OCEAN-MBRL: Offline Conservative Exploration for Model-Based Offline Reinforcement Learning

Fan Wu[1,2], Rui Zhang [3], Qi Yi[4], Yunkai Gao[4], Jiaming Guo[3], Shaohui Peng[1], Siming Lan[4], Husheng Han[2, 3], Yansong Pan[2], Kaizhao Yuan[2], Pengwei Jin[2, 3], Ruizhi Chen[1], Yunji Chen[2, 3], Ling Li[1, 2, *]

[1] Intelligent Software Research Center, Institute of Software, CAS, Beijing, China
[2] University of Chinese Academy of Sciences, UCAS, Beijing, China
[3] SKL of Processors, Institute of Computing Technology, CAS, Beijing, China
[4] University of Science and Technology of China, USTC, Hefei, China
wufan2020@iscas.ac.cn

Abstract

Model-based offline reinforcement learning (RL) algorithms have emerged as a promising paradigm for offline RL. These algorithms usually learn a dynamics model from a static dataset of transitions, use the model to generate synthetic trajectories, and perform conservative policy optimization within these trajectories. However, our observations indicate that policy optimization methods used in these model-based offline RL algorithms are not effective at exploring the learned model and induce biased exploration, which ultimately impairs the performance of the algorithm. To address this issue, we propose Offline Conservative ExplorAtioN (OCEAN), a novel rollout approach to model-based offline RL. In our method, we incorporate additional exploration techniques and introduce three conservative constraints based on uncertainty estimation to mitigate the potential impact of significant dynamic errors resulting from exploratory transitions. Our work is a plug-in method and can be combined with classical model-based RL algorithms, such as MOPO, COMBO, and RAMBO. Experiment results of our method on the D4RL MuJoCo benchmark show that OCEAN significantly improves the performance of existing algorithms.

Introduction

Reinforcement learning (RL) (Sutton and Barto 2018) is a machine learning method that enables agents to learn optimal decision-making policy in complex environments through interaction with their surroundings. While current RL algorithms have made significant advances in fields such as robot control (Kalashnikov et al. 2018; Quillen et al. 2018), autonomous driving (Yurtsever et al. 2020; Kiran et al. 2021), and game intelligence (Silver et al. 2016), the associated risks and costs of agent-environment interaction cannot be overlooked. In this regard, offline RL (Lange, Gabel, and Riedmiller 2012), which learns strategies from existing datasets, presents a promising solution, making it a critical component in implementing RL algorithms in the real world.

Restricted datasets are the main reason for the performance limitations of offline RL algorithms (Fujimoto, Meger, and Precup 2019). The current research mainly focuses on how to better utilize existing data to update policies and achieve better performance (Kumar et al. 2019; Wu, Tucker, and Nachum 2019; An et al. 2021; Wu et al. 2021). Simultaneously, model-based offline RL algorithms are a natural fit for addressing the issue of limited data in offline settings by modeling the dataset and learning the dynamic model of the environment (Yu et al. 2020, 2021; Rigter, Lacerda, and Hawes 2022), which is used for rolling out trajectories and updating policy. However, the cumulative error incurred during the interaction with the dynamic model renders the rollout of a complete trajectory detrimental to policy learning. Therefore, most model-based offline RL algorithms adopt the k-step rollout method of MBPO (Janner et al. 2019).

Our work highlights the fact that previous studies have not explicitly investigated the exploration mechanism in the k-step rollout method, which is particularly crucial in model-based offline RL. Existing approaches have mainly relied on simplistic mechanisms such as maximum entropy exploration from SAC (Haarnoja et al. 2018), along with a rollout stop mechanism based on uncertainty. However, we have observed that such a mechanism introduces bias, as depicted

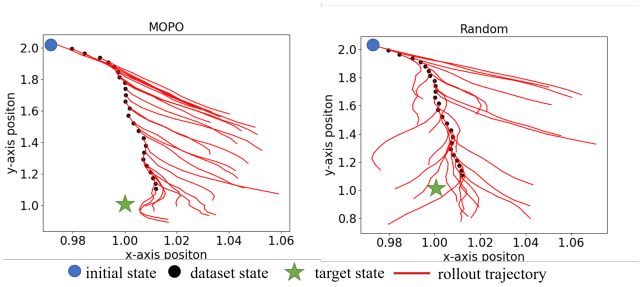


Figure 1: Comparison of random policy against maximum entropy policy from MOPO (Yu et al. 2020). We train MOPO for 2M gradient steps in the D4RL (Fu et al. 2020) Maze2d-umaze-v1 task and rolled out trajectories from states in the task dataset. The MOPO policy in the left figure can only roll out on the right side of the existing data, while a random policy can better explore the entire state space.

*Corresponding author.

in Figure 1. It is evident that the maximum entropy exploration employed by MOPO is biased and fails to explore the state space effectively. Biased exploration results in a biased training data distribution, which in turn contributes to a degradation in the overall performance of the algorithm and introduces instability. As a result, by ensuring a more comprehensive exploration during the rollout phase, we aim to enhance the overall performance and stability of the learned policy.

In this work, we present Offline Conservative ExplorAtioN (OCEAN), introducing additional exploration during trajectory rollouts to solve the problem of biased exploration in model-based offline RL. However, in the case of offline RL, direct exploration during the rollout stage is bound to be influenced by model errors, which can ultimately undermine the effectiveness of policy training. To minimize the impact of exploration-induced model error, we restrict the exploration strategy to ensure conservative exploration. Specifically, we employ three constraints to limit excessive exploration by the policy. Firstly, we define the uncertainty of the current state using uncertainties in the transition model to ensure that the policy explores states with low uncertainty. Secondly, in states with low uncertainty, our approach selects a more conservative action from the exploration strategy to generate the transition. Lastly, we limit the rollout length and automatically adjust it. Our method is an insertable approach that can be combined with the current SOTA model-based offline algorithms. We incorporate OCEAN with three model-based offline RL algorithms, MOPO, COMBO, and RAMBO and empirical results show that OCEAN markedly boosts their performance across a variety of benchmark tasks. Furthermore, our ablation experiments provide additional evidence of the efficacy of our method.

