MOPP - Model-Based Offline Planning with Trajectory Pruning

Master’s Degree in Artificial Intelligence and Robotics

Reinforcement Learning Course

Giuseppina Iannotti 1938436

Maria Emilia Russo 1966203

Academic Year 2024/25

SAPIENZA UNIVERSITÀ DI ROMA

Table Of Contents

› Introduction

› Data

› Proposed Method

› Experiments

› Conclusions and Future Works

› References

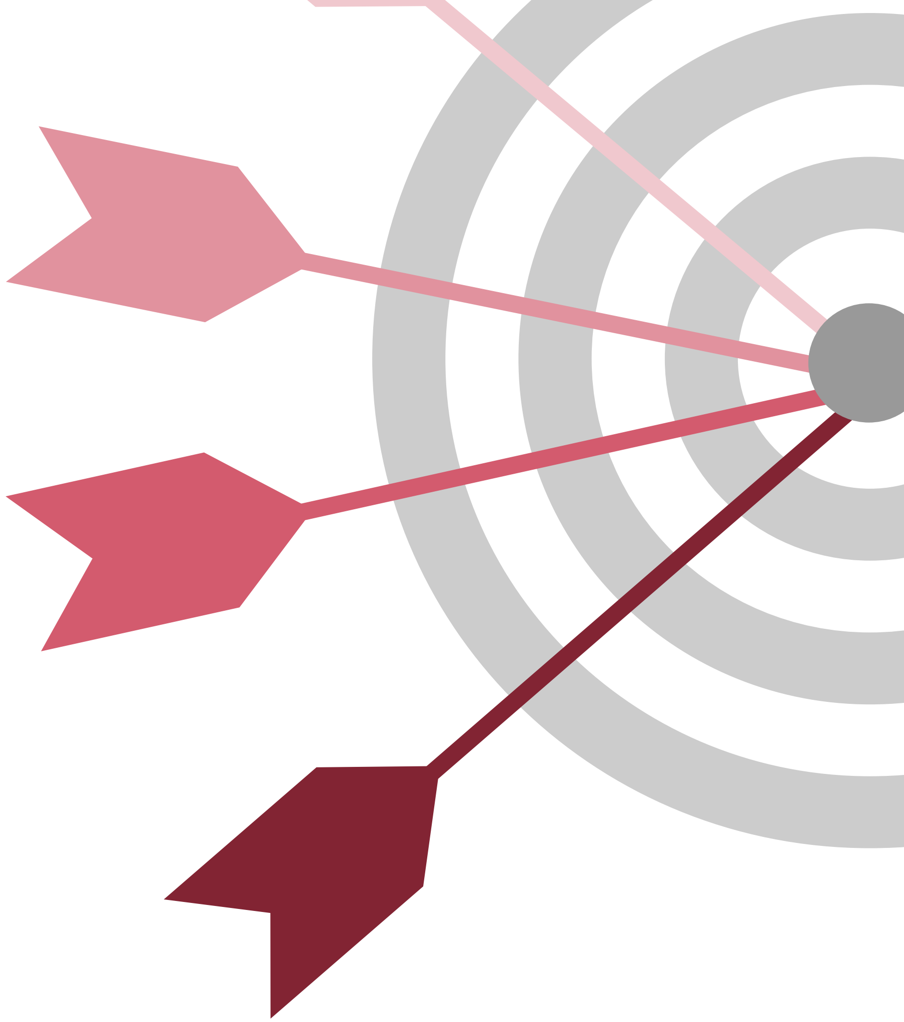
Introduction

Offline RL is a branch of reinforcement learning where the policy is learned from a static, pre-collected dataset of environment interactions, without additional interactions with the environment.

A key **challenge** in Offline RL is keeping policy learning close to the data distribution. Taking actions that deviate from the dataset (**OOD samples**) can lead to inaccurate predictions by the learned models, causing suboptimal or unsafe decisions.

To address this, **Model-Based Planning** builds a dynamics model of the environment, which predicts how the environment will respond to actions.

Work Focus



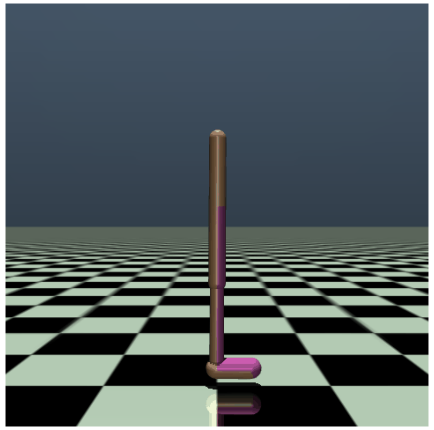
Our work focuses on implementing a **Model-Based Offline Planning** algorithm. Specifically, it conducts trajectory rollouts guided by a behavior policy learned directly from data. The algorithm identifies and prunes problematic trajectories to minimize out-of-distribution (OOD) actions, improving decision-making and optimizing outcomes.

