	0.0	4.0
2	-1	0.0	2.0
2	0	0.0	0.0
2	1	0.0	2.0
2	2	0.0	4.0

Table 1: Optimal Control and Cost-to-go Values for Different States and Time Steps

Solution (b): Sequence Analysis

For the three initial conditions:

Case 1: $x_0 = 0$

- States: $[0 \to 0 \to 0 \to 1]$
- Controls: $[-1 \to -1 \to 0]$
- Optimal Cost: 4.0

Case 2: $x_0 = -2$

- States: $[-2 \to 2 \to 0 \to 1]$
- Controls: $[-1 \to 1 \to 0]$
- Optimal Cost: 8.0

Case 3: $x_0 = 2$

- States: $[2 \to 0 \to 0 \to 1]$
- Controls: $[1 \to -1 \to 0]$
- Optimal Cost: 4.0

Solution (c): Probabilistic Model

The probabilistic model introduces uncertainty with:

$$x_{n+1} = \begin{cases} -x_n + w_n + u_n & \text{if } -2 \leq -x_n + w_n + u_n \leq 2 \\ 2 & \text{if } -x_n + w_n + u_n > 2 \\ -2 & \text{otherwise} \end{cases}$$

where $w_n \in \{0, 1\}$ with probabilities:

$$p(w_n = 0) = 0.4, \quad p(w_n = 1) = 0.6$$

The expected cost function:

$$J_E = E\left[\sum_{k=0}^{2}(2|x_k| + |u_k|) + x_3^2\right]$$

Time	State	Optimal Control	Expected Cost
0	-2	-1.0	10.600
0	-1	-1.0	5.952
0	0	0.0	2.952
0	1	0.0	4.880
0	2	1.0	7.880
1	-2	-1.0	9.200
1	-1	-1.0	4.680
1	0	0.0	1.680
1	1	0.0	3.800
1	2	1.0	6.800
2	-2	-1.0	7.800
2	-1	-1.0	3.600
2	0	0.0	0.600
2	1	0.0	2.400
2	2	1.0	5.400

Table 2: Optimal Control and Expected Cost-to-go Values for the Probabilistic Model

Solution (d): Comparison Analysis

Key Differences between Deterministic and Probabilistic Models:

1. **Cost Values:**
   - Deterministic costs are lower due to perfect knowledge
   - Probabilistic costs are higher due to uncertainty

2. **Control Strategies:**
    - Deterministic: More aggressive control actions
    - Probabilistic: More conservative control actions to account for uncertainty
3. **State Evolution:**
    - Deterministic: Perfectly predictable
    - Probabilistic: Accounts for random disturbances

The primary differences for this could be:

- Uncertainty in state transitions
- Need for robust control strategies
- Risk management in decision-making

Exercise 2 [60 points]
Consider the grid-world shown in Figure 1. In each (non grey) cell, it is possible to perform five actions: move up, down, left, right or do nothing as long as the resulting move stays inside the grid world. Grey cells are obstacles and are not allowed. We would like to find the optimal value function and optimal policy that minimize the following cost

$$\min \sum_{n}^{\infty} \alpha^n g_n(x_n)$$

with discount factor $\alpha = 0.99$ and where the instantaneous cost is defined as

$$g_n(x_n) = \begin{cases} -1 & \text{if } x_n \text{ is a violet cell} \\ 0 & \text{if } x_n \text{ is a white cell} \\ 1 & \text{if } x_n \text{ is a green cell} \\ 10 & \text{if } x_n \text{ is a red cell} \end{cases}$$

Figure 1: *Grid world*

Figure 1: Grid world

In a Jupyter notebook answer the following questions:

a) Implement the value iteration algorithm to solve the problem (initialize the value function to 0). How many iterations does it take to attain convergence? (we assume here that convergence happens when all the elements of the value function do not change more than 10^{-6} in a new iteration.

b) Implement the policy iteration algorithm to solve the problem (use the version that solves the linear equation $I - \alpha A)J_\mu = \bar{g}$). Start with an initial policy that does not move. How many iterations does it take to converge?

c) Compare the solutions and convergence/complexity of each algorithm to solve this problem.

(a) Value Iteration Implementation and Convergence

The value iteration algorithm was implemented to solve the grid world problem with a discount factor $\alpha = 0.99$. The algorithm iteratively updates the value function until convergence (defined as changes less than 10^{-6})
The implementation: - Convergence was achieved after approximately 1376 iterations.
- Initial value function initialized to zero for all states.
- The final value function shows clear patterns reflecting the cost structure:
* Lowest values near violet cells (reward -1)
* Highest values near red cells (cost 10)
* Intermediate values for green cells (cost 1)
* Gradient values showing optimal paths to violet cells.

(b) Policy Iteration Implementation and Convergence

- The policy iteration algorithm was implemented starting with a "do nothing". initial policy. Key observations: - Convergence achieved in 8 iterations.
- Each iteration involved:
* Policy evaluation: Solving linear equations.
* Policy improvement: Updating actions based on new value estimates.
- Final policy showed clear directional preferences:
* Movement towards violet cells when safe.
* Avoidance of red cells.
* Strategic navigation around obstacles.

(c) Comparative Analysis

Convergence Comparison:

- Value Iteration:
  - Required more iterations (1376)
  - Each iteration is computationally simpler.
  - Monotonic improvement in value estimates.

- Policy Iteration:
  - Fewer iterations (8).
  - More complex computation per iteration.
  - Faster overall convergence.

Computational Complexity:

- Value Iteration:
  - Memory efficient.
  - Easier to implement.

- Policy Iteration:
  - Higher memory requirements.
  - More complex implementation.

Solution Quality:

Both algorithms converged to the same optimal solution, demonstrating:
- Similar final value functions.
- Equivalent optimal policies.
- Consistent state-action preferences.

The choice between these algorithms depends on specific requirements:
- Value iteration should be preferred for memory-constrained systems.
- Policy iteration is better for problems requiring faster convergence.
- Both guarantee optimal solutions for this deterministic MDP.