To the best of our knowledge, our research is the first to specifically tackle the exploration problem in model-based offline RL settings. Our work makes three main contributions, which are outlined as follows:

- We address a problem of biased exploration in model-based offline RL settings.
- We propose a novel rollout method that effectively addresses the exploration challenge.
- We conduct extensive experimentation and validate the effectiveness of our proposed approach, OCEAN, on the D4RL MuJoCo benchmark.

Related Work

Offline RL aims to learn policies from a logged static dataset. Starting from the vanilla model-free multistep actor-critic method, there exist several modifications that can be incorporated into an offline RL algorithm to improve its performance (Prudencio, Maximo, and Colombini 2023; Levine et al. 2020). Policy constraint methods modify the objective of unconstrained policy improvement to maximize a constrained objective (Fujimoto, Meger, and Precup 2019; Kumar et al. 2019; Wu, Tucker, and Nachum 2019). Regularization methods penalize learned value functions to produce more conservative estimates (Kumar et al. 2020; Nachum et al. 2019). One-step methods avoid iterative policy evaluation, instead performing a single step of evaluation followed by a single policy improvement step (Gulcehre et al. 2020; Brandfonbrener et al. 2021; Kostrikov, Nair, and Levine 2021). Trajectory optimization methods learn a model of the trajectory distribution induced by the behavior policy (Chen et al. 2021; Janner, Li, and Levine 2021). Our research focuses on model-based methods, which are similar to online model-based RL and employ a dynamics and rewards model as a proxy for the real environment, simulating transitions and then using them for planning or policy optimization.

Model-based offline RL algorithms use models learned offline, which cannot correct their mistakes by interacting with the environment. To mitigate the extrapolation error (Fujimoto, Meger, and Precup 2019), the most direct way is to penal the model when visiting regions far away from real dynamics, which is usually realized by uncertainty estimation (Deisenroth and Rasmussen 2011; Depeweg et al. 2016; Argenson and Dulac-Arnold 2020; Kidambi et al. 2020; Yu et al. 2020; Yang et al. 2021). There are also other research ways like constraining the learned policy to be close to the behavior policy (Swazinna, Udluft, and Runkler 2021; Cang et al. 2021; Matsushima et al. 2020) or leveraging model-generated synthetic data for better training (Yu et al. 2021; Wang et al. 2021). We, instead, focus on offering higher-quality exploration data for offline training. Previous work, such as COMBO (Yu et al. 2021) and RAMBO (Rigter, Lacerda, and Hawes 2022), attempted to use a random rollout policy but did not emphasize and address the exploration problem. The most relevant to our work are TATU (Zhang et al. 2023). However, TATU only considers truncating the synthetic trajectory and does not address the exploration problem, which has also been overlooked in previous work. OCEAN, instead, not only sheds light on the lack of exploration capabilities but also takes into account conservative constraints on exploration trajectories.

Preliminaries

A Markov Decision Process (MDP) is a mathematical formulation used to model an RL environment, which is defined by a 6-tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T, d_0, r, \gamma \rangle$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, $T(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ denotes the transition distribution, $d_0(\mathbf{s}_0)$ denotes the initial state distribution, $r(\mathbf{s}_t, \mathbf{a}_t)$ denotes the reward function, and $\gamma \in (0, 1]$ denotes the discount factor. The goal of RL is to find an optimal policy $\pi^*(\mathbf{a}|\mathbf{s})$ that maximizes the expected return for all trajectories induced by the policy, such that

$$\pi^* = \arg\min_{\pi} \mathbb{E}_{\tau \sim p_\pi(\cdot)} \left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right], \qquad (1)$$

where $p_\pi(\tau) = d_0(\mathbf{s}_0) \Pi_{t=0}^{\infty} [\pi(\mathbf{a}_t|\mathbf{s}_t) T(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)]$ is the probability density function for given trajectory τ and policy π.

In offline RL setting, a static dataset of transitions $\mathcal{D} = (\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, r_t)_i$ is given, where i indexes a transition in the dataset, the actions are generated by a behavior policy $\mathbf{a}_t \sim \pi_\beta(\cdot|\mathbf{s}_t)$. The objective of offline RL is the same as

the online case: to find a policy that maximizes the expected return. However, we cannot evaluate this objective under an arbitrary trajectory distribution $p_\pi(\tau)$, since π might experience distributional shift and visit states that we do not have any information from our static dataset.

Model-based offline RL typically estimates the transition dynamics $T_{\Phi_T}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ and the reward function $r_{\Phi_r}(\mathbf{s}_t, \mathbf{a}_t)$ using standard supervised regression with the dataset $\mathcal{D}$. Then the dynamics and rewards models serve as proxies for the real environment and are utilized to simulate transitions. These transitions are employed during the planning or policy optimization phase. Model-based methods tend to perform effectively when the data distribution offers extensive coverage, as it becomes easier to learn a precise model using such data.

