ENTROPY-REGULARIZED MODEL-BASED OFFLINE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Model-based approaches to offline Reinforcement Learning (RL) aim to remedy the problem of sample complexity in offline learning via first estimating a pessimistic Markov Decision Process (MDP) from offline data, followed by freely exploring in the learned MDP for policy optimization. Recent advances in model-based RL techniques mainly rely on an ensemble of models to quantify the uncertainty of the empirical MDP which is leveraged to penalize out-of-distribution state-action pairs during the policy learning. However, the performance of ensembles for uncertainty quantification highly depends on how they are implemented in practice, which can be a limiting factor. In this paper, we propose a systematic way to measure the epistemic uncertainty and present EMO, an Entropy-regularized Model-based Offline RL approach, to provide a smooth error estimation when leaving the support of data toward uncertain areas. Subsequently, we optimize a single neural architecture that maximizes the likelihood of offline data distribution while regularizing the transitions that are outside of the data support. Empirical results demonstrate that our framework achieves competitive performance compared to state-of-the-art offline RL methods on D4RL benchmark datasets.

1 INTRODUCTION

Following the major success of deep Reinforcement Learning (RL) in numerous applications (Mnih et al., 2013; 2015; Silver et al., 2018), offline RL has emerged to cope with the problems where simulation or online interaction is impractical, costly, and/or dangerous, thus, allowing to automate a wide range of decision-making problems from healthcare and education to finance and robotics (Levine et al., 2020). The primary challenge in these scenarios is however that learning new policies from data stored with a different (possibly sup-optimal) policy, aka behavior policy, suffers from distributional shifts resulting in *extrapolation error*, which is infeasible to improve due to lack of additional exploration (Fujimoto et al., 2019; Kumar et al., 2019). This is why standard (online) RL methods perform poorly in offline settings (Yu et al., 2020). Consequently, several model-free offline RL algorithms are introduced to regularize the learned policies to stay close to the behavior policy, by constraining out-of-distribution trajectories (Fujimoto et al., 2019; Kumar et al., 2019; Wu et al., 2019; Kumar et al., 2020; Agarwal et al., 2020).

Nevertheless, the overly conservative optimization scheme in model-free approaches impedes generalization, leading to policies that are only slightly improved w.r.t. the behavior policy. In addition, policy optimization is limited to already observed states which most likely do not provide sufficient coverage of the entire state space. Alternatively, model-based methods first learn the corresponding empirical Markov Decision Process (MDP) using the offline dataset and then freely explore in the learned environment for policy optimization, which can attain excellent sample efficiency (Chua et al., 2018; Janner et al., 2019). Most recently, model-based algorithms are specifically designed for offline settings to address distributional shifts in the learned model and have been proved effective in certain problems compared to their model-free counterparts (Yu et al., 2020; 2021; Kidambi et al., 2020; Zhan et al., 2021; Swazinna et al., 2021; Chen et al., 2021; Rigter et al., 2022)

However, most prominent model-based methods, i.e., MOPO (Yu et al., 2020) and MOReL (Kidambi et al., 2020), mainly leverage an ensemble of models for uncertainty quantification, in which a measure of ensemble discrepancy determines the estimation error. This can be particularly restrictive when theoretical assumptions on the ensemble do not hold in practical scenarios. In practice, an

ensemble usually consists of a small number of models, where each model is a different initialization of the same neural architecture, trained on the same data. Hence, the models in the ensemble are likely to become correlated to one another after training, which might make their variation a poor indicator of uncertainty. Yu et al. (2021) study this behavior and demonstrate that the uncertainty estimated via maximum variance over the learned ensemble (as in MOPO) struggles to accurately predict the model's error , and thus, could lead to poor performance (see Fig. 2 in Yu et al. (2021)).

Accordingly, there have been efforts to eliminate the need for ensemble uncertainty estimation in model-based offline RL methods. Yu et al. (2021) propose utlizing model rollouts to conservatively learn the Q-function by penalizing the values over out-of-distribution areas, while Rigter et al. (2022) introduce an adversarial framework for training the policy and the model at the same time, such that at each step, the policy is trained to maximize the return, while the model is tuned to minimize it. Although RAMBO-RL (Rigter et al., 2022) and COMBO (Yu et al., 2021) have shown promising empirical results on standard benchmark datasets, they both forgo the modularity aspect of methods such as MOPO and MOReL. Instead, we aim to get the best of two worlds and present a general-purpose, task-agnostic framework to learn a pessimistic model of the environment that can be coupled with any RL algorithm to learn optimal policies, without ensemble learning.

In this paper, we address this problem by proposing a novel method that eliminates the need for ensemble uncertainty quantification, while still being modular in the sense that the trained model can be combined with arbitrary RL algorithms to learn arbitrary tasks. Therefore, we present EMO, an Entropy-regularized Model-based Offline RL approach, that learns a pessimistic MDP using only a single model which can provide accurate estimates of the dynamics in the support of offline data while softly quantifies an upper bound for the uncertainty of model predictions when leaving the data support. To this end, we devise a regularized loss function to minimize the negative log-likelihood of the model w.r.t. the offline data distribution, and simultaneously, maximize the entropy of predictions outside of the data support in a single model. Furthermore, we propose to warm-start the learning procedure by only optimizing the unconstrained objective function, where the initial learned model in this step is used to generate rollouts for optimizing the uncertainty estimation.

Our extensive empirical study illustrates that our approach achieves better or on par performance compared to state-of-the-art offline RL techniques, both model-free and model-based, on D4RL benchmark datasets for MuJoCo environments.

2 Related Work

Offline reinforcement learning (Lange et al., 2012), which allows for optimizing policies from static offline datasets, has received a lot of attention throughout the recent years, as the practical issues of applying online RL to many real-world scenarios became more apparent. Model-free offline RL approaches optimize a policy solely based on the visited states from the static offline data, without utilizing a learned model of the environment. Constraining the policy to be close to the behavior policy (Kumar et al., 2019; Fujimoto et al., 2019; Wu et al., 2019; Fujimoto & Gu, 2021), conservative estimation of value functions (Kumar et al., 2020; Kostrikov et al., 2021), and incorporating the uncertainty of predictions to stabilize Q-functions (Agarwal et al., 2020; Wu et al., 2021) are among active lines of work in model-free offline RL. However, due to their limited generalization, the performance of model-free methods is highly reliant on the optimality of the offline data.