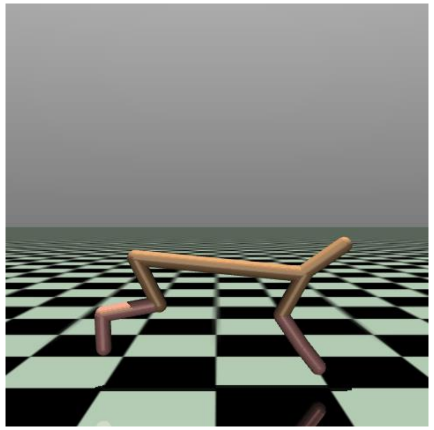
Table Of Contents

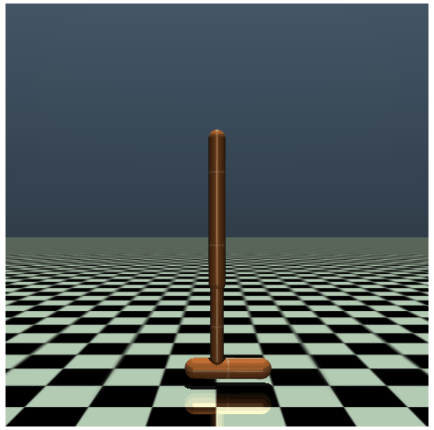
- Introduction
- Data
- Proposed Method
- Experiments
- Conclusions and Future Works
- References

Dataset: Mujoco

- **Mujoco**: **Multi-Joint dynamics with Contact**. It provides realistic physics simulations for contact dynamics and balances physical accuracy and computational efficiency.
- The environments chosen are: **Half Cheetah**, **Walker2D** and **Hopper**. They are all stochastic in terms of their initial state, with a Gaussian noise added to a fixed initial state in order to add stochasticity.


a) Walker2D

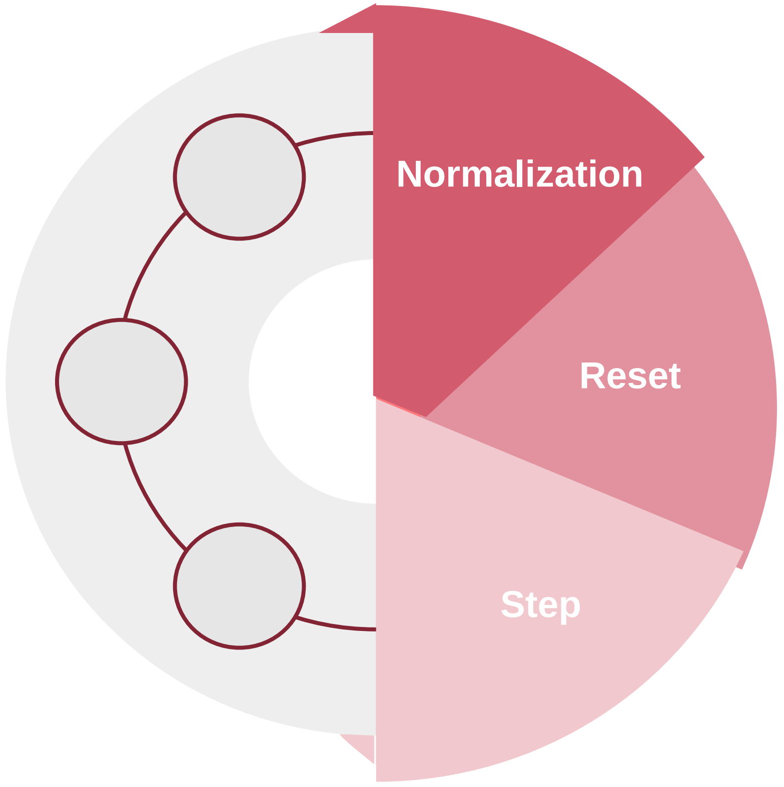

b) Half Cheetah


c) Hopper

Environment set-up

Data Loading & Preprocessing

Loads observations, actions, rewards, next states, and done flags from the dataset.



Normalization (optional)

- **State Normalization**: adjusts observations and next states by shifting and scaling based on dataset statistics.
- **Reward Normalization**: rescales rewards to [0,1]

Reset

The reset method is called at the start of each episode to initialize the environment.

Step

It is called repeatedly during an episode to sample transitions until the done flag is true or the dataset is exhausted.

