
DEEP REINFORCEMENT LEARNING

HOMEWORK 4 | MODEL-BASED RL

Professor Mohammad Hossein Rohban

Sharif University of Technology
Spring 2026



Contents

1	Monte Carlo Tree Search	1
2	Cross-Entropy Method	3
3	Model Predictive Control	4

Important Notes

- **Grading Policy:** This homework is considered complete once you reach a total of **100 points**. Up to *5 additional bonus points* may be earned; however, any score above 100 will still be recorded as 100. Bonus points do *not* transfer or overflow to other assignments.
- **Homework Deadline:** The assignment is due on the posted deadline. Please make sure to submit your work on time.
- **Delay Usage Policy:** You may use *at most 3 days* of your allowed late-day budget for this homework. Late days will be automatically applied if you submit after the deadline.
- **Solution Release:** Official solutions will be released exactly **3 days after the deadline**. No submissions will be accepted once the solutions are published.
- **Cheating and Collaboration Policy:** You must write all solutions in your own words and produce all code independently. Discussing general ideas is allowed, but sharing written solutions, code, or using someone else's work (including from online sources, previous semesters, or AI tools without proper authorization) is strictly prohibited. Violations will result in academic misconduct procedures.

Advice

Please avoid asking LLMs for the answer. Sure, doing so might give you an immediate reward of 5 points, but in the long run, it won't help you become the next Rockstar of DRL. Remember the key lesson of this course: chasing immediate rewards is rarely the best strategy. Instead, struggle with the problems, read the papers, and truly understand the algorithms. The satisfaction of solving a challenging problem yourself is far greater than any grade. Good luck! [1]

Grading Breakdown

Task / Criteria	Points
MCTS	40 pts
Implementation	20 pts
Questions	20 pts
Cross-Entropy Methods	30 pts
Questions	15 pts
Model Predictive Control	30 pts
MPC as Feedback	10 pts
Planning Horizon	10 pts
CEM-Based Optimization	5 pts
Bonus	+5 pts
Clarity and Quality of Code	+2 pts
Clarity and Quality of Report	+3 pts
Total	100+5 pts

1 Monte Carlo Tree Search

In this section, we study **Monte Carlo Tree Search (MCTS)** and its neural variant with policy priors.

The goal is to understand how MCTS balances exploration and exploitation, and how neural priors can both improve and bias the search. We focus on two theoretical cases:

Implementation 1: MCTS Theory Questions

[20 Points]

1. Failure of UCT under misleading early estimates and delayed rewards.
2. Effect of neural policy priors in PUCT-based MCTS.

Question 1: When Can MCTS Become Confidently Wrong?

[10 Points]

Consider a root state s_0 with two possible actions, a_1 and a_2 . The true optimal action is a_2 , but early simulations make a_1 look better. MCTS uses

$$U(s_0, a) = Q(s_0, a) + c \sqrt{\frac{\ln N(s_0)}{N(s_0, a) + 1}}.$$

Assume

$$Q(s_0, a_1) = 0.65, \quad N(s_0, a_1) = n_1,$$

and

$$Q(s_0, a_2) = 0.35, \quad N(s_0, a_2) = n_2.$$

Answer the following:

- Derive the condition under which MCTS selects a_2 , i.e.,

$$U(s_0, a_2) > U(s_0, a_1).$$

- Explain how the exploration constant c affects recovery from the early wrong estimate.
- Suppose the good outcome under a_2 is only visible after depth d . Explain why MCTS may fail with a limited simulation budget.
- In what situation could naive depth search outperform MCTS?

Algorithm 1: UCT Selection Rule

- 1: **Input:** root state s_0 , statistics $Q(s, a)$, $N(s, a)$, exploration constant c
- 2: **for** each action $a \in \mathcal{A}(s_0)$ **do**
- 3: Compute

$$U(s_0, a) = Q(s_0, a) + c \sqrt{\frac{\ln N(s_0)}{N(s_0, a) + 1}}$$

- 4: **end for**
- 5: Select

$$a^* = \arg \max_a U(s_0, a)$$

Question 2: PUCT, Neural Priors, and Biased Exploration**[10 Points]**

In neural MCTS, the tree policy may use

$$a^* = \arg \max_a \left[Q(s, a) + c_{\text{puct}} P(s, a) \frac{\sqrt{N(s)}}{1 + N(s, a)} \right],$$

where $P(s, a)$ is the prior probability predicted by a policy network.