On the other hand, model-based approaches incorporate a model of the environment to improve generalization and sample efficiency, which is used as a surrogate for the actual MDP to optimize a policy, combined with the original offline data. MOPO (Yu et al., 2020) and MOReL (Kidambi et al., 2020) incorporate ensemble uncertainty estimation to penalize highly uncertain transitions. COMBO (Yu et al., 2021) combines the idea of conservative estimation of value functions in CQL (Kumar et al., 2020) with a model-based learning framework. RAMBO-RL (Rigter et al., 2022) proposes an adversarial framework for training the policy and the model at the same time, such that at each step, the policy is trained to maximize the return, while the model is tuned to minimize it.

As discussed in (Yu et al., 2021), algorithms like MOPO and MOReL that rely on ensemble uncertainty estimation, are prone to erroneous estimations of uncertainty, in scenarios where the models within an ensemble become highly correlated to one another. On the other hand, COMBO and RAMBO-RL are not modular frameworks; COMBO applies the idea of conservative learning of

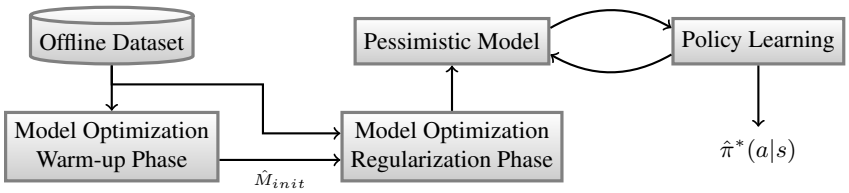

Figure 1: General scheme of EMO.

value function as a part of its proposed RL algorithm, and as a result, it cannot be coupled with any arbitrary RL algorithm. RAMBO-RL tunes the environment to be pessimistic with respect to the current policy of the agent; therefore, the resulting model from learning a particular task cannot be utilized to learn a new task in the environment. EMO, however, tries to eliminate the need for ensemble uncertainty quantification, while remaining a modular framework that can be extended to various tasks and RL algorithms.

3 EMO: Entropy-Regularized Model-based Offline RL

3.1 Preliminaries

The Reinforcement Learning (RL) problem is characterized via a Markov Decision Process (MDP) (Sutton & Barto, 2018), represented by $M = (\mathcal{S}, \mathcal{A}, T, r, \gamma, \rho_0)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state space and action space, respectively, $T(s'|s,a)$ is the transition distribution, $r(s,a)$ is the reward function, $\gamma \in (0,1)$ is the discount factor, and ρ_0 is the distribution of the initial state. A policy $\pi(a|s)$ is defined as a mapping from states to a distribution over actions, $\pi : \mathcal{S} \times \mathcal{A} \to [0,1]$, and the goal is to learn a policy π^* which maximizes the expected discounted return $\eta_M(\pi)$ when followed

$$\pi^* = \arg\max_{\pi} \ \eta_M(\pi) \qquad \text{where} \quad \eta_M(\pi) = \mathbb{E}_{\pi,\rho_0,T}\Big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\Big].$$

However, in offline settings, we are not able to accurately evaluate the return values under different policies as the interaction with the environment is no longer possible. Instead, a static dataset $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^n$ which is collected under an arbitrary policy during the past interactions with the environment is provided. Therefore, the objective turns into finding the "best-performing" policy that can be solely optimized on the available offline data $\mathcal{D}$. Note that in this context, "best-performing" policy might be different from the optimal policy, as the performance of the resulting policy, regardless of the learning algorithm, is affected and capped by factors such as distribution and optimality of the static data.

In this paper, we focus on model-based offline reinforcement learning in which an empirical model of the environment, i.e., the transition dynamics T along with the reward function r, is estimated and leveraged to enhance sample efficiency. In this framework, the offline dataset is used to train a *pessimistic* MDP $\hat{M}$ which is then employed as a surrogate for the actual model of the environment. Subsequently, the RL agent interacts with this model, generates samples of transitions, and optimizes its policy based on both the acquired information and the original transitions from the offline data. Ideally, we aim to find a policy $\hat{\pi}^*$ with minimum sub-optimality with respect to the optimal policy, i.e., $\hat{\pi}^* = \arg\min_{\pi} \ \eta_M(\pi^*) - \eta_M(\pi)$. However, as we do not have access to the actual MDP and the policy can only be optimized through the surrogate model $\hat{M}$, the problem statement turns into

$$\hat{\pi}^* = \arg\min_{\pi} \ \eta_M(\pi^*) - \eta_{\hat{M}}(\pi).$$

The general workflow of EMO is depicted in Figure 1. In this figure, the offline dataset is first utilized to train a pessimistic model of the environment in a two-phase training regime. After that, the model is considered as a surrogate for the actual environment to train a policy $\hat{\pi}^*(a|s)$.

3.2 MODEL OPTIMIZATION

In this section, we present a novel approach for model optimization, as the main part of EMO, that learns a pessimistic MDP using only a single neural network with two aims. First, the estimated model has to be as close to the actual dynamics of the environment as possible (which can only be reliable in the support of offline data,) and second, the uncertainty associated with model predictions must be relatively quantified in the form of an error estimator. Consequently, we address these goals in a regularized optimization setting, where both the likelihood of the data as well as the error estimation are jointly optimized in a single model.

The error estimator, denoted by u, allows the framework to relatively differentiate between the reliable and unreliable predictions of the model, which accordingly, can be utilized to penalize unreliable predictions based on their estimated uncertainty. Let $r(s, a)$ and $u(s, a)$ be the associated reward function and estimated error for a particular state-action pair (s, a), respectively. A pessimistic model can be determined by defining an alternative reward value in the form of $\tilde{r}(s, a) = r(s, a) - \lambda u(s, a)$, where λ is a constant value to control the amount of penalty associated with the state-action pair. As a result, the policies will be prevented from exploiting unreliable predictions, which can translate to conservative exploration in the actual environment.

Inspired by Yu et al. (2020); Kidambi et al. (2020), we characterize the one-step model of the environment with a Gaussian distribution over the next state and reward, conditioned on the current state and action, i.e., $P(s', r|s, a) = \mathcal{N}\big(\mu_\theta(s, a), \Sigma_\phi(s, a)\big)$, where θ and ϕ are the weights of the corresponding neural networks. Both the mean vector $\mu_\theta(s, a)$ and covariance matrix $\Sigma_\phi(s, a)$ are of the size $d + 1$, where d is the dimensionality of the state space and the covariance matrix is assumed to be diagonal. In this work, the main reason for explicitly estimating the covariance matrix lies in the fact that the entropy of a Gaussian distribution, as a quantitative measure for uncertainty, is formulated as a function of the covariance matrix

$$H = \frac{1}{2} \log \Big(\det \big(\Sigma_\phi(s, a)\big)\Big) + \frac{(d+1)}{2}\big(1 + \log(2\pi)\big),$$

and assuming that the covariance matrix is diagonal, we have:

$$H = \frac{1}{2} \sum_{i=1}^{d+1} \log \big(\Sigma_{\phi,i}(s, a)\big) + \frac{(d+1)}{2}\big(1 + \log(2\pi)\big). \tag{1}$$

Consequently, $\Sigma_\phi(s, a)$ can solely act as the uncertainty/error estimator, while $\mu_\theta(s, a)$ is trained to model the transition dynamics. In this way, both terms can be optimized in a single model to learn a pessimistic MDP that allows for utilizing the offline data in a more efficient and reliable way.

