DDA6105 Lecture 18
Offline Reinforcement Learning

Yingru Li

yingruli@link.cuhk.edu.cn
The Chinese University of Hong Kong, Shenzhen, China

December 2, 2020

Data and Decision Analytics (DDA) 6105 Reinforcement Learning, Instructor: Prof. Xinyun Chen & Prof. Jim Dai

Outline

Outline

Current RL success paradigm

- RL algorithms can learn complex behaviors in simulation, where active (on-policy) data collection is straightforward.

Figure: Go and Game: 'good simulator ≈ infinite accessible data with almost no expense as long as the computation resources is provided'

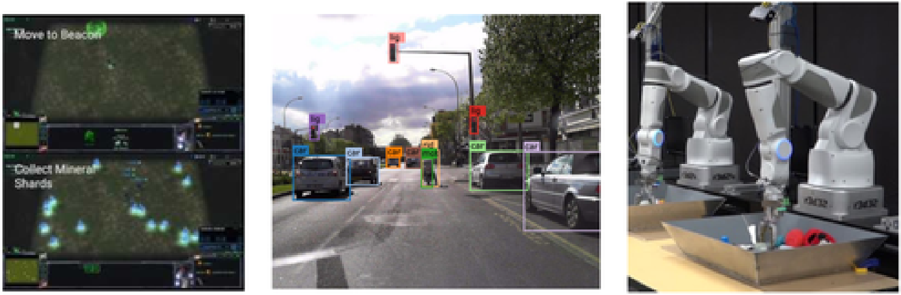
Motivation: the real-world applications

- In real-world applications, the performance is limited by the expense of active data collection.
  - Deploying a policy to collect new data is costly. (E.g. Recommendation systems, DiDi/Uber.)
  - Safety concern with updating/executing the policy online. (E.g. Robotic control, Healthcare applications, autonomous driving, communication networks.)
- Deploying a new policy may only be done at a **low frequency** after extensive testing and evaluation.
- Good news:
  - In some of these cases, the offline dataset are often very large, potentially encompassing years of logged experience. (**Our focus today**)
  - We can build good simulators based on some specific applications and try to transfer what we learn in simulators to real environments. (Sim2Real)

Motivation: the real-world applications

- In real-world applications, the performance is limited by the expense of active data collection.
  - Deploying a policy to collect new data is costly. (E.g. Recommendation systems, DiDi/Uber.)
  - Safety concern with updating/executing the policy online. (E.g. Robotic control, Healthcare applications, autonomous driving, communication networks.)
- Deploying a new policy may only be done at a **low frequency** after extensive testing and evaluation.
- Good news:
  - In some of these cases, the offline dataset are often very large, potentially encompassing years of logged experience. (**Our focus today**)
  - We can build good simulators based on some specific applications and try to transfer what we learn in simulators to real environments. (Sim2Real)

Offline Datasets



Starcraft Replays (1M) | Self-driving cars (1100h) | Robotic Grasping (1M)

Figure: We want to make use of these fixed and static offline datasets when doing RL as environment interaction is (often) costly and even dangerous.

Outline

Introduction

- Motivation
- Offline RL Problems

Methods

- Model-free policy constraint based algorithm
- Model-based uncertainty-aware methods

Offline Reinforcement Learning

Fundamental Question:

How to effectively utilize offline datasets for future decisions while the agents are not able to interact with the environment to gather new data?

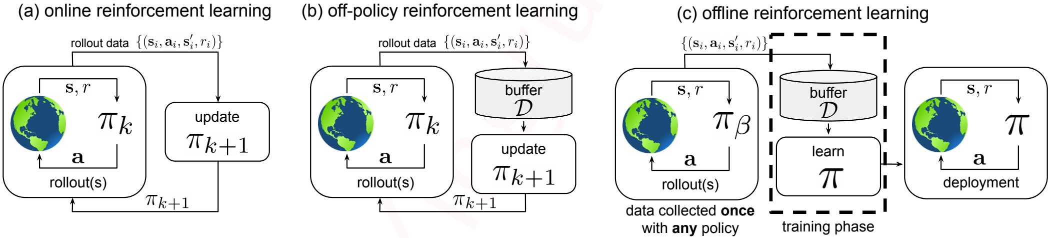


Figure: Pictorial illustration of classic online reinforcement learning (a), classic off-policy reinforcement learning (b), and offline reinforcement learning (c). In online reinforcement learning. (Figure from [Levine, Kumar, Tucker, and Fu, 2020])

Offline Reinforcement Learning

Fundamental Question:

How to effectively utilize offline datasets for future decisions while the agents are not able to interact with the environment to gather new data?

- Learning and generalizing by incorporating diverse historical experience without further trial and errors,
  - Not just imitating historical experience. (Introspective Intelligence)
- Better sample efficiency.

Offline RL Problematics (I)

Insufficient coverage and Distributional shift

- **Fixed under-explored offline dataset**: dataset without enough exploration often cannot cover enough states and actions.
- Even for **tabular setting**, there is no guarantee that the optimal policy can be found using the under-explored dataset.
  - Not possible to find optimal policy with little data coverage on the state-action region that optimal policy frequently visits.
- Problems with large or continuous state and action spaces require **function approximation** to generalize across states and actions.
  - Under-explored data will lead to erroneous generalization of the function for state-action pairs in under-explored region.