TATU (Zhang et al. 2023) introduces trajectory truncation with uncertainty to model-based offline RL. They first calculate the truncation threshold ϵ as follow:

$$\epsilon = \frac{1}{\alpha} \max_{i \in [|\mathcal{D}|]} u(\mathbf{s}_i, \mathbf{a}_i), \tag{2}$$

where $u(\mathbf{s}_i, \mathbf{a}_i)$ is the uncertainty estimation of state-action pair $(\mathbf{s}_i, \mathbf{a}_i)$ and α is a hyperparameter. Then in the rollout phase, they calculate accumulated uncertainty along the rollout trajectory $U_j = \sum_{k=1}^{j} [u(\mathbf{s}_k, \mathbf{a}_k)]$. TATU will truncate the synthetic trajectory where $U_j > \epsilon$.

Method

In this section, we begin by further explaining the biased exploration in model-based offline RL settings. Subsequently, we present a comprehensive algorithm for OCEAN (Offline Conservative Exploration), providing step-by-step details of its implementation.

Explanation of Biased Exploration

We begin by further analyzing the reason behind the biased exploration depicted in Figure 1. It is widely acknowledged that achieving a balance between exploration and exploitation is fundamental in designing reinforcement learning algorithms. In the case of online RL algorithms, it is common practice to explore initially, followed by utilizing the gathered data and repeating this process. However, offline model-free RL does not inherently support further exploration. Hence, the model-free offline RL approaches aim to effectively utilize available data while suppressing exploration, such as constraining the strategy within the range supported by the dataset.

In the conventional model-based offline setting, the optimization objective is $\sum_{s,a} r(s,a) - U(s,a) + H(\pi(s))$, where U and H are correlated. A higher entropy corresponds to a policy that is more likely to produce out-of-distribution actions, leading to an increase in U, which results in a decrease in the overall objective and impedes the algorithm's ability to effectively explore. Also, the presence of $r - U$ has an antagonistic effect on H. This is because maximizing $r - U$ tends to promote a deterministic policy, which in turn leads to biased exploration.

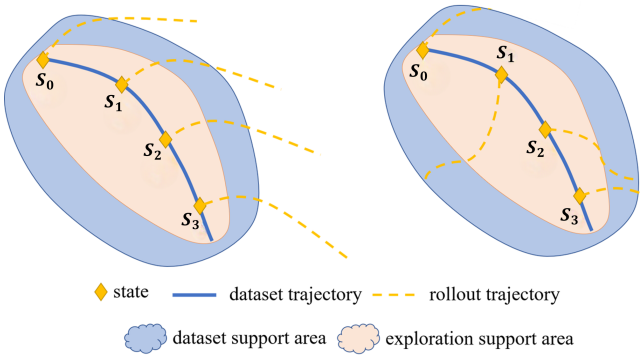


Figure 2: Comparison of OCEAN (right) against MOPO (left) on rollout trajectories. For states characterized by higher uncertainty, such as $\mathbf{S}_0$, we refrain from further exploration. Conversely, for states with lower uncertainty, such as $\mathbf{S}_1$ and $\mathbf{S}_2$, we explore within regions of exploration support area. In all trajectories, we ensure truncation within the range supported by the dataset.

By introducing additional exploration, we can decouple exploration and exploitation to facilitate a more comprehensive exploration of the transition model and promote stability in the distribution of sampled data. This stabilization ensures that the algorithm's asymptotic performance becomes more consistent and reliable. However, unlike interactions with the real environment, model-based exploration necessitates a delicate balance between exploration and model error. If the explored transition yields a substantial model error, it could detrimentally impact the algorithm's performance. To address this issue, we propose the introduction of an exploration module based on uncertainty estimation, ensuring that we explore regions where the model error is minimal. We will now delve into the details of Offline Conservative Exploration (OCEAN) below.

Offline Conservative Exploration

OCEAN is a plug-in method that can be seamlessly integrated into existing model-based offline frameworks. The algorithmic overview of OCEAN is depicted in Algorithm 1.

Following MBPO (Janner et al. 2019), we first model the dynamics using a neural network that outputs a Gaussian distribution over the next state and reward $\hat{T}_{\theta,\phi}(\mathbf{s}_{t+1}, r_t|\mathbf{s}_t, \mathbf{a}_t) = \mathcal{N}(\mu_\theta(\mathbf{s}_t, \mathbf{a}_t), \sum_\phi(\mathbf{s}_t, \mathbf{a}_t))$. In order to better estimate state transitions and facilitate the computation of uncertainties, we learn an ensemble of N dynamics models $\{\hat{T}^i_{\theta,\phi}\}_{i=1}^N$ with each model initialized randomly and trained independently via maximum likelihood. The loss function for training the forward dynamics models is as follows:

$$\mathcal{L}_{\theta,\phi} = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t, r, \mathbf{s}_{t+1}) \sim \mathcal{D}} \left[-\log \hat{T}_{\theta,\phi}(\mathbf{s}_{t+1}, r_t|\mathbf{s}_t, \mathbf{a}_t) \right]. \tag{3}$$

In practice, models output the difference between the current state and the next state, i.e., $\mu_\theta(\mathbf{s}_t, \mathbf{a}_t) = \mathbf{s} + \delta_\theta(\mathbf{s}_t, \mathbf{a}_t)$.

Then we will choose M elite models from N ensemble models for policy training.

After the completion of model training, similar to MBPO, we use a policy to roll out h steps with the model and update the policy with the collected data. However, OCEAN modifies the policy rollout phase to ensure the generation of more exploratory trajectories and guarantee that the generated data falls within the range supported by the model. Specifically, our method involves three constraints to enable conservative exploration. Firstly, for every state in the rollout trajectory, we determine whether it is suitable for exploration. We aim to avoid exploring areas where the model estimates are inaccurate. Secondly, we perform exploration on filtered states and select more conservative transitions to add to the buffer. Finally, we truncate the rollouts to ensure safe exploration within the dataset support region. In Figure 2, we provide an example to illustrate our algorithm. Below, we provide a detailed explanation of how each constraint is implemented.