Furthermore, we propose a two-step algorithmic framework for learning the pessimistic model of the environment. During the first phase, indicated by the *warm-up* phase, the model is trained via maximum likelihood following the prior work (Yu et al., 2020; Kidambi et al., 2020), while in the second phase, which we call the *regularization* phase, the model is leveraged to generate synthetic rollouts, that are then used to maximize the entropy of model predictions over the out-of-distribution data points. Note that the warm-up phase is essential to the next phase, since the additionally generated data from the model needs to be as close to the actual dynamics of the environment as possible, particularly, in the support of offline data and possibly its generalizable neighborhood.

3.2.1 WARM-UP PHASE

In this phase, we employ the Gaussian negative log-likelihood (NLL) loss to train an initial model $\hat{M}_{init}$ of the environment on the offline dataset. Let $\mathbb{B}_\mathcal{D}$ be a batch sampled from the offline data $\mathcal{D}$, the NLL objective function denoted by $\mathcal{L}_1$ can be written as in Equation 2

$$\mathcal{L}_1(\theta, \phi; \mathbb{B}_\mathcal{D}) = \frac{1}{|\mathbb{B}_\mathcal{D}|} \sum_{(s,a,r,s') \in \mathbb{B}_\mathcal{D}} \sum_{i=1}^{d+1} \frac{1}{2} \Big(\log(\Sigma_{\phi,i}(s, a)) + \frac{(\mu_{\theta,i}(s, a) - [s', r]_i)^2}{\Sigma_{\phi,i}(s, a)} \Big). \tag{2}$$

At the end of the warm-up phase, we expect $\mu_\theta(s, a)$ to provide accurate and reliable predictions for (s', r) conditioned on (s, a) in the support of the offline data, given that the training is complete.

Algorithm 1 Generating Rollouts

Require: $\mathcal{D}$, μ_θ, Σ_ϕ, π_e, batch size b, rollout horizon h, penalty coefficient λ

Set $\bar{\mathbb{B}}_{\pi_e} \leftarrow \emptyset$
for $1, 2, ..., b$ (in parallel) **do**
    Set $\rho \leftarrow \emptyset$
    Sample state s_1 from $\mathcal{D}$ for the initialization of the rollout.
    for $j = 1, 2, ..., h$ **do**
        Sample an action $a_j \sim \pi_e(s_j)$
        Obtain $s_{j+1}, r_j = \mu_\theta(s_j, a_j)$
        (Optional) Compute $\tilde{r}_j = r_j - \lambda\sqrt{\text{tr}(\Sigma_\phi(s_j, a_j)}$
        Add $(s_j, a_j, \tilde{r}_j, s_{j+1})$ to ρ
    end for
    Add ρ to $\bar{\mathbb{B}}_{\pi_e}$
end for
return $\bar{\mathbb{B}}_{\pi_e}$

Additionally, depending on the generalizability of the model, the performance can be extended to a neighborhood around the offline data distribution. Similarly, $\Sigma_\phi(s, a)$ will be able to capture the uncertainty in the areas that we have the support of offline data. However, no devoted argument can be made about the performance of $\mu_\theta(s, a)$ and $\Sigma_\phi(s, a)$ when queried about out-of-distribution (OOD) data points, and thus, they might predict arbitrary values in those regions.

3.2.2 Regularization Phase

If we directly employ the trained model from the warm-up phase as a surrogate for the actual environment and optimize a policy, the learned policy will most likely perform poorly in the real MDP (Levine et al., 2020). This happens because while training the policy, the RL algorithm will query the model in OOD data points as well, meaning that it will rely on potentially inaccurate predictions of the model, which might lead to inferior policies due to over- or under-estimation of the values. Nevertheless, this is not an inherent issue in standard model-based RL as the model can be improved over time by collecting more transitions, whereas in offline settings, the problem remains due to lack of additional interactions with the real environment. Therefore, we aim to find an efficient way to either prevent the policies from exploiting unreliable predictions, or prevent the unreliable predictions from affecting the values of state-action pairs.

To address this problem, we propose using a regularized objective function and a second phase for training the model. Once we have a preliminary empirical model, we expand the loss function to include a second objective to maximize the entropy of model predictions over the OOD domains. Since the model is characterized by a Gaussian distribution, entropy is a monotonically increasing function of $\det\big(\Sigma_\phi(s, a)\big) = \Sigma_i \Sigma_{\phi,i}(s, a)$ (equation 1). Hence, higher entropy, which translates to higher uncertainty, results in a higher value for $\det\big(\Sigma_\phi(s, a)\big)$, and vice versa. As a result, the maximized entropy, which can be attained by maximizing $\det\big(\Sigma_\phi(s, a)\big)$, provides an upper-bound estimation on the uncertainty of model predictions.

Let $\mu_\theta(s, a)$ and $\Sigma_\phi(s, a)$ be the mean and covariance of the Gaussian distribution over the next state and reward, i.e., $P(s', r|s, a) = \mathcal{N}\big(\mu_\theta(s, a), \Sigma_\phi(s, a)\big)$, and $\bar{\mathbb{B}}_{\pi_e}$ be a sample batch of rollouts generated from the trained dynamics $\mu_\theta(s, a)$ using an exploration policy $\pi_e(a|s)$, i.e., $(s', r) = \mu_\theta\big(s, a \sim \pi_e(a|s)\big)$, the second loss $\mathcal{L}_2$ is thus defined as

$$\mathcal{L}_2(\phi; \bar{\mathbb{B}}_{\pi_e}) = \frac{1}{|\bar{\mathbb{B}}_{\pi_e}|} \sum_{(s,a,r,s') \in \bar{\mathbb{B}}_{\pi_e}} \sum_{i=1}^{d+1} -\frac{1}{2} \log\big(\Sigma_{\phi,i}(s, a)\big), \tag{3}$$

which together with $\mathcal{L}_1$ introduced in Equation 2 yield the hybrid loss $\mathcal{L}$

$$\mathcal{L}(\theta, \phi; \mathbb{B}_\mathcal{D}, \bar{\mathbb{B}}_{\pi_e}) = \mathcal{L}_1(\theta, \phi; \mathbb{B}_\mathcal{D}) + \alpha \mathcal{L}_2(\phi; \bar{\mathbb{B}}_{\pi_e}), \tag{4}$$

and α is a fixed constant to control the effect of regularization on the NLL loss.