Table Of Contents

› Introduction

› Data

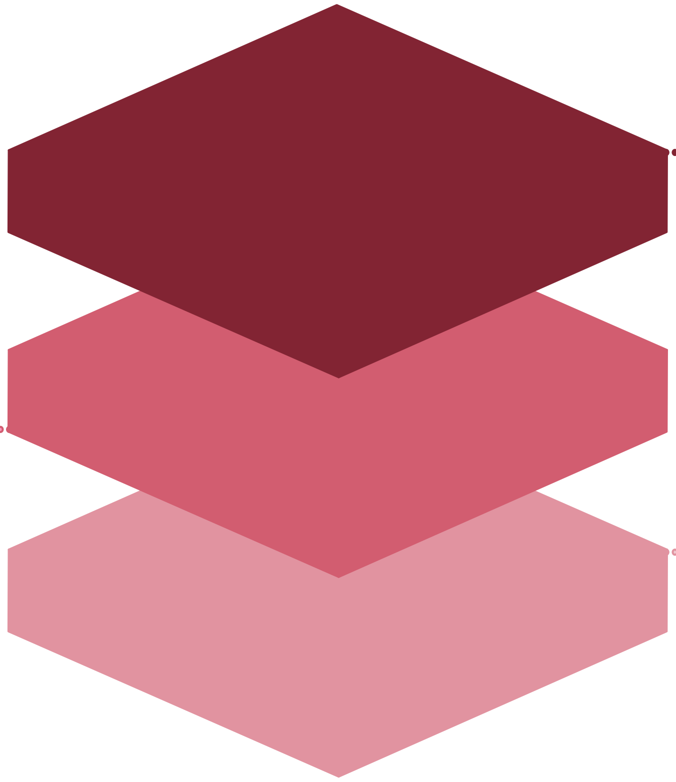
› Proposed Method

› Experiments

› Conclusions and Future Works

› References

Overview



Q-Network
Guides the policy to estimate the value of various state-action pairs

Dynamics Model
It learns the environmet's behavior by predicting its response to different actions.

MOPP
Planning algorithm that leverages trajectories generated by the dynamics model to optimize action sequences and improve decision-making.

ADM: Autoregressive Dynamic Model

ADM models the **joint probability** of outputs by breaking it down into conditional probabilities for each dimension of the output:

$$p(\mathbf{o}) = \prod_i^I p(o_i \mid \mathbf{x}, \mathbf{o}_{[<i]})$$

with:
- $o_i =$ value of the output at i-th dimension
- $o_{<i} =$ all previous output dimensions not including i

It has 2 main **purposes**:

- **Probabilistic Dynamics Model**: $f_m(s_t, a_t) = (r_t, s_{t+1})$
- **Behavior Policy**: $f_b(s_t) = a_t$

ADM: Network and Ensemble

ADM Network Design

❑ Input Processing

The network processes two types of inputs:

- **State**: used to predict the behavior policy, which outputs actions as mean and variance.
- **State-Action Pair** : used to model dynamics, predicting the next state and reward.

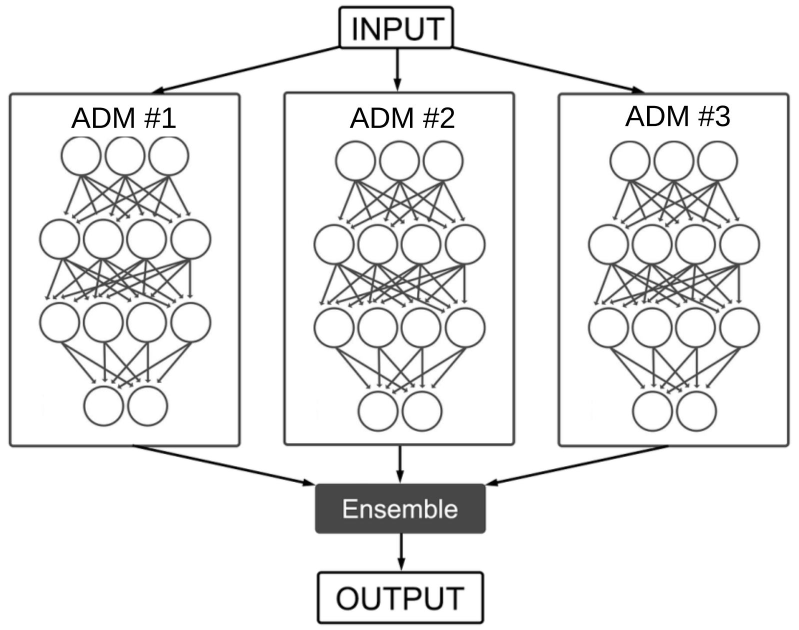
❑ Output Modeling

For each output dimension, separate MLPs predict:

- **Mean**: central tendency of the distribution
- **Standard Deviation (Std)**: captures uncertainty, enabling probabilistic sampling and log-likelihood computations.