State Evalution Constraint For the first constraint, at the beginning of each rollout step, to avoid exploring states where the model estimation deviates significantly from the real dynamics, we only explore current states with low uncertainty. It is important to highlight that in this context, we estimate the uncertainty of the states, rather than the uncertainty of the state-action pairs. To estimate the uncertainty of the current state, we employ a sampling approach. Specifically, we sample a set of n exploration actions $\{\mathbf{a}^e_{i,j}\}_{i=1,\dots,n}$ for state $\mathbf{s}_j$ from our exploration strategy

$$\mathbf{a}^e_{i,j} = \pi(\mathbf{s}_j) + \epsilon_i \tag{4}$$

where π refers to the origin policy and $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. In practice, our exploration policy is obtained by adding Gaussian noise with zero mean and a variance of δ^2 to the original policy π, which is named π^e. It is worth noting that in our implementation, the original policy is also modeled using a Gaussian distribution. However, the variance of the original policy is significantly smaller than that of the added Gaussian noise, which enables exploration of the policy in a broader range.

We then calculate the average uncertainty $\sum_{i=1}^{n} U^e_{i,j}$ of explored state-action pairs by an uncertainty estimator. The choice of uncertainty estimator is flexible, and based on the findings of (Lu et al. 2021), the available uncertainty estimators in our work include Max Aleatoric (Yu et al. 2020), Max Pairwise Diff (Kidambi et al. 2020), and Ensemble Standard Variance (Lakshminarayanan, Pritzel, and Blundell 2017). We set a hyperparameter of penalty threshold u_T to decide whether to explore the current state.

Exploration Range Constraint The second constraint limits the extent of exploration to the boundaries supported by the model. For the states deemed suitable for exploration, we reuse the exploration actions sampled previously for simplifying the calculation process. From the explored transitions, we adopt a more conservative approach in selecting the transitions, which can be done in the following three ways:

Algorithm 1: OCEAN

Require: Offline dataset $\mathcal{D}$, rollout horizon h, penalty threshold u_T, truncation coefficient λ

Initialize model buffer $\mathcal{D}_{\text{model}} \leftarrow \emptyset$
Train the N ensemble dynamics models on $\mathcal{D}$ with Equation 3
Calculate the max uncertainty u_{max} in dataset
for epoch in 1 to n **do**
  Sample state s_0 from dataset $\mathcal{D}$
  for j in 1 to h **do**
    Sample an action $\mathbf{a}_j \sim \pi(\cdot|\mathbf{s}_j)$
    Sample n actions $\{\mathbf{a}^e_{i,j}\}_{i=1,\dots,n}$ from exploration policy π_e
    Randomly pick dynamics $\hat{T}$ from $\{\hat{T}^i_\psi\}_{i=1}^N$
    sample next states $\mathbf{s}_{j+1}, \{\mathbf{s}^e_{i,j+1}\}_{i=1,..,n} \sim \hat{T}$ corresponds to $\mathbf{a}_j$ and $\{\mathbf{a}^e_{i,j}\}_{i=1,\dots,n}$
    Calculate the uncertainty U_j and $\{U^e_{i,j}\}_{i=1,\dots,n}$ respectively
    if $U_j \le \lambda \cdot u_{max}$ **then**
      if $\sum_{i=1}^{n} U^e_{i,j} \le u_T$ **then**
        Select an imagined transition from $\{(\mathbf{s}^e_{i,j}, \mathbf{a}^e_{i,j}, r_j, \mathbf{s}^e_{i,j+1})\}_{i=1,\dots,n}$ into the model buffer $\mathcal{D}_{\text{model}}$
      else
        Put the imagined transition $(\mathbf{s}_j, \mathbf{a}_j, r_j, \mathbf{s}_{j+1})$ into the model buffer $\mathcal{D}_{\text{model}}$
      end if
    else
      break
    end if
  end for
  Sample data from $\mathcal{D} \cup \mathcal{D}_{\text{model}}$ and optimize policy π
end for

- Min Uncertainty: $\arg\min_{\mathbf{a}^e_i}\{u(\mathbf{s}, \mathbf{a}^e_i), i = 1, ..., n\}$, which corresponds to the state-action pair that has the minimum uncertainty.
- Median Uncertainty: $\arg\text{medium}_{\mathbf{a}^e_i}(\{u(\mathbf{s}, \mathbf{a}^e_i), i = 1, ..., n\})$, which corresponds to the state-action pair that has the medium uncertainty.
- Random Uncertainty: random sample action from the exploration actions.

Trajectory Truncation Constraint Finally, to prevent the influence of model error caused by excessively long rollouts, we impose a limit on the rollout length. Similar to the TATU method (Zhang et al. 2023), we employ an automatic clipping approach based on uncertainty. Specifically, we calculate the maximum uncertainty u_{max} exhibited by the model within the dataset. Then, at each rollout step, we calculate the uncertainty corresponding to the state-action pair generated by the rollout policy. If the calculated uncertainty exceeds the threshold of $\lambda \cdot u_{max}$, we truncate the trajectory at that step. Here, λ is a tunable hyperparameter, which is typically set to 1 in our experiments. By implementing this automatic clipping method, we restrict the length of rollouts to mitigate the impact of model errors. The threshold based