Algorithm 1 summarizes the procedure to generate a sample batch of b rollouts of length h given an offline dataset $\mathcal{D}$, a trained model in the form of $\mathcal{N}\big(\mu_\theta(s, a), \Sigma_\phi(s, a)\big)$, and an exploration policy

Algorithm 2 EMO

Require: Offline data $\mathcal{D}$, exploration policy π_e, batch size b, rollout horizon h, penalty coefficient λ, regularization coefficient α

Initialize θ and ϕ
for K_1 iterations **do** ▷ Warm-up phase
    Sample a batch of transitions $\mathbb{B}_{\mathcal{D}}$ from the offline dataset $\mathcal{D}$.
    Compute $\mathcal{L}_1(\theta, \phi; \mathbb{B}_{\mathcal{D}})$ (Equation 2.)
    Compute gradients and update θ and ϕ.
end for
for K_2 iterations **do** ▷ Regularization phase
    Sample a batch of transitions $\mathbb{B}_{\mathcal{D}}$ from the offline dataset $\mathcal{D}$.
    Compute $\mathcal{L}_1(\theta, \phi; \mathbb{B}_{\mathcal{D}})$ (Equation 2.)
    Generate a batch of transitions $\bar{\mathbb{B}}_{\pi_e}$ using model rollouts (Algorithm 1.)
    Compute $\mathcal{L}_2(\phi; \bar{\mathbb{B}}_{\pi_e})$ (Equation 3.)
    Compute $\mathcal{L}(\theta, \phi; \mathbb{B}_{\mathcal{D}}, \bar{\mathbb{B}}_{\pi_e}) = \mathcal{L}_1(\theta, \phi; \mathbb{B}_{\mathcal{D}}) + \alpha \mathcal{L}_2(\phi; \bar{\mathbb{B}}_{\pi_e})$.
    Compute gradients and update θ and ϕ.
end for

$\pi_e(a|s)$. In this Algorithm, $\mu_\theta(s, a)$ models the transition dynamics and $\Sigma_\phi(s, a)$ is used as the uncertainty estimator. We leverage an exploratory policy as well as the dynamics model, which is initially trained in the warm-up phase, to generate rollouts additional to samples from offline data to be employed in the regularized optimization problem. Note that the dynamics model will be gradually updated as the training progresses. Accordingly, Algorithm 2 plots the overall learning procedure of EMO, in which the batches of data from both the offline dataset as well as the generated rollouts from the model are utilized to minimize the hybrid loss in Equation 4.

In the regularization phase, the model is trained to maximize the likelihood of the offline data, collected by a (possibly unknown) behavior policy, while maximizing the entropy of predictions over the distribution induced by an exploratory policy. Hence, $\mathcal{L}_1$ ensures that μ_θ and Σ_ϕ maintain their accuracy in the support of offline data, while $\mathcal{L}_2$ aims to increase Σ_ϕ as we leave the support of offline dataset (over the distribution induced by the exploration policy π_e.) As a result, $\Sigma_\phi(s, a)$ can be used as an upper bound indicating how reliable/accurate the trained model is for a certain pair of s and a, as long as the distribution of rollouts generated by the exploration policy covers this particular state-action pair. In other words, as long as the distribution of rollouts generated from π_e covers the potential distributions of other exploratory policies, (which will be used later during policy optimization in Section 3.3,) $\Sigma_\phi(s, a)$ can be leveraged as the error estimator to relatively penalize unreliable predictions.

Furthermore, by considering a small value for α, we ensure that the effect of $\mathcal{L}_2$ will be negligible compared to $\mathcal{L}_1$ where the distribution of offline data $\mathcal{D}$ overlaps with the distribution of rollouts under the exploratory policy π_e. In this way, the performance of the model is practically unaffected in the support of offline data and possibly other generalizable neighborhoods. Note that $\mathcal{L}_2$ will still be effective in the OOD areas, since $\mathcal{L}_1$ is non-existent in those regions.

Algorithm 3 General Framework for Model-based Offline RL

Require: Offline dataset $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^n$; penalty coefficient λ.

Train the dynamics model μ_θ and admissible uncertainty estimator Σ_ϕ using $\mathcal{D}$. (Algorithm 2)
Define $u(s, a) = \sqrt{\det\left(\Sigma_\phi(s, a)\right)}$
Define empirical MDP $\hat{M}$ with dynamics μ_θ and reward $\tilde{r}(s, a) = r(s, a) - \lambda u(s, a)$.
Run any RL algorithm on $\hat{M}$ until convergence to obtain $\hat{\pi}^* = \arg\max_{\pi} \eta_{\hat{M}}(\pi)$.

3.3 Policy Learning

Once the model is trained using EMO, we will utilize its components to define a pessimistic MDP, which can be coupled with any policy optimization technique to obtain the output policy $\hat{\pi}^*$. Sub-

sequently, the overall learning framework, which also adheres to the prior work (Yu et al., 2020; Kidambi et al., 2020), is summarized in Algorithm 3. In this algorithm, the offline dataset $\mathcal{D}$ is first used to train a dynamics model μ_θ along with an admissible error/uncertainty estimator Σ_ϕ. Next, a new pessimistic MDP is defined as $\hat{M} = (\mathcal{S}, \mathcal{A}, T_\mu, \tilde{r}, \gamma, \rho_0)$, with $T_\mu = \{\mu_{\theta,i}(s,a)\}_{i=1}^{d}$ and $\tilde{r}(s,a) = r(s,a) - \lambda u(s,a)$, where $u(s,a) = \sqrt{\text{tr}(\Sigma_\phi(s,a))}$ (see section 3.2). Lastly, the pessimistic MDP $\hat{M}$ is leveraged as a surrogate model to train an RL algorithm to obtain $\hat{\pi}^*$.

3.4 Theoretical Grounds of EMO

We argue that EMO is an extension to MOPO (Yu et al., 2020), while addresses the limitations of MOPO regarding learning an ensemble of models for uncertainty estimation. Accordingly, the theoretical grounds of MOPO can be extended to EMO as well. We reiterate the theoretical formulation and analysis of MOPO in Appendix A.1. However, for a more exhaustive discussion on the theory, we refer the reader to Yu et al. (2020).

4 Empirical Study

In this section, we evaluate the performance of our approach on D4RL benchmark datasets for MuJoCo environments (Fu et al., 2020). We include data from three different environments: halcheetah, hopper, and walker2d, and four different types from each, i.e., random, medium, medium-replay, and medium-expert, which results in twelve different datasets. Throughout the experiments, we aim to investigate the following questions: (1) How does EMO perform compared to state-of-the-art methods on standard offline RL benchmark? (2) What impact do entropy regularization and reward penalty have on the performance of the trained policies? (3) What is the effect of exploration policy π_e on the generalization ability of the trained models and the performance of the resulting policies? (4) What is the effect of regularization coefficient α on the performance of EMO?