Suppose

$$P(s_0, a_1) = 0.80, \quad P(s_0, a_2) = 0.15, \quad P(s_0, a_3) = 0.05.$$

The true best action is a_3 , but the policy network gives it the smallest prior.

Answer the following:

- Explain why $P(s, a)$ is useful in large action spaces.
- Explain why a wrong prior can be harmful with a limited simulation budget.
- If all $Q(s_0, a)$ values are initially equal, compare the exploration bonus of a_1 and a_3 .
- Suppose

$$Q(s_0, a_1) = 0.4, \quad N(s_0, a_1) = 50,$$

and

$$Q(s_0, a_3) = 0.2, \quad N(s_0, a_3) = 1.$$

Write the inequality that determines whether a_3 will be selected next.

- Propose one modification that reduces over-reliance on the policy prior.

Algorithm 2: PUCT Selection Rule

- 1: **Input:** state s , statistics $Q(s, a)$, $N(s, a)$, prior $P(s, a)$, constant c_{puct}
- 2: **for** each action $a \in \mathcal{A}(s)$ **do**
- 3: Compute

$$\text{PUCT}(s, a) = Q(s, a) + c_{\text{puct}} P(s, a) \frac{\sqrt{N(s)}}{1 + N(s, a)}$$

- 4: **end for**
- 5: Select

$$a^* = \arg \max_a \text{PUCT}(s, a)$$

2 Cross-Entropy Method

In this section, we study the use of the **Cross-Entropy Method (CEM)** for trajectory optimization in model-based reinforcement learning.

The goal is to optimize an open-loop action sequence

$$a_{0:H-1} = (a_0, a_1, \dots, a_{H-1})$$

using a learned or known dynamics model

$$s_{t+1} = \hat{f}(s_t, a_t).$$

CEM searches over candidate action sequences, evaluates their predicted returns, keeps the elite samples, and refits the sampling distribution.

Question 1: CEM as Trajectory Optimization

[10 Points]

Assume that CEM samples candidate action sequences from

$$a_{0:H-1}^{(i)} \sim \mathcal{N}(\mu, \Sigma),$$

rolls them out using the model, and evaluates

$$J(a_{0:H-1}^{(i)}) = \sum_{t=0}^{H-1} r(s_t^{(i)}, a_t^{(i)}).$$

Answer the following:

- Explain how CEM uses elite samples to improve the sampling distribution.
- Why is CEM usually better than simple random shooting?
- Why does CEM not require gradients of the dynamics model or reward function?
- What information from one CEM iteration is carried to the next?

Question 2: Horizon Length and Dimensionality in CEM

[10 Points]

Suppose the action dimension is d_a and the planning horizon is H . Therefore, the CEM search variable has dimension

$$D = Hd_a.$$

Answer the following:

- Explain why increasing H can improve planning quality.
- Explain why increasing H also makes CEM harder to optimize.
- If $d_a = 6$ and $H = 40$, compute the dimension of the trajectory optimization problem.
- Why can a limited number of samples cause CEM to miss good trajectories in high-dimensional action spaces?

Question 3: Open-Loop Weakness of CEM-Based Planning

[10 Points]

CEM optimizes a full action sequence

$$(a_0, a_1, \dots, a_{H-1})$$

before executing the actions.

Answer the following:

- Why is this considered open-loop trajectory optimization?
- What can go wrong if the real next state differs from the predicted next state?
- Explain how using CEM inside MPC can reduce this problem.
- Why does MPC not completely remove the difficulty of optimizing long action sequences with CEM?