Task Name	MOPO	MOPO+OCEAN	CQL	TD3+BC	IQL
halfcheetah-m	73.2(4.1)	**74.4(2.0)**	**49.4**	48.2	47.4
hopper-m	31.4(25.7)	**87.4(22.8)**	59.1	60.8	**65.7**
walker2d-m2	87.3(9.6)	**89.9(2.1)**	83.6	**84.4**	81.1
halfcheetah-m-r	**70.1(2.2)**	68.8(2.1)	**47.0**	45.0	44.2
hopper-m-r	98.1(16.2)	**103.4(1.8)**	**98.6**	67.3	94.8
walker2d-m-r	72.7(20.1)	**90.7(3.3)**	71.3	**83.4**	77.3
halfcheetah-m-e	79.3(12.6)	**99.3(2.0)**	**93.0**	90.7	88.0
hopper-m-e	82.5(33.2)	**110.0(1.2)**	**111.4**	91.4	106.2
walker2d-m-e	102.2(11.7)	**108.7(7.0)**	109.8	**110.2**	108.3
Average Score	77.4(5.9)	**92.5(2.7)**	**80.35**	75.71	79.13

Table 1: Normalized average score comparison of OCEAN+MOPO against MOPO and some recent baselines on the D4RL MuJoCo "-v2" dataset. r=random, m=medium, m-r=medium-replay, m-e=medium-expert. MOPO-based algorithms run for 2M gradient steps across 8 different random seeds and the final mean performance of 100 episodes is reported. $(\cdot)$ captures the standard deviation. The top score of each environment for each part is bolded.

Task Name	COMBO	COMBO+OCEAN	RAMBO	RAMBO+OCEAN
halfcheetah-m	**76.7(5.3)**	75.6(5.1)	**74.9(1.7)**	73.1(2.0)
hopper-m	97.4(2.6)	**100.9(0.3)**	98.5(19.0)	**103.7(2.7)**
walker2d-m2	81.2(4.2)	**82.6(0.9)**	84.7(5.5)	**84.7(11.1)**
halfcheetah-m-r	68.1(4.5)	**70.8(1.3)**	68.0(2.2)	**68.9(2.2)**
hopper-m-r	95.4(15.5)	**104.1(0.8)**	81.7(26.3)	**93.5(13.9)**
walker2d-m-r	**77.1(10.1)**	76.4(17.2)	79.8(9.5)	**93.2(6.4)**
halfcheetah-m-e	97.3(1.1)	**98.9(0.4)**	94.9(5.6)	**96.3(4.1)**
hopper-m-e	107.2(6.0)	**111.2(1.9)**	**86.6(27.1)**	86.1(26.1)
walker2d-m-e	108.0(4.1)	**110.4(0.7)**	87.8(32.5)	**101.7(23.7)**
Average Score	89.8(2.4)	**92.3(2.0)**	84.1(6.1)	**89.0(4.5)**

Table 2: Normalized average score comparison of OCEAN+COMBO and OCEAN+RAMBO against their base algorithms on the D4RL MuJoCo "-v2" dataset. r=random, m=medium, m-r=medium-replay, m-e=medium-expert. All algorithms run for 1M gradient steps across 8 different random seeds and the final mean performance of 100 episodes is reported. $(\cdot)$ captures the standard deviation. The top score of each environment is bolded.

on uncertainty allows us to control the influence of uncertain predictions and maintain the reliability of the rollout process. We truncate the synthetic trajectory like TATU, but TATU calculates the cumulative uncertainty of the entire trajectory, whereas we solely focus on the uncertainty of a single state-action pair. This narrower focus on individual state uncertainty allows us to adopt a more optimistic approach when truncating rollouts.

Experiments

Our experiments aim to: a) evaluate how well OCEAN improves the performance of state-of-art offline model-based methods, b) examine whether conservative exploration is necessary, c) determine the impact of different hyperparameters on the performance of the algorithm.

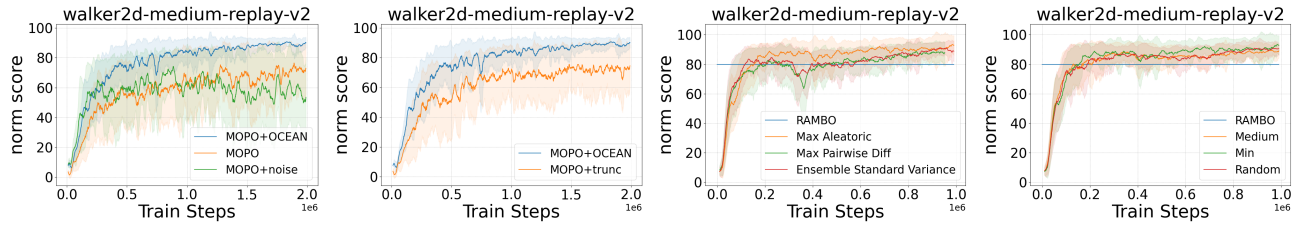
To achieve these goals, we first combine OCEAN with popular model-based offline RL algorithms, MOPO, COMBO and RAMBO. Then we proceed to validate the efficacy of exploration and conservatism in the OCEAN algorithm individually. Additionally, we conduct experiments to explore various uncertainty estimators and exploration strategies. Lastly, we perform parameter tuning experiments on the two pivotal hyperparameters in OCEAN, namely the penalty threshold u_T and noise standard deviation δ.

We evaluate our approach on D4RL (Fu et al. 2020) MuJoCo datasets and our code is based on OfflineRL-Kit library[1]. It is a relatively comprehensive and high-performance library in the current offline model-based RL implementations. The base hyperparameters that we use for OCEAN mostly follow the OfflineRL-Kit library.