4.1 Experimental Setups

Following the setup in Yu et al. (2020), we characterize the model as a 4-layer feed-forward neural network across all domains, with 200 hidden units in each layer. Subsequently, the output of the last hidden layer is fed into a two-head network architecture to generate $\mu_\theta(s,a)$ and $\Sigma_\phi(s,a)$. Instead of directly estimating the reward function, the model predicts the center of mass velocity, and the reward is calculated afterwards based on its formulation in each domain.

For all the experiments, a soft actor-critic (SAC) agent[1] (Haarnoja et al., 2018) is used as the reinforcement learning agent for policy optimization on the trained pessimistic MDP $\hat{M}$. At each time step, a batch of k-step rollouts are generated and added to the replay memory of the SAC agent, where the actions for generating the rollouts are taken based on the current policy of the agent, while transitions and rewards are produced by $\hat{M}$. Next, the agent optimizes its policy based on samples from both its replay memory and the offline data $\mathcal{D}$. Note that in our implementation, the agent will only utilize its own replay memory for policy optimization, meaning that samples from offline data are not directly utilized for policy optimization (see algorithm 5 in appendix A.2). Consequently, the resulting policies from SAC are evaluated in the real MuJoCo environment for testing.

4.2 Overall performance

We compare the performance of EMO to the state-of-the-art model-based offline algorithms, i.e., MOPO (Yu et al., 2020), MOReL (Kidambi et al., 2020), COMBO (Yu et al., 2021), and RAMBO-RL (Rigter et al., 2022), as well as two best-performing model-free counterparts, i.e., UWAC (Wu et al., 2021) and CQL (Kumar et al., 2020). The performance results of both model-based and model-free techniques are summarized in Table 1. All the presented scores are normalized according to the procedure proposed in Fu et al. (2020). For EMO, the results are the performance of policy at the last iteration of training, averaged over 3 random seeds $\pm$ standard deviation. The values for RAMBO-RL, COMBO, UWAC, MOPO , and MOReL are reported from their respective papers. The results for CQL are taken from D4RL benchmark white-paper (Fu et al., 2020).

[1] https://github.com/pranz24/pytorch-soft-actor-critic

Table 1: Performance results of offline RL algorithms on D4RL benchmark datasets.

Dataset	Environment	EMO (Ours)	RAMBO-RL	COMBO	MOPO	MOReL	UWAC	CQL
random	halfcheetah	36.3 $\pm$ 2.9	**39.5** $\pm$ 3.5	38.8 $\pm$ 3.7	35.4 $\pm$ 2.5	25.6	14.5 $\pm$ 3.3	35.4
random	hopper	31.6 $\pm$ 0.2	25.4 $\pm$ 7.5	17.9 $\pm$ 1.4	11.7 $\pm$ 0.4	**53.6**	22.4 $\pm$ 12.1	10.8
random	walker2d	5.3 $\pm$ 5.1	0.0 $\pm$ 0.3	7.0 $\pm$ 3.6	13.6 $\pm$ 2.6	**37.3**	15.5 $\pm$ 11.7	7.0
medium	halfcheetah	68.5 $\pm$ 1.6	**77.9** $\pm$ 3.4	54.2 $\pm$ 1.5	42.3 $\pm$ 1.6	42.1	46.5 $\pm$ 2.5	44.4
medium	hopper	36.5 $\pm$ 43.5	87.0 $\pm$ 15.4	**97.2** $\pm$ 2.2	28.0 $\pm$ 12.4	95.4	88.9 $\pm$ 12.2	86.6
medium	walker2d	**84.1** $\pm$ 4.3	**84.9** $\pm$ 2.6	81.9 $\pm$ 2.8	17.8 $\pm$ 19.3	77.8	57.5 $\pm$ 7.8	79.2
medium-replay	halfcheetah	56.7 $\pm$ 10.2	**68.7** $\pm$ 5.3	55.1 $\pm$ 1.0	53.1 $\pm$ 2.0	40.2	46.8 $\pm$ 3.0	46.2
medium-replay	hopper	77.4 $\pm$ 29.3	**99.5** $\pm$ 4.8	89.5 $\pm$ 1.8	67.5 $\pm$ 24.7	**93.6**	39.4 $\pm$ 6.1	48.6
medium-replay	walker2d	**83.6** $\pm$ 2.2	**89.2** $\pm$ 6.7	56.0 $\pm$ 8.6	39.0 $\pm$ 9.6	49.8	27.0 $\pm$ 6.3	26.7
medium-expert	halfcheetah	80.2 $\pm$ 6.8	**95.4** $\pm$ 5.4	90.0 $\pm$ 5.6	63.3 $\pm$ 38.0	53.3	**127.4** $\pm$ 3.7	62.4
medium-expert	hopper	92.1 $\pm$ 4.8	88.2 $\pm$ 20.5	111.1 $\pm$ 2.9	23.7 $\pm$ 6.0	108.7	**134.7** $\pm$ 21.2	111
medium-expert	walker2d	**110.8** $\pm$ 5.2	56.7 $\pm$ 39.0	103.3 $\pm$ 5.6	44.6 $\pm$ 12.9	95.6	99.7 $\pm$ 12.2	98.7

The outlined results in Table 1 demonstrate that EMO outperforms MOPO in almost all cases (except for `walker2d-random`), which shows that EMO, as an extension to MOPO, clearly improves upon its predecessor by replacing ensemble uncertainty quantification with entropy regularization. Furthermore, EMO achieves competitive results compared to COMBO and MOReL, outperforming both on 5 out of 12 datasets, which places EMO among the highest-performing model-based algorithms. Although EMO can achieve comparable results to RAMBO-RL in certain scenarios, its performance falls short in some cases, and only outperforms RAMBO-RL on one dataset (i.e., `walker2d`). This can be attributed to the fact that RAMBO-RL is a task-specific method, which tunes the model to be pessimistic with respect to the current policy of the agent, while EMO utilizes a general purpose, task-agnostic model of the environment for policy optimization. In addition, bear in mind that EMO achieves this level of performance using only a single model of the environment, while an ensemble of models is utilized in other model-based methods. Moreover, Table 1 illustrates that EMO outperforms the model-free techniques in 8 out of 12 datasets, which highlights the effectiveness of learning a pessimistic MDP in improving the sample efficiency.