ADM Ensemble

MOPP uses an **ensemble of K ADM models**, each designed with a unique output ordering leading to diverse probabilistic outcomes. Through the ensemble, the model can identify and mitigate high-uncertainty trajectories.



Q-Network

Understanding the expected return of a given state-action pair under the behavior policy is crucial. This is achieved by learning the Q-function, which estimates the cumulative future rewards starting from state when taking a specific action and following the behavior policy thereafter.

Learn Q-Function with FQE

This approach iteratively trains the Q-Function by minimizing the difference between the predicted Q-value and the targte value.

$$Q_b^k(s_i, a_i) = \arg\min_{f \in F} \frac{1}{N} \sum_{i=1}^{N} [f(s_i, a_i) - y_i]^2$$

with:

$$y_i = r_i + \gamma' Q_b^{k-1}(s_{i+1}, a_{i+1})$$

$$(s_i, a_i, s_{i+1}, a_{i+1}) \sim \mathcal{B}$$

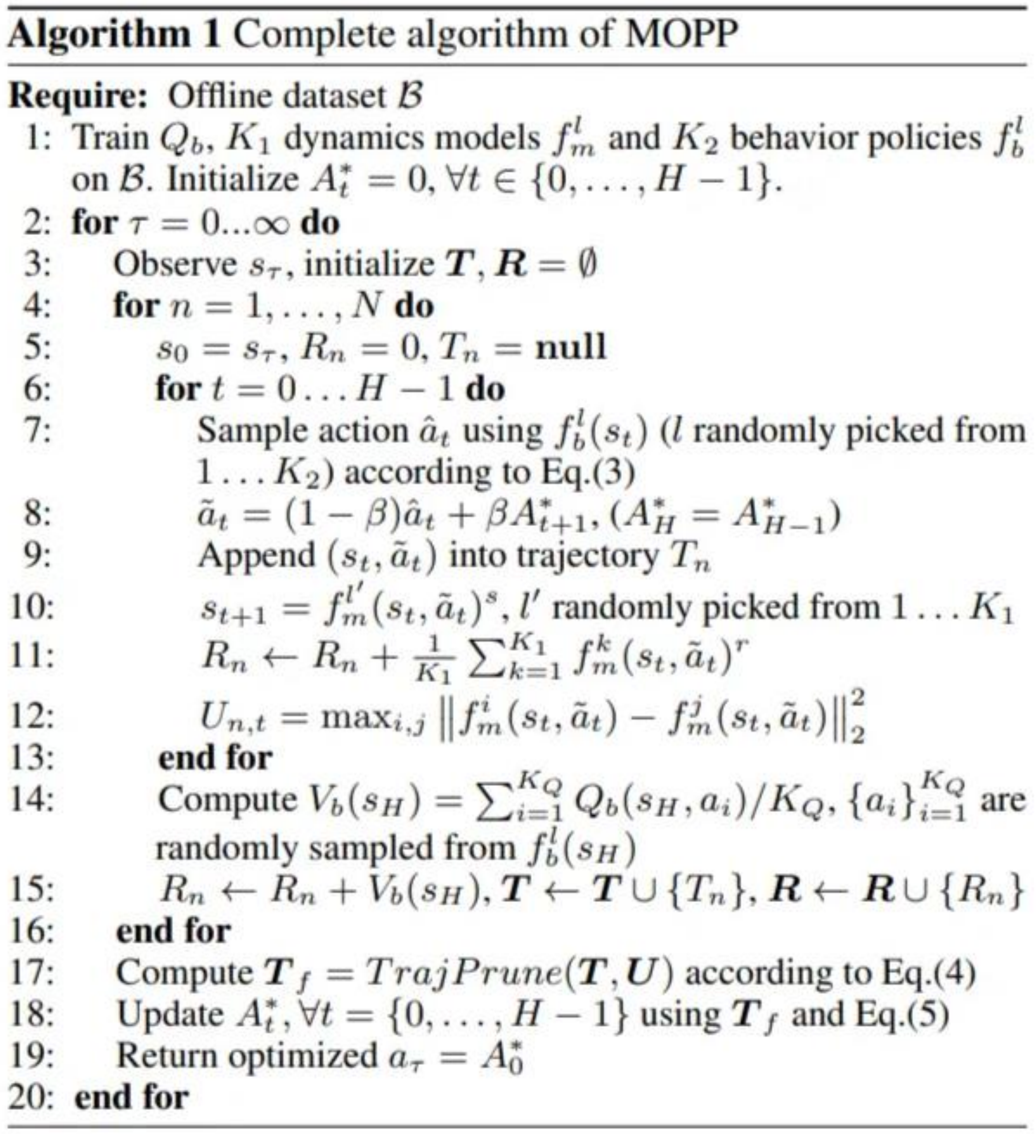
Evaluating Value Function

Once the Q-function is learned, it's used to evaluate the Value function, which represents the expected value of a state under the behavior policy.