Results

In order to assess the potential performance improvements of OCEAN compared to MOPO, COMBO, and RAMBO, we conducted a series of experiments on nine D4RL MuJoCo datasets. These datasets are commonly used for evaluating the performance of model-based offline RL algorithms. Our result is shown in Table 1 and Table 2. It is important to highlight that the experimental results obtained from our base algorithms may not precisely match those of the OfflineRL-Kit library. This discrepancy arises because, for each task and seed combination, we utilize the same dynamic model instead of training a unique model for

[1]https://github.com/yihaosun1124/OfflineRL-Kit



(a) (b) (c) (d)

Figure 3: Ablation study experiments on the walker2d-medium-replay-v2 task. Figure 3a compares MOPO, MOPO+OCEAN and MOPO with noise, which illustrates the effectiveness of the conservative. Figure 3b compares MOPO+OCEAN and MOPO with trajectory truncation and validates the effectiveness of the exploration. Figures 3c and 3d run on RAMBO+OCEAN and illustrate the effectiveness of different uncertainty estimators and different exploration action selection strategies, respectively. MOPO-based experiments run for 2M gradient steps and RAMBO-based experiments run for 1M gradient steps.

each seed. This approach allows us to expedite the iterative process and promptly validate the effectiveness of our approach. The results of model-free approaches are from the OfflineRL-Kit library. More implementation details can be found in our appendix and we also show all the training curves compared to the baseline in the appendix.

Table 1 shows results for MOPO, MOPO+OCEAN, and model-free baselines CQL (Kumar et al. 2020), TD3+BC (Fujimoto and Gu 2021), and IQL (Kostrikov, Nair, and Levine 2021). It can be found that OCEAN markedly boosts the performance of MOPO on most of the datasets, especially in the hopper task(hopper-m, hopper-m-r, and hopper-m-e) which improved from 70.7 to 100.3 on average score. The total average score over 9 tasks gains 19.5% improvement and the total standard deviation reduces by 54.2%, which means with the inclusion of our method, the algorithm experiences enhanced stability and demonstrates superior performance. Also, we observe a competitive or better performance of MOPO+OCEAN on the evaluated datasets against model-free methods like CQL, IQL, and TD3+BC. MOPO+OCEAN has the best average score of 92.5 across all of the algorithms.

Table 2 shows results for COMBO, COMBO+OCEAN, RAMBO, and RAMBO+OCEAN. We find that OCEAN improves the performance of COMBO and RAMBO on many datasets. Additionally, we observe that the impact of OCEAN on the overall performance improvement of RAMBO and COMBO is relatively limited. This can be attributed to the fact that RAMBO inherently influences the rollout stage through implicit model updates and possesses certain exploration capabilities and COMBO adopts a more conservative strategy, resulting in a reduced contribution of our conservative constraints to the performance enhancement.

Ablation Study

To demonstrate the effectiveness of OCEAN, we conduct exhaustive ablation experiments on the walker2d-medium-replay-v2 task. The result is shown in Figure 3.

Conservative validity. To illustrate the significance of conservative strategy, we conducted a comparison among three approaches: MOPO, MOPO with action noise, and MOPO with OCEAN, as depicted in Figure 3a. Both action noise and OCEAN employ Gaussian noise with a standard deviation of 0.5. It can be observed that due to excessive exploration, the disparity between the rollout transition and the real environment transition becomes too large, resulting in a similar performance of the original MOPO and MOPO with noise during the later stages of training. In contrast, by adopting the conservative estimation of OCEAN, MOPO+OCEAN demonstrates not only a substantial improvement in asymptotic performance but also a more stable convergence value, which gains 24.8% performance improvement in average score and 83.6% standard deviation reduction. Based on the experiment, we conclude that OCEAN experiences a significant performance boost due to the incorporation of conservative constraints.

The effectiveness of exploration. Figure 3b presents a comparison between MOPO with OCEAN and MOPO solely with trajectory truncation. The latter is not explored within the supported scope of the model. From the results, we can find that on average, OCEAN with trajectory truncation exhibits a significant decrease of 14.3 in score compared to the regular OCEAN algorithm. Additionally, the use of trajectory truncation introduces an increase of 12 in the performance standard deviation. It becomes evident that although the algorithm's performance has improved with the inclusion of trajectory truncation, there remains a substantial gap in both performance and stability when compared to OCEAN. By considering both Figure 3a and Figure 3b together, it becomes apparent that OCEAN emphasizes not only conservation but also the importance of exploration as a fundamental component.

Choices of uncertainty estimator. The uncertainty heuristics we used in our work is listed below:

- Max Aleatoric: $\max_{i=1}^{N} ||\sum_{\phi}^{i}(\mathbf{s},\mathbf{a})||_F$, which computed over the variance heads of model ensemble.
- Max Pairwise Diff: $\max_{i,j} |||\mu_{\theta}^{i}(\mathbf{s},\mathbf{a}) - \mu_{\theta}^{j}(\mathbf{s},\mathbf{a})|$, which corresponds to the pairwise maximum difference of the ensemble predictions.
- Ensemble Standard Variance: $\sum^{*}(\mathbf{s},\mathbf{a}) = \frac{1}{N}\sum_{i=1}^{N}((\sum_{\phi}^{i}(\mathbf{s},\mathbf{a}))^2 + (\mu_{\phi}^{i}(\mathbf{s},\mathbf{a}))^2 - (\mu^{*}(\mathbf{s},\mathbf{a}))^2$ where $\mu^{*}(\mathbf{s},\mathbf{a}) = \frac{1}{N}\sum_{i=1}^{N}\mu_{\phi}^{i}(\mathbf{s},\mathbf{a})$, which corresponds

to a combination of epistemic and aleatoric model uncertainty.

