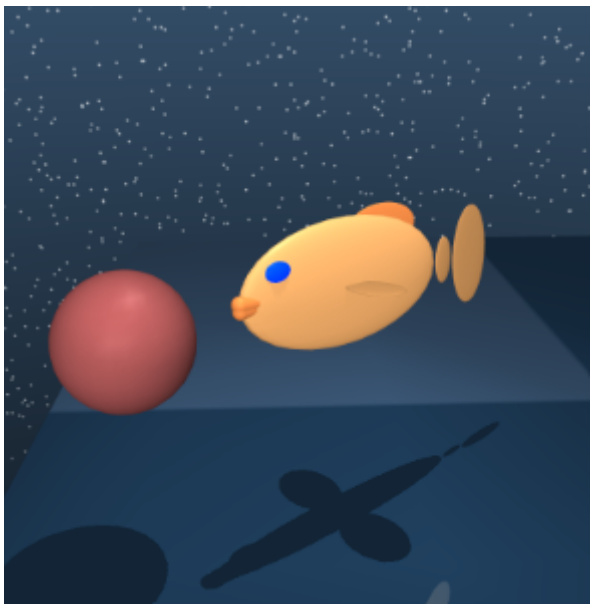
Embodied AI Homework 3——Model-based Planning && Learning

Part 0: Introduction

In this assignment, we aim to solve the task `fish-swim` from `deepmind control suite` with model-based approaches. The task involves guiding a fish to a target, taking into account the complexities of fluid dynamics. It is important to note that the reward function is affected by the distance from the fish's mouth to the target, which in turn is influenced by the initial positions of the fish. Consequently, to ensure robust evaluation, the evaluation process should contain at least **5** episodes.



Q0. Setting up the environment (5 pt)

Install PyTorch according to https://pytorch.org. Run `pip install -r requirements.txt` to install other dependencies. After setting up the environment, run `bash runs/q0.sh` to evaluate the random policy. Report the performance.

Part 1. Model Predictive Control Using the Ground-Truth Model

In this section, our attention is centered on the planning aspect under the premise that the dynamics model, denoted as $f: s_{t+1}, r_t = f(s_t, a_t)$, is fully known. This assumption allows us to directly engage with the optimization problem at hand:

$$\underset{a_T, a_{T+1}, \ldots, a_{T+H-1}}{argmax} \quad E[\sum_{t=0}^{H-1} \gamma^t r(s_{T+t}, a_{T+t})]$$

$$s_{t+1}, r(s_t, a_t) = f(s_t, a_t)$$

To address this optimization challenge, we will employ a strategy that involves sampling a variety of action sequences. From this pool of possibilities, we will then select the most suitable sequence based on predefined criteria.

Q1. Random shooting (10 pts)

A straightforward approach to solving the optimization problem is to implement a random shooting method. This involves sampling K random action sequences and selecting the best one among them. Complete the code in `ModelBasedAgent.plan` and `ModelBasedAgent.estimate_value`. Then run `bash runs/q1.sh` to evaluate the policy. Report your results.

Q2. Cross Entropy Method (15 pts)

An advanced alternative to the random shooting method is the cross-entropy method(check the EAI course slides if you are unfamiliar with it). Complete the code in `ModelBasedAgent.plan`. Then run `bash runs/q2.sh` to evaluate the policy. Experiment with different sampling hyperparameters, and detail your observations and any significant findings in your report.

Q3. MPPI (10pts)

MPPI(https://arxiv.org/pdf/1509.01149) is another MPC algorithm like CEM. represents another MPC algorithm similar to CEM, with a key distinction in how the sampling distribution is updated. Specifically, the update incorporates a weighted mean of the elite actions, where the weights are proportional to $exp(\tau \cdot (s - s_{max}))$ where s being the estimated value of action sequence. Complete the code in `ModelBasedAgent.plan`. Then run `bash runs/q3.sh` to evaluate the policy. Experiment with different sampling hyperparameters(*optional*), and detail your observations and any significant findings in your report.