$$V_b(s_t) = E_{a \sim \pi_b} Q(s_t, a)$$

MOPP: Algorithm

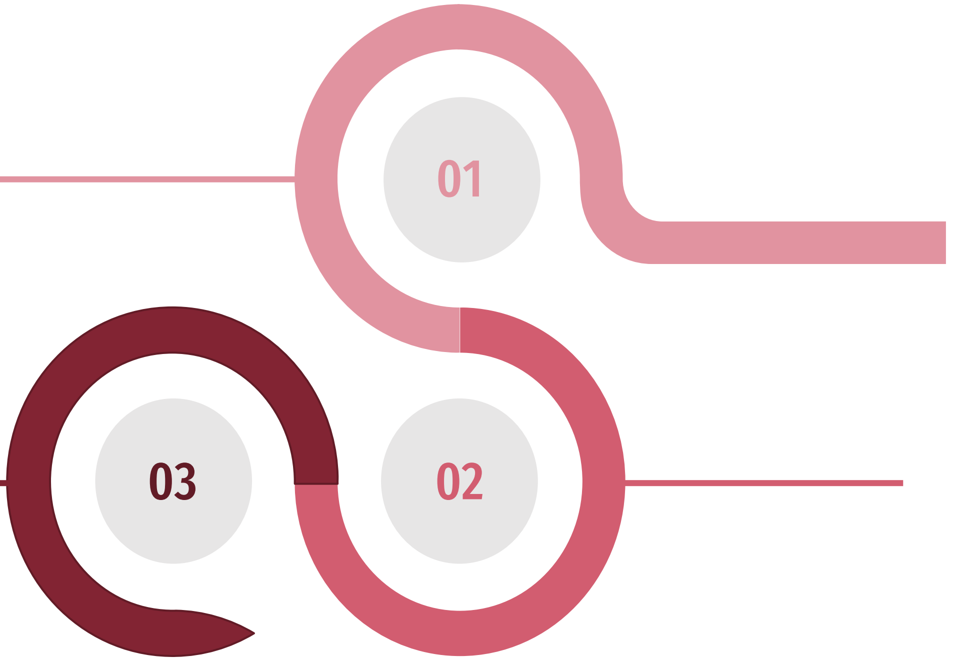
- MOPP is a model-based offline planning framework designed to optimize actions for high-reward trajectories in offline reinforcement learning
- It enables diverse sampling from the behavior policy with boosted variance and performs max-Q operations on sampled actions.
- It leverages an expressive autoregressive dynamics model for learning system dynamics and optimizing behavior
- By integrating a value function at the terminal state, MOPP extends the planning horizon within finite-horizon Model Predictive Control (MPC), optimizing actions over a fixed horizon H



Algorithm 1 Complete algorithm of MOPP

```
Require: Offline dataset $\mathcal{B}$
 1: Train $Q_b$, $K_1$ dynamics models $f_m^l$ and $K_2$ behavior policies $f_b^l$
    on $\mathcal{B}$. Initialize $A_t^* = 0, \forall t \in \{0, \dots, H-1\}$.
 2: for $\tau = 0 \dots \infty$ do
 3:   Observe $s_\tau$, initialize $\boldsymbol{T}, \boldsymbol{R} = \emptyset$
 4:   for $n = 1, \dots, N$ do
 5:     $s_0 = s_\tau$, $R_n = 0$, $T_n =$ null
 6:     for $t = 0 \dots H-1$ do
 7:       Sample action $\hat{a}_t$ using $f_b^l(s_t)$ ($l$ randomly picked from
          $1 \dots K_2$) according to Eq.(3)
 8:       $\tilde{a}_t = (1-\beta)\hat{a}_t + \beta A_{t+1}^*$, $(A_H^* = A_{H-1}^*)$
 9:       Append $(s_t, \tilde{a}_t)$ into trajectory $T_n$
10:       $s_{t+1} = f_m^{l'}(s_t, \tilde{a}_t)^s$, $l'$ randomly picked from $1 \dots K_1$
11:       $R_n \leftarrow R_n + \frac{1}{K_1}\sum_{k=1}^{K_1} f_m^k(s_t, \tilde{a}_t)^r$
12:       $U_{n,t} = \max_{i,j} \left\| f_m^i(s_t, \tilde{a}_t) - f_m^j(s_t, \tilde{a}_t) \right\|_2^2$
13:     end for
14:     Compute $V_b(s_H) = \sum_{i=1}^{K_Q} Q_b(s_H, a_i)/K_Q$, $\{a_i\}_{i=1}^{K_Q}$ are
        randomly sampled from $f_b^l(s_H)$
15:     $R_n \leftarrow R_n + V_b(s_H)$, $\boldsymbol{T} \leftarrow \boldsymbol{T} \cup \{T_n\}$, $\boldsymbol{R} \leftarrow \boldsymbol{R} \cup \{R_n\}$
16:   end for
17:   Compute $\boldsymbol{T}_f = TrajPrune(\boldsymbol{T}, \boldsymbol{U})$ according to Eq.(4)
18:   Update $A_t^*, \forall t = \{0, \dots, H-1\}$ using $\boldsymbol{T}_f$ and Eq.(5)
19:   Return optimized $a_\tau = A_0^*$
20: end for
```