Based on Figure 3c, it is evident that the performance disparity among the three uncertainty estimates is not significantly large. The performance of Max Aleatoric and Ensemble Standard Variance is relatively similar, both slightly outperforming Max Pairwise Diff. In our experiments, we default to using Max Aleatoric as the uncertainty estimator.

Choices of exploration strategy. As depicted in figure 3d, we examined the impact of selecting three different exploration strategies, which have been elaborated in the methods section. It is evident that the disparity among various strategies for selecting noise actions is relatively small. The "min" strategy, which selects the noise action with the least uncertainty, exhibits slightly better performance compared to the other two strategies. This observation suggests that the conservative nature of the "min" strategy could be the contributing factor to its relatively superior performance.

Hyperparameter Tuning

There are generally two key hyperparameters in OCEAN, the penalty threshold u_T and the Gaussian noise standard deviation δ.

Penalty threshold $\mathbf{u_T}$. The threshold parameter u_T is indeed a crucial parameter for OCEAN, as it determines the extent of exploration in the algorithm. A larger value of u_T will result in more exploration across states, thereby increasing the algorithm's exploration strength. The parameter u_T is closely tied to model loss. In tasks where the model fits well and exhibits a smaller loss, it is advisable to set a smaller value for the parameter u_T, which allows for more conservative exploration and vice versa. We reported the experiment results in Table 3, where 'mean' refers to using the average uncertainty of a batch of data as the threshold. Our findings indicate that both excessively large and excessively small thresholds have a detrimental effect on the algorithm's performance. Smaller thresholds limit exploration opportunities, while larger thresholds tend to explore regions with high uncertainty, leading to increased model errors. Additionally, our experiments demonstrate that utilizing the average uncertainty of a batch as the threshold yields satisfactory performance.

u_T	MOPO+	COMBO+	RAMBO+
1	86.3(2.8)	60.5(21.2)	82.2(6.6)
2	87.7(4.0)	76.4(17.2)	66.2(20.0)
3	83.0(7.8)	76.1(21.0)	83.3(4.0)
mean	90.7(3.3)	74.5(12.0)	93.2(6.4)

Table 3: Comparison of MOPO+OCEAN, COMBO+OCEAN and RAMBO+OCEAN with different penalty threshold u_T on walker2d-medium-replay-v2. MOPO results run for 2M gradient steps and other results run for 1M. All results run across 8 different random seeds and averaged over the final 10 evaluations. + means +OCEAN. "mean" means using the average uncertainty of a batch of data as the threshold.

Gaussian noise standard deviation δ. The Gaussian noise standard deviation controlled the intensity of exploration. As indicated in Table 4, selecting a smaller δ brings the algorithm's performance closer to that of the baseline algorithm, while a larger δ may have a detrimental effect on the algorithm's performance. For instance, in both COMBO and RAMBO, using a delta value of 1 leads to performance degradation, which can be attributed to the fact that larger noise values often correspond to greater model errors. Furthermore, we conducted a comparison of the performance of random actions. It is observed that random actions yield relatively good performance. This can be attributed to the higher degree of exploration associated with random actions compared to noise actions. In general, striking a balance between exploration and conservatism is crucial for achieving optimal algorithm performance. Therefore, the algorithm's sensitivity to changes in δ is higher. In our experiments, we typically opt for a δ value of 0.5 or random actions.

δ	MOPO+	COMBO+	RAMBO+
0.1	82.0(2.5)	72.8(12.9)	77.7(3.2)
0.3	80.4(10.9)	68.5(13.1)	82.1(6.7)
0.5	88.8(3.1)	74.5(12.0)	93.2(6.4)
0.7	89.9(2.2)	57.5(20.3)	87.7(3.2)
1	90.7(3.3)	60.4(26.5)	69.1(11.9)
random	89.3(2.5)	76.4(17.2)	76.8(4.7)

Table 4: Comparison of MOPO+OCEAN, COMBO+OCEAN and RAMBO+OCEAN with Gaussian standard deviation δ on walker2d-medium-replay-v2. MOPO results run for 3M gradient steps and other results run for 1M. All results run across 8 different random seeds and are averaged over the final 10 evaluations. + means +OCEAN. "random" means using random actions sampled from a uniform distribution from -1 to 1.

Conclusion

This paper addresses the issue of biased exploration in model-based offline RL and attempts to enhance the performance of existing algorithms by introducing additional exploration. We propose offline Conservative Exploration (OCEAN), which employs noise exploration while utilizing uncertainty estimation to prevent the policy from exploring regions with high model uncertainty. Our approach is a plug-in method that can be combined with state-of-the-art model-based offline methods such as MOPO, COMBO, and RAMBO. Experimental results demonstrate that our method significantly improves the performance and stability of current approaches. Furthermore, our ablation experiments validate the effectiveness of conservative constraints and additional exploration.

Acknowledgements

This work is partially supported by the NSF of China (under Grant 92364202).

References

An, G.; Moon, S.; Kim, J.-H.; and Song, H. O. 2021. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems*, 34: 7436–7447.

Argenson, A.; and Dulac-Arnold, G. 2020. Model-based offline planning. *arXiv preprint arXiv:2008.05556*.

Brandfonbrener, D.; Whitney, W.; Ranganath, R.; and Bruna, J. 2021. Offline rl without off-policy evaluation. *Advances in neural information processing systems*, 34: 4933–4946.

Cang, C.; Rajeswaran, A.; Abbeel, P.; and Laskin, M. 2021. Behavioral priors and dynamics models: Improving performance and domain transfer in offline rl. *arXiv preprint arXiv:2106.09119*.