Part 2. Learning the model

In practice, the ground-truth model may either be unknown or too computationally intensive for large-scale simulation. In this part, we will learn an approximate world model $f_\theta : \hat{s}_{t+1}, \hat{r}_t = f_\theta(s_t, a_t)$, using data collected online.

Q4.1 Maintain the state difference statistics. (10 pts)

It is often observed that s_{t+1} and s_t are similar, with the difference between these states revealing insights into the dynamics of the environment. Therefore, we predict the state difference $\delta s = s_{t+1} - s_t$ instead of the next state s_{t+1}. We also maintain the moving mean and std of the state difference to avoid numerical issues. Complete `update_statistics` and `next` methods of `LearnedModel`.

Q4.2 Compute the prediction loss (code & report, 15 pts)

The effectiveness of the learned world model is contingent upon minimizing the prediction loss, which is $MSE(\hat{s}, s) + MSE(\hat{r}, r)$. Complete the code in `ModelBasedAgent.update`. The hints in the comments may be helpful.

Part 3. TD-MPC

In this part, we will leverage some model-free approaches to improve our MPC algorithm. Specifically, we consider two improvements:

- Long planning horizon requires expensive computation, while short planning horizon guarantees only the local optimization. Therefore, we introduce a value function $V(s)$ that estimate the accumulated return beyond the terminal state of planning.

- MPC with purely random initial actions may be inefficient. Therefore, we incorporate a learned deterministic policy enhanced with Gaussian noise. This policy proposes initial actions for subsequent iterations, streamlining the action selection process.

We implement these by parametrizing a Q network $Q_\theta(s, a)$ and a policy network $\pi_\phi(s)$.

The learning objective of the Q network is the n-step TD-target (read https://medium.com/zero-equals-false/n-step-td-method-157d3875b9cb for reference)
$MSE[Q_\theta(s_T, a_T), \sum_{t=0}^{n-1} \gamma^t \hat{r}(s_{T+t}, a_{T+t}) + \gamma^n Q_\theta(s_{T+n}, \pi_\phi(s_{T+n}))]$.

The learning objective of the policy network is the deterministic policy gradient $-\nabla_\phi Q(s, \pi_\phi(s))$.

Q5. TD-MPC

1. Complete the `LearnedModel.V` method. Express the `V` function with the `Q` function and the deterministic policy `pi`. Add final state value estimation in `ModelBasedAgent.estimate_value`. (10pt)
2. Complete the `LearnedModel._td_target` method. (5pt)
3. Compute the `policy_loss` and `value_loss` in `ModelBasedAgent.update_pi` and `ModelBasedAgent.update` method. (10pts)
4. Add actions sampled from the learned policy to initial actions in `ModelBasedAgent.plan`. (5pts)
5. Present your results in the report. The final performance is expected to be over 650. (10pts)

Hints:

- Always pass `cfg.min_std` to the policy, so that the agent can explore.

Part 4. Network Ensemble

Network ensemble is a robust technique in model-based reinforcement learning that aims to enhance model reliability and performance. This approach involves several key strategies:

- Averaging the outputs of differently initialized dynamics models to reduce model bias. (e.g., https://arxiv.org/pdf/1805.12114)
- Estimating the model epistemic uncertainty from ensemble divergence. (e.g., https://arxiv.org/pdf/1807.01675)
- Reducing bias in TD-target computation by ensembling Q networks. (e.g., https://arxiv.org/abs/2101.05982. This technique is used in TD-MPC2)

Q6. Network Ensemble. (25pts)

Your task is to implement any network ensemble trick that fits within the existing codebase, analyze and report your findings in your report. You can use the reserved `d_ensemble` and `q_ensemble` keys in the configuration.

Submission Guidelines

This is the last homework of this course. The total score for these exercises is 130, allowing you to choose certain problems to complete. The final score is capped at 100. Submit your code and report in a `.zip` file on Web Learning by **Jan 11, 23:59, 2025**. **Each part of the codes takes a few hours to run on cuda. Therefore, we highly recommend starting early!** Contact TA Can Chang if you have any questions.