MOPP: Trajectory Rollout



Action Sampling

Actions sampled from a Gaussian distribution, scaling uncertainty to enhance exploration

Max-Q Selection

Actions with the highest Q-value are selected for long-term reward optimization.

Trajectory Mixing

Actions are blended with previous ones using a coefficient to balance diversity and prior knowledge.

MOPP: Trajectory Pruning and Optimization

Trajectory Pruning

Goal: Focus on high-value trajectories by removing low-reward or uncertain ones.

1. Compute state-action **uncertainty matrix** for all trajectories, using dynamics model ensemble disagreement

$$\mathrm{disc}(s,a) = \max_{i,j} \left\| f_m^i(s,a) - f_m^j(s,a) \right\|_2^2$$

2. **Retain** trajectories with **low** uncertainty or supplement with the least uncertain ones.

$$\mathrm{TrajPrune(T,\ U)} := \begin{cases} T_p, & \text{if } |T_p| \geq N_m, \\ T_p \cup \mathrm{sort}(T - T_p, U)[0 : N_m - |T_p|], & \text{if } |T_p| < N_m. \end{cases}$$

Trajectory Optimization

Goal: being an extended version of MPPI, it generates and evaluates candidate action sequences to identify optimal actions

1. Calculate **cumulative returns** for each trajectory

$$R_n = \sum_{t=0}^{H-1} \gamma^t r(s_t^n, a_t^n) + \gamma^H V_b(s_H^n)$$

2. **Re-weight** actions based on exponentiated returns

$$A_t^* = \frac{\sum_{n=1}^{|T_f|} \exp(\kappa R_n) a_t^n}{\sum_{n=1}^{|T_f|} \exp(\kappa R_n)}, \quad \forall t = \{0, \ldots, H-1\}$$

Table Of Contents

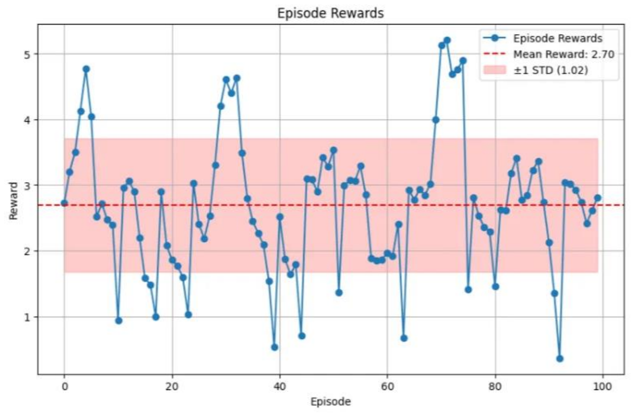
› Introduction
› Data
› Proposed Method
› Experiments
› Conclusions and Future Works
› References