Algorithm 3: Cross-Entropy Method

1: **Input:** current state s_t , model $\hat{f}$, reward r , horizon H , samples N , elite fraction ρ , CEM iterations K

2: Initialize sampling distribution over action sequences:

$$a_{t:t+H-1} \sim \mathcal{N}(\mu_0, \Sigma_0)$$

3: **for** $k = 0, \dots, K - 1$ **do**

4: Sample N candidate action sequences:

$$a_{t:t+H-1}^{(1)}, \dots, a_{t:t+H-1}^{(N)} \sim \mathcal{N}(\mu_k, \Sigma_k)$$

5: **for** each candidate sequence $i = 1, \dots, N$ **do**

6: Roll out the sequence using the model:

$$\hat{s}_{t+j+1}^{(i)} = \hat{f}(\hat{s}_{t+j}^{(i)}, a_{t+j}^{(i)})$$

7: Compute predicted return:

$$J^{(i)} = \sum_{j=0}^{H-1} \gamma^j r(\hat{s}_{t+j}^{(i)}, a_{t+j}^{(i)})$$

8: **end for**

9: Select the top ρN elite sequences with the highest predicted returns.

10: Update μ_{k+1} and Σ_{k+1} using the mean and covariance of the elite sequences.

11: **end for**

12: Let $a_{t:t+H-1}^*$ be the best sequence found by CEM.

3 Model Predictive Control

In this section, we study **Model Predictive Control (MPC)** in the context of model-based reinforcement learning.

MPC repeatedly solves a finite-horizon planning problem, executes only the first action, observes the next

state, and replans. At each time step t , the planner solves

$$a_{t:t+H-1}^* = \arg \max_{a_{t:t+H-1}} \left[\sum_{k=0}^{H-1} \gamma^k r(\hat{s}_{t+k}, a_{t+k}) + \gamma^H \hat{V}(\hat{s}_{t+H}) \right],$$

subject to

$$\hat{s}_t = s_t, \quad \hat{s}_{t+k+1} = \hat{f}(\hat{s}_{t+k}, a_{t+k}).$$

Then only the first action a_t^* is executed.

Implementation 2: MPC Theory Questions

[20 Points]

1. MPC as receding-horizon feedback.
2. Planning horizon and terminal value.
3. CEM-based optimization inside MPC.
4. Learned model errors and replanning.

Question 1: MPC as Receding-Horizon Feedback

[10 Points]

MPC optimizes an action sequence

$$a_{t:t+H-1}^* = (a_t^*, a_{t+1}^*, \dots, a_{t+H-1}^*),$$

but executes only a_t^* .

Answer the following:

- Explain why the optimization problem solved at time t is open-loop.
- Explain why the overall behavior of MPC is still feedback-based.
- Give a simple example where executing the whole planned sequence would fail, but replanning after each step would help.
- Explain why MPC is not the same as directly learning a closed-loop policy $\pi(a \mid s)$.

Question 2: Planning Horizon and Terminal Value

[10 Points]

The MPC objective uses a finite horizon H and may include a terminal value function:

$$\sum_{k=0}^{H-1} \gamma^k r(\hat{s}_{t+k}, a_{t+k}) + \gamma^H \hat{V}(\hat{s}_{t+H}).$$

Answer the following:

- What problem can happen if the horizon H is too short?
- What problem can happen if the horizon H is too long?
- Explain how the terminal value $\hat{V}(\hat{s}_{t+H})$ can help when H is short.
- Give one case where a wrong terminal value function can make MPC choose a bad action.

Question 3: CEM-Based Optimization inside MPC

[5 Points]

Assume that MPC uses the Cross-Entropy Method (CEM) to optimize the action sequence. CEM

samples candidate sequences,

$$a_{t:t+H-1}^{(1)}, \dots, a_{t:t+H-1}^{(N)},$$

evaluates them using the model, keeps the elite samples, and refits the sampling distribution.

Answer the following:

- Explain why CEM is useful when the action space is continuous.
- If the action dimension is d_a , explain why the CEM search dimension is

$$D = Hd_a.$$

- Why does increasing H make CEM harder, even though it gives the planner more foresight?
- Explain why executing only the first action of the best CEM sequence can make the method more robust than executing the full sequence.

References

[1] Based on INF8250AE - Introduction to Reinforcement Learning. Fall 2025.