Offline RL Problematics (II)

Extrapolation error from distributional shift

- Problems with large or continuous state and action spaces require function approximation.
- Erroneous generalization/extrapolation error of the state-action value function (Q-value function) learned with function approximators leads to high bootstrapping error. [Kumar et al., 2019, Wu et al., 2019]

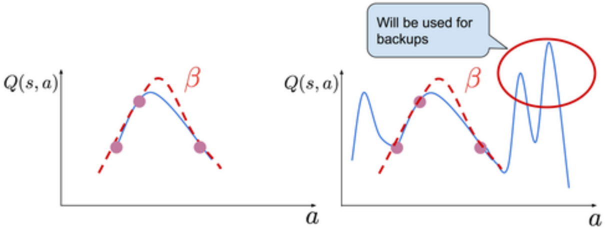


Figure: Incorrectly high Q-values for OOD actions may be used for backups, leading to accumulation of error.

Offline RL Problematics (III)

Boostrapping Error

- Suppose the offline dataset is collected by the behavior policy $\pi_\beta(a|s)$ (possibly multiple).
- For one transition tuple collected by behaviour policy $\pi_\beta(a|s)$ with policy induced state-action distribution $\beta(s,a)$:

$$(s, a, s') \sim \beta(s,a) P(s'|s,a)$$

- Illustration via Q value iteration:

$$\underbrace{Q^{k+1}(s,a)}_{\text{Errors accumulated into } Q(s,a)} \leftarrow r(s,a) + \gamma \underbrace{\max_{a'} Q^k(s',a')}_{\text{usually query at unseen } a'}$$

  - $Q(s',a')$ for $s' \nsim \beta$: Out-of-distribution (OoD) state
  - $Q(s',a')$ for $s' \sim \beta$, a' far from $\pi_\beta(a'|s')$: OoD action.

Offline RL Problematics (IV)

Error Propagation

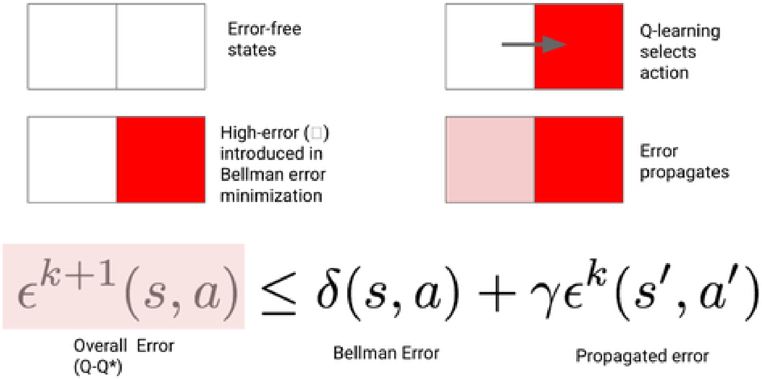


Figure: Error Propagation: Figure from [Kumar, Fu, Soh, Tucker, and Levine, 2019]

Offline RL Problematics Lead to Wrong Behavior Consequences

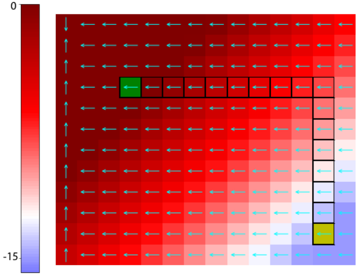


Figure: Learning goal-reaching policy from offline dataset $\mathcal{D}$. **Wrongly linear extrapolation!** (Figure from [Luo et al., 2019])

- Reward = -1 if not reaching the goal
- V^* = - minkovski distance to goal
- Learned (linear) value function
  - Correct within the support of offline dataset $\mathcal{D}$
  - Wrong outside the support
- Resulting wrong behavior induced from learned value
- Conclusions: Learning from $\mathcal{D}$ only guarantees accurate predictions on the offline data distribution
  - e.g. Q-learning with $\mathcal{D}$ results over-estimation outside the support of $\mathcal{D}$.

Outline

Overview of representative algorithms for Offline RL

- **Policy based constraints/penalities**: MARWIL[Wang, Xiong, Han, Liu, Zhang, et al., 2018], BCQ[Fujimoto et al., 2019] BRAC[Wu et al., 2019], BEAR[Kumar et al., 2019], AWAC[Nair et al., 2020], EMaQ [Ghasemipour et al., 2020].
- **Model based uncertainty-aware penalization**: MoRel[Kidambi, Rajeswaran, Netrapalli, and Joachims, 2020], MOPO[Yu et al., 2020]
- MBS-PI/MBS-QI [Liu et al., 2020]: **Filter out infrequent state-action pairs** in the offline dataset when performing policy based or value based model free methods:
- Conservative Q-learning [Kumar et al., 2020] consider regularization on fitted Q learning which provides the **value lower bound under fixed policy**. (Less conservative than the pointwise lower bound Q.)

Typical algorithmic framework: Actor-critic

- Given a dataset $\mathcal{D} = \{(s, a, r, s')\}$ of tuples from trajectories collected under π_β :

$$\widehat{Q}^{k+1} \leftarrow