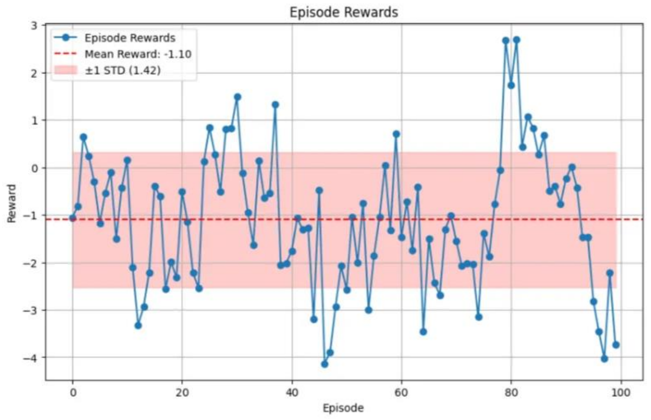
Experiments ●○

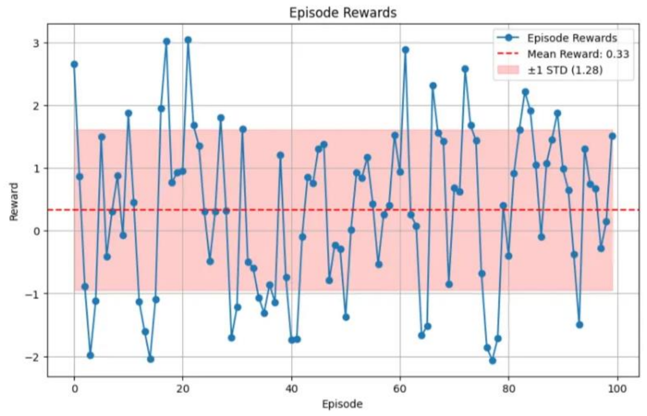
Results

Hopper | **Half Cheetah** | **Walker2D**

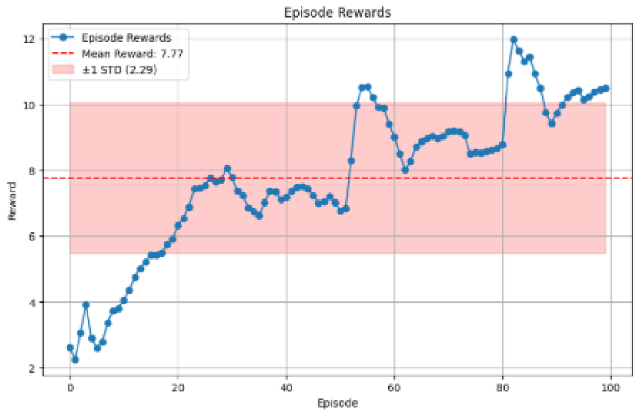
Random

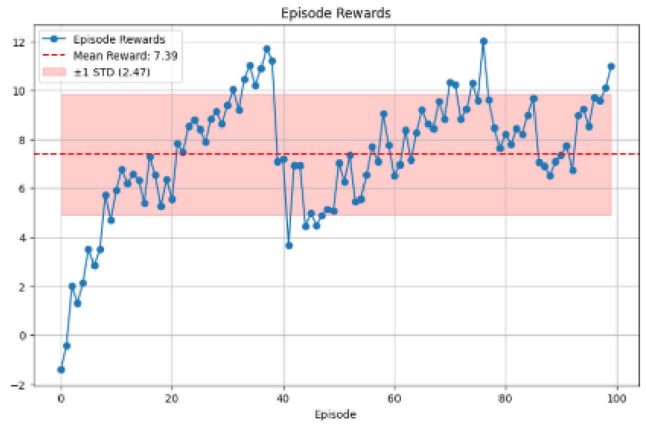


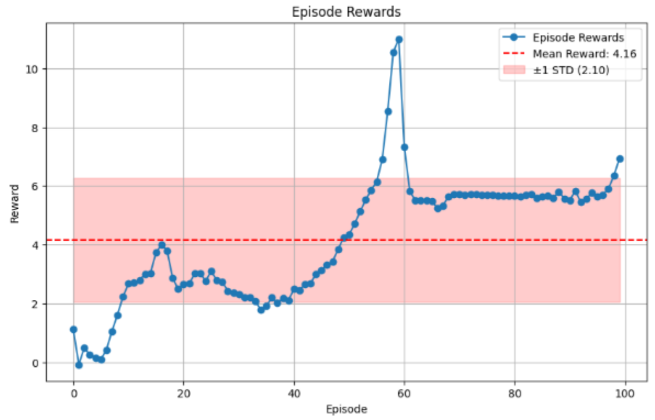




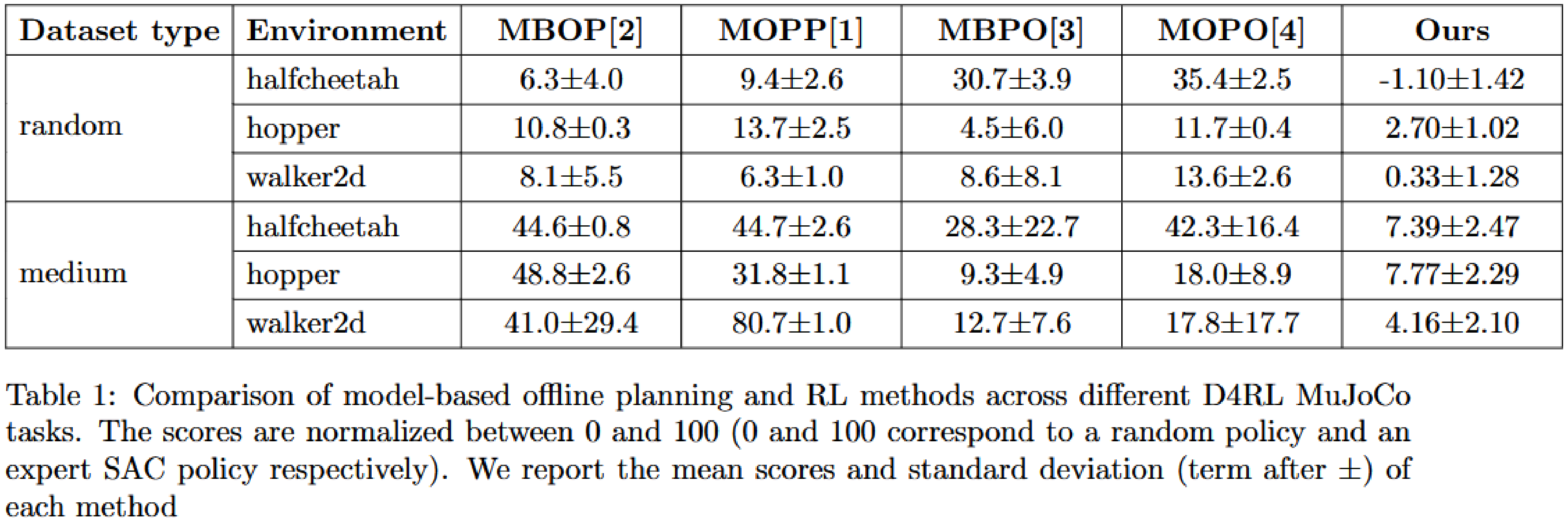
Medium







SOTA Comparison



Dataset type	Environment	MBOP[2]	MOPP[1]	MBPO[3]	MOPO[4]	Ours
random	halfcheetah	6.3±4.0	9.4±2.6	30.7±3.9	35.4±2.5	-1.10±1.42
	hopper	10.8±0.3	13.7±2.5	4.5±6.0	11.7±0.4	2.70±1.02
	walker2d	8.1±5.5	6.3±1.0	8.6±8.1	13.6±2.6	0.33±1.28
medium	halfcheetah	44.6±0.8	44.7±2.6	28.3±22.7	42.3±16.4	7.39±2.47
	hopper	48.8±2.6	31.8±1.1	9.3±4.9	18.0±8.9	7.77±2.29
	walker2d	41.0±29.4	80.7±1.0	12.7±7.6	17.8±17.7	4.16±2.10

Table 1: Comparison of model-based offline planning and RL methods across different D4RL MuJoCo tasks. The scores are normalized between 0 and 100 (0 and 100 correspond to a random policy and an expert SAC policy respectively). We report the mean scores and standard deviation (term after ±) of each method

Notice: due to lack of computational power, experiments are conducted on a reduced part of the dataset (100.000 samples instead of 1M). Moreover, components are trained for 1000 epochs rather than 500.000, as in the SOTA papers.

Table Of Contents

- Introduction
- Data
- Proposed Method
- Experiments