Table 2: Performance results of policy optimization on different model configurations

Dataset	Environment	EMO	NLL+SAC ($\lambda \neq 0$)	NLL+SAC ($\lambda = 0$)
medium	halfcheetah	**68.5** $\pm$ 1.6	35.1 $\pm$ 13.8	0.9 $\pm$ 1.5
medium	walker2d	**84.1** $\pm$ 4.3	13.0 $\pm$ 22.6	5.6 $\pm$ 8.8
medium-replay	halfcheetah	**56.7** $\pm$ 10.2	4.0 $\pm$ 6.6	10.8 $\pm$ 2.8
medium-replay	walker2d	**83.6** $\pm$ 2.2	18.7 $\pm$ 10.1	6.4 $\pm$ 2.1

4.3 Ablation Studies

In order to address question (2), we conduct an experiment to compare the performance of EMO with two simplified versions of the algorithm: (i) training the policy on a model that is only trained via the NLL loss (i.e., the resulting model from the warm-up phase) *without* utilizing the reward penalty ($\lambda = 0$), denoted by **NLL+SAC** ($\lambda = 0$), and (ii) the same model *with* the reward penalty ($\lambda \neq 0$) leveraging the covariance matrix from the warm-start phase, indicated by **NLL+SAC** ($\lambda \neq 0$). The outcomes of the experiment are outlined in table 2 on 4 different datasets. The results confirm that although penalizing the reward values in OOD areas improves the performance of the model, it is still insufficient to achieve comparable results to state-of-the-arts. The reason lies in the fact that the penalties depend on the arbitrary predictions of the covariance matrix in the OOD domains. However, by incorporating the entropy regularization step, we ensure that reward penalties will be proportional to the associated uncertainty of the predictions, which leads to a considerable gap between the performance of the resulting policies. This finding empirically validates the impact of the entropy regularization coupled with penalizing the reward values in uncertain regions.

In the second ablation study, we investigate the impact of using an informed exploration policy vs. a random policy to address question (3). One could argue that for a specific task, utilizing an informed policy (i.e., the current policy of the agent) instead of a random policy can be beneficial. In this case, the entropy regularization phase can be guided more efficiently toward the areas that are more likely to be explored by the learning agent. Subsequently, the coverage of the rollouts generated under an informed policy is expected to be a small subset of the rollouts generated from a random policy.

Table 3: Performance results of policy optimization for random vs. informed exploration policy

Dataset	Environment	EMO	Modified EMO	RAMBO-RL
medium-replay	halfcheetah	56.7 $\pm$ 10.2	**61.1** $\pm$ 0.7	**68.7** $\pm$ 5.3
medium-replay	walker2d	83.6 $\pm$ 2.2	**85.5** $\pm$ 6.1	**89.2** $\pm$ 6.7

Consequently, we propose a modified version of EMO, summarized in algorithm 4 in appendix A.2, in which the (SAC) agent and the model (only the regularization phase) are trained simultaneously, rather than modular. In this variant, the model utilizes the current policy of the agent as the exploration policy for entropy regularization in order to benefit from an informed exploration scheme. We further examine this modification on two offline datasets, namely `halfcheetah-medium-replay` and `walker2d-medium-replay`, compared to RAMBO-RL. The summarized results in table 3 indicate that informed exploration can indeed improve the performance of trained policies. Such improvements in the performance are expected since the modified version of EMO has a more specialized, task-specific approach for model optimization compared to the original version. Furthermore, although EMO can achieve comparable results compared to RAMBO-RL and even outperform it on one set of data (see Table 1), the modified version can tighten the gap and make the results more competitive to RAMBO-RL. However, we still prioritize the original EMO over the modified one to have a general purpose, task-agnostic pessimistic model of the environment, which can be employed in any modular framework with any RL algorithm for training policies. Whereas this cannot be achieved using the modified version of EMO or methods such as RAMBO-RL.

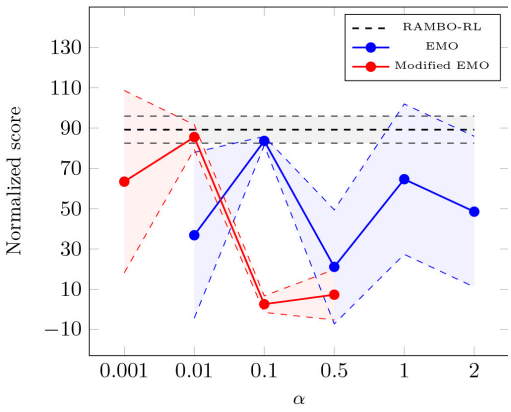

Figure 2: Performance of EMO variants in terms of α on `walker2d-medium-replay`.

To answer question (4), we demonstrate the performance of EMO and modified EMO for different configurations of the regularization coefficient α on `walker2d-medium-replay` dataset in figure 2. The figure illustrates how changing the value of α affects the performance of both methods, also compared to RAMBO-RL. As we increase α, the regularization term in equation 4 becomes more dominant, leading to algorithms become more conservative, and vice versa. As a result, smaller values of α are preferred to achieve best performance when offline data has limited coverage of the state-action space or contains less informative data. Conversely, when the offline data provides a better coverage, higher values for α are preferred. In addition, since the datastes from the `medium-replay` category are considerably smaller than other D4RL benchmark data (aka limited coverage), both EMO and modified EMO are expected to perform better when α is set to smaller numbers, as shown in figure 2. However, if α is too small, then the algorithms will not regularize very well, and that can lead a decrease in performance as well. The same can be said when the algorithm becomes too conservative, which is the case when α is set too large.

5 Conclusions

In this paper, we presented EMO, an entropy-regularized optimization algorithm to learn a pessimistic MDP for model-based offline RL problems. In this framework, we devised a hybrid loss function to minimize the negative log-likelihood of the model on the distribution of the offline data while maximizing the entropy in the areas that the support of data is none or minimal. We thus optimized both objectives in a single neural network architecture rather than an ensemble of K models compared to the state-of-the-art model-based approaches. Moreover, our empirical study on D4RL benchmark data showed that our approach performs competitively compared to state-of-the-art model-based as well as model-free offline RL techniques.

6 REPRODUCIBILITY STATEMENT

To ensure the reproducibility of the results, the codes are provided in the supplementary materials. Configurations and the choice of hyperparameters are also included in the supplementary materials in `readme.txt` file. To facilitate the understanding of theoretical formulation and practical implementation, additional algorithms and theories are also included in the appendix.

REFERENCES

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pp. 104–114. PMLR, 2020.

Xiong-Hui Chen, Yang Yu, Qingyang Li, Fan-Ming Luo, Zhiwei Qin, Wenjie Shang, and Jieping Ye. Offline model-based adaptable policy learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

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

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pp. 2052–2062. PMLR, 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/haarnoja18b.html`.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/5faf461eff3099671ad63c6f3f094f7f-Paper.pdf`.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 21810–21823. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/f7efa4f864ae9b88d43527f4b14f750f-Paper.pdf`.

Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*, pp. 5774–5783. PMLR, 2021.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pp. 45–73. Springer, 2012.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020. URL https://arxiv.org/abs/2005.01643.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Alfred Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997. ISSN 00018678. URL http://www.jstor.org/stable/1428011.

Marc Rigter, Bruno Lacerda, and Nick Hawes. Rambo-rl: Robust adversarial model-based offline reinforcement learning, 2022. URL https://arxiv.org/abs/2204.12581.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Phillip Swazinna, Steffen Udluft, and Thomas Runkler. Overcoming model bias for robust offline deep reinforcement learning. *Engineering Applications of Artificial Intelligence*, 104:104366, 2021.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Yue Wu, Shuangfei Zhai, Nitish Srivastava, Joshua Susskind, Jian Zhang, Ruslan Salakhutdinov, and Hanlin Goh. Uncertainty weighted actor-critic for offline reinforcement learning. *arXiv preprint arXiv:2105.08140*, 2021.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 14129–14142. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/a322852ce0df73e204b7e67cbbef0d0a-Paper.pdf.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in Neural Information Processing Systems*, 34, 2021.

Xianyuan Zhan, Xiangyu Zhu, and Haoran Xu. Model-based offline planning with trajectory pruning. *arXiv preprint arXiv:2105.07351*, 2021.

A Appendix

A.1 Theoretical Grounds of EMO

As discussed in section 3.4, EMO can be considered as an extension to MOPO (Yu et al., 2020), with the aim to address the limitations of ensemble uncertainty estimation. Accordingly, the theoretical grounds of MOPO can be extended to EMO as well. In this section, we briefly reiterate the theoretical formulation of MOPO. However, we encourage the reader to refer to Yu et al. (2020) for a more detailed explanation.

A.1.1 PRELIMINARIES

To better understand the remainder of this section, we first introduce a series of required notations. Assume having an MDP $M = (\mathcal{S}, \mathcal{A}, T, r, \gamma, \rho_0)$, and an arbitrary policy π. The value function, $V_M^\pi(s) = \mathbb{E}_{\pi,T}[\Sigma_{t=0}^{\infty}\gamma^t r(s_t, a_t)|s_0 = s]$, is regarded as the value of a particular state s under the policy π, which is defined as the expected discounted return under π when starting from s. $\mathbb{P}_{T,t}^\pi(s)$ is considered as the probability of being in state s at time step t if actions are sampled according to π and transitions according to T; and $\rho_T^\pi(s,a)$ will be the discounted occupancy measure of policy π under dynamics T: $\rho_T^\pi(s,a) := \pi(a|s)\Sigma_{t=0}^{\infty}\gamma^t\mathbb{P}_{T,t}^\pi(s)$. Since ρ_T^π is not a properly normalized probability distribution, (improper) expectations with respect to ρ_T^π will be denoted by $\bar{\mathbb{E}}$.

A.1.2 THEORETICAL FORMULATION

The main idea is to define a pessimistic MDP (P-MDP) $\hat{M}$, such that the performance (i.e. the return) of a policy π under the actual MDP is lower bounded by its performance under the defined P-MDP, and proceed with optimizing a policy on the defined P-MDP, which translates to optimizing the lower bound on the actual return of the policy. First, we start by quantifying the relationship between the performance of a policy under two different MDPs with same characteristics, but different dynamics T and $\hat{T}$.

Lemma. (Telescoping lemma) *Let M and $\hat{M}$ be two MDPs with the same reward function r, but different dynamics T and $\hat{T}$ respectively. Let $G_{\hat{M}}^\pi(s,a) := \mathbb{E}_{s'\sim\hat{T}(s,a)}[V_M^\pi(s')] - \mathbb{E}_{s'\sim T(s,a)}[V_M^\pi(s')]$. Then,*

$$\eta_{\hat{M}}(\pi) - \eta_M(\pi) = \gamma\bar{\mathbb{E}}_{(s,a)\sim\rho_{\hat{T}}^\pi}[G_{\hat{M}}^\pi(s,a)] \tag{5}$$

Consequently, we have

$$\eta_M(\pi) = \bar{\mathbb{E}}_{(s,a)\sim\rho_{\hat{T}}^\pi}[r(s,a) - \gamma G_{\hat{M}}^\pi(s,a)] \geq \bar{\mathbb{E}}_{(s,a)\sim\rho_{\hat{T}}^\pi}[r(s,a) - \gamma|G_{\hat{M}}^\pi(s,a)|] \tag{6}$$

If we consider T and $\hat{T}$ as the true and learned dynamics of the same environment respectively, we can see that a lower bound on $\eta_M(\pi)$ can be established in the form of 6. However, since $G_{\hat{M}}^\pi(s,a)$ is a function of V_M^π (which is an unknown function by itself), the RHS of 6 cannot be accurately computed. To eliminate the reliance on V_M^π, Yu et al. (2020) propose to use an upper bound of $G_{\hat{M}}^\pi(s,a)$ instead.

Assumption 1 *Assume a scalar c and a function class $\mathcal{F}$ such that $V_M^\pi \in c\mathcal{F}$ for all π.*

If $\mathcal{F}$ is a set of functions mapping $\mathcal{S}$ to $\mathbb{R}$ that contains V_M^π, then,

$$|G_{\hat{M}}^\pi(s,a)| \leq sup_{f\in\mathcal{F}}|\mathbb{E}_{s'\sim\hat{T}(s,a)}[f(s')] - \mathbb{E}_{s'\sim T(s,a)}[f(s')]| =: d_{\mathcal{F}}(\hat{T}(s,a), T(s,a)) \tag{7}$$

where $d_{\mathcal{F}}$ is the integral probability metric (IPM) defined by $\mathcal{F}$ (Müller, 1997). Based upon assumption 1 and equation (7), we have

$$|G_{\hat{M}}^\pi(s,a)| \leq cd_{\mathcal{F}}(\hat{T}(s,a), T(s,a)) \tag{8}$$

Now that we have an upper bound for $|G_{\hat{M}}^\pi(s,a)|$ that solely depends on $d_{\mathcal{F}}(\hat{T}(s,a), T(s,a))$, which is the error of the model, we assume that we have access to an uncertainty quantification module that provides an upper bound on the aforementioned error.