Chen, L.; Lu, K.; Rajeswaran, A.; Lee, K.; Grover, A.; Laskin, M.; Abbeel, P.; Srinivas, A.; and Mordatch, I. 2021. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34: 15084–15097.

Deisenroth, M.; and Rasmussen, C. E. 2011. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, 465–472.

Depeweg, S.; Hernández-Lobato, J. M.; Doshi-Velez, F.; and Udluft, S. 2016. Learning and policy search in stochastic dynamical systems with bayesian neural networks. *arXiv preprint arXiv:1605.07127*.

Fu, J.; Kumar, A.; Nachum, O.; Tucker, G.; and Levine, S. 2020. D4RL: Datasets for Deep Data-Driven Reinforcement Learning. arXiv:2004.07219.

Fujimoto, S.; and Gu, S. S. 2021. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34: 20132–20145.

Fujimoto, S.; Meger, D.; and Precup, D. 2019. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, 2052–2062. PMLR.

Gulcehre, C.; Colmenarejo, S. G.; Sygnowski, J.; Paine, T.; Zolna, K.; Chen, Y.; Hoffman, M.; Pascanu, R.; de Freitas, N.; et al. 2020. Addressing Extrapolation Error in Deep Offline Reinforcement Learning.

Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. 2018. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.

Janner, M.; Fu, J.; Zhang, M.; and Levine, S. 2019. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32.

Janner, M.; Li, Q.; and Levine, S. 2021. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34: 1273–1286.

Kalashnikov, D.; Irpan, A.; Pastor, P.; Ibarz, J.; Herzog, A.; Jang, E.; Quillen, D.; Holly, E.; Kalakrishnan, M.; Vanhoucke, V.; et al. 2018. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, 651–673. PMLR.

Kidambi, R.; Rajeswaran, A.; Netrapalli, P.; and Joachims, T. 2020. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33: 21810–21823.

Kiran, B. R.; Sobh, I.; Talpaert, V.; Mannion, P.; Al Sallab, A. A.; Yogamani, S.; and Pérez, P. 2021. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6): 4909–4926.

Kostrikov, I.; Nair, A.; and Levine, S. 2021. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*.

Kumar, A.; Fu, J.; Soh, M.; Tucker, G.; and Levine, S. 2019. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32.

Kumar, A.; Zhou, A.; Tucker, G.; and Levine, S. 2020. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191.

Lakshminarayanan, B.; Pritzel, A.; and Blundell, C. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30.

Lange, S.; Gabel, T.; and Riedmiller, M. 2012. Batch reinforcement learning. In *Reinforcement learning: State-of-the-art*, 45–73. Springer.

Levine, S.; Kumar, A.; Tucker, G.; and Fu, J. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*.

Lu, C.; Ball, P. J.; Parker-Holder, J.; Osborne, M. A.; and Roberts, S. J. 2021. Revisiting Design Choices in Offline Model-Based Reinforcement Learning. *arXiv preprint arXiv:2110.04135*.

Matsushima, T.; Furuta, H.; Matsuo, Y.; Nachum, O.; and Gu, S. 2020. Deployment-efficient reinforcement learning via model-based offline optimization. *arXiv preprint arXiv:2006.03647*.

Nachum, O.; Dai, B.; Kostrikov, I.; Chow, Y.; Li, L.; and Schuurmans, D. 2019. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*.

Prudencio, R. F.; Maximo, M. R.; and Colombini, E. L. 2023. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*.

Quillen, D.; Jang, E.; Nachum, O.; Finn, C.; Ibarz, J.; and Levine, S. 2018. Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 6284–6291. IEEE.

Rigter, M.; Lacerda, B.; and Hawes, N. 2022. Rambo-rl: Robust adversarial model-based offline reinforcement learning. *Advances in neural information processing systems*, 35: 16082–16097.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587): 484–489.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Swazinna, P.; Udluft, S.; and Runkler, T. 2021. Overcoming model bias for robust offline deep reinforcement learning. *Engineering Applications of Artificial Intelligence*, 104: 104366.

Wang, J.; Li, W.; Jiang, H.; Zhu, G.; Li, S.; and Zhang, C. 2021. Offline reinforcement learning with reverse model-based imagination. *Advances in Neural Information Processing Systems*, 34: 29420–29432.

Wu, Y.; Tucker, G.; and Nachum, O. 2019. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*.

Wu, Y.; Zhai, S.; Srivastava, N.; Susskind, J.; Zhang, J.; Salakhutdinov, R.; and Goh, H. 2021. Uncertainty weighted actor-critic for offline reinforcement learning. *arXiv preprint arXiv:2105.08140*.

Yang, Y.; Jiang, J.; Zhou, T.; Ma, J.; and Shi, Y. 2021. Pareto policy pool for model-based offline reinforcement learning. In *International Conference on Learning Representations*.

Yu, T.; Kumar, A.; Rafailov, R.; Rajeswaran, A.; Levine, S.; and Finn, C. 2021. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34: 28954–28967.

Yu, T.; Thomas, G.; Yu, L.; Ermon, S.; Zou, J. Y.; Levine, S.; Finn, C.; and Ma, T. 2020. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33: 14129–14142.

Yurtsever, E.; Capito, L.; Redmill, K.; and Ozgune, U. 2020. Integrating deep reinforcement learning with model-based path planners for automated driving. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, 1311–1316. IEEE.

Zhang, J.; Lyu, J.; Ma, X.; Yan, J.; Yang, J.; Wan, L.; and Li, X. 2023. Uncertainty-driven Trajectory Truncation for Model-based Offline Reinforcement Learning. *arXiv preprint arXiv:2304.04660*.