- Conclusions and Future Works
- References

Conclusions and Future Works

RESULTS
The method showed promising performance, especially considering the limited training time.

ROLLOUT IMRPOVEMENT
Future works can explore better rollout strategies to improve trajectory quality and diversity.

TESTING ON NEW DATASETS
Expanding the evaluation to other datasets like Androit or different MuJoCo types can help assess the method's versatility and effectiveness.

Table Of Contents

› Introduction
› Data
› Proposed Method
› Experiments
› Conclusions and Future Work Directions
› References

References

[1] Zhan, Xianyuan, Xiangyu Zhu, and Haoran Xu. "Model-based offline planning with trajectory pruning”. arXiv preprint arXiv:2105.07351 (2021)

[2] Argenson, Arthur, and Gabriel Dulac-Arnold. "Model-based offline planning.” arXiv preprint arXiv:2008.05556 (2020)

[3] Janner, Michael, et al. "When to trust your model: Model-based policy optimization.” Advances in neural information processing systems 32 (2019)

[4] Yu, Tianhe, et al. "Mopo: Model-based offline policy optimization.” Advances in Neural Information Processing Systems 33 (2020): 14129-14142

[5] Le,Hoang, Cameron Voloshin, and Yisong Yue. “Batch policy learning under constraints.” International Conference on Machine Learning. PMLR, 2019

MOPP - Model-Based Offline Planning with Trajectory Pruning

Thank you for listening!
Any questions?