Assumption 2 *Let $\mathcal{F}$ be the function class in assumption 1. We say $u : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ is an admissible error estimator for $\hat{T}$ if $d_{\mathcal{F}}(\hat{T}(s,a), T(s,a)) \leq u(s,a)$ for all $s \in \mathcal{S}, a \in \mathcal{A}$.*

If we assume having access to an error estimator module defined in assumption 2, we can define a *pessimistic MDP* in the form of $\tilde{M} = (\mathcal{S}, \mathcal{A}, \hat{T}, \tilde{r}, \gamma, \rho_0)$, where $\tilde{r}(s,a) := r(s,a) - \lambda u(s,a)$ and $\lambda := \gamma c$. We observe that the return of a policy under $\tilde{M}$ bounds the true return from below:

$$\eta_M(\pi) \geq \bar{\mathbb{E}}_{(s,a)\sim\rho^{\pi}_{\hat{T}}}[r(s,a) - \gamma|G^{\pi}_{\hat{M}}(s,a)|] \geq \bar{\mathbb{E}}_{(s,a)\sim\rho^{\pi}_{\hat{T}}}[r(s,a) - \lambda u(s,a)] \\ \geq \bar{\mathbb{E}}_{(s,a)\sim\rho^{\pi}_{\hat{T}}}[\tilde{r}(s,a)] = \eta_{\tilde{M}}(\pi) \quad (9)$$

We proceed with investigating the optimality of the learned policy. Let π^* be the optimal policy on M, i.e. $\pi^* = \arg\max_{\pi} \eta_M(\pi)$, and π^B be the behavior policy, i.e. the policy that has generated the offline data. Define $\epsilon_u(\pi)$ as

$$\epsilon_u(\pi) := \bar{\mathbb{E}}_{(s,a)\sim\rho^{\pi}_{\hat{T}}}[u(s,a)] \quad (10)$$

for $\delta \geq \delta_{min} := \min_{\pi} \epsilon_u(\pi)$, let π^{δ} be the best policy among those that their $\epsilon_u(\pi)$ is capped by δ:

$$\pi^{\delta} := \arg\max_{\pi:\epsilon_u(\pi)\leq\delta} \eta_M(\pi) \quad (11)$$

The main theorem provides a performance guarantee on the policy $\hat{\pi}$ produced by MOPO.

Theorem 1 *Under assumptions 1 and 2, the learned policy $\hat{\pi}$ in MOPO satisfies*

$$\eta_M(\hat{\pi}) \geq \sup_{\pi}\{\eta_M(\pi) - 2\lambda\epsilon_u(\pi)\} \quad (12)$$

In particular, for all $\delta \geq \delta_{min}$,

$$\eta_M(\hat{\pi}) \geq \eta_M(\pi^{\delta}) - 2\lambda\delta \quad (13)$$

In practice, MOPO trains an ensemble of K dynamics models, where each model learns a Gaussian distribution over the next state and reward, conditioned on the current state and action $\{\hat{T}^i_{\theta,\phi}(s', r|s,a) = \mathcal{N}(\mu^i_{\theta}(s,a), \Sigma^i_{\phi}(s,a))\}_{i=1}^{K}$, and the maximum variance within the ensemble elements is leveraged as the admissible error estimator, i.e. $u(s,a) = \max_{i=1}^{K} ||\Sigma^i_{\phi}(s,a)||_F$. While MOPO provides promising results on standard benchmark datasets, its performance can be limited in some scenarios due to how ensembles are practically implemented, and how their characteristics can be different than what is theoretically assumed, as discussed in sections 1 and 2. In EMO, however, we use a single model and try to directly optimize an upper bound on the uncertainty/error of predictions (i.e. $u(s,a)$), which eliminates the need for ensemble uncertainty quantfication, with the aim to address the limitations of MOPO associated with utilizing an ensemble of models to compute $u(s,a)$.

A.2 REFERENCED ALGORITHMS

Algorithm 4 Modified Version of EMO

Require: Offline data $\mathcal{D}$, batch size b, rollout horizon h, penalty coefficient λ, regularization coefficient α

Initialize θ and ϕ
Initialize policy π
for K_1 iterations **do** ▷ Warm-up phase
  Sample a batch of transitions $\mathbb{B}_{\mathcal{D}}$ from the offline dataset $\mathcal{D}$.
  Compute $\mathcal{L}_1(\theta, \phi; \mathbb{B}_{\mathcal{D}})$ (Equation 2.)
  Compute gradients and update θ and ϕ.
end for
for K_2 iterations **do**
  for K_3 iterations **do**
    Sample a batch of transitions $\mathbb{B}_{\mathcal{D}}$ from the offline dataset $\mathcal{D}$.
    Compute $\mathcal{L}_1(\theta, \phi; \mathbb{B}_{\mathcal{D}})$ (Equation 2.)
    Generate a batch of transitions $\bar{\mathbb{B}}_{\pi}$ using model rollouts (Algorithm 1.)
    Compute $\mathcal{L}_2(\phi; \bar{\mathbb{B}}_{\pi_e})$ (Equation 3.)
    Compute $\mathcal{L}(\theta, \phi; \mathbb{B}_{\mathcal{D}}, \bar{\mathbb{B}}_{\pi_e}) = \mathcal{L}_1(\theta, \phi; \mathbb{B}_{\mathcal{D}}) + \alpha \mathcal{L}_2(\phi; \bar{\mathbb{B}}_{\pi_e})$.
    Compute gradients and update θ and ϕ.
  end for
  Define empirical MDP $\hat{M}$ with dynamics μ_θ and reward $\tilde{r}(s, a) = r(s, a) - \lambda u(s, a)$, where $u(s, a) = \sqrt{\det\left(\Sigma_\phi(s, a)\right)}$.
  for K_4 iterations **do**
    Update π on $\hat{M}$ with any arbitrary RL algorithm.
  end for
end for
return π

Algorithm 5 Policy Optimization Method for Experiments

Require: pessimistic MDP $\hat{M} = (\mathcal{S}, \mathcal{A}, T_\mu, \tilde{r}, \gamma, \rho_0)$, rollout batch size b, rollout horizon h, offline dataset $\mathcal{D}$

Initialize policy π and empty replay buffer $\mathcal{D}_{model} \leftarrow \emptyset$
for K iterations **do**
  for $1, 2, ..., b$ (in parallel) **do**
    Sample the initial state s_1 of the rollout by sampling from offline data $\mathcal{D}$.
    for $j = 1, 2, ..., h$ **do**
      Sample an action based on the current policy of the agent $a_j \sim \pi(s_j)$.
      Compute the next state in the pessimistic MDP $s'_j = T_\mu(s_j, a_j)$.
      Compute the reward in the pessimistic MDP $\tilde{r}_j = \tilde{r}(s_j, a_j)$.
      Add transition $(s_j, a_j, \tilde{r}_j, s'_j)$ to $\mathcal{D}_{model}$.
    end for
  end for
  Draw samples from $\mathcal{D} \cup \mathcal{D}_{model}$, use SAC to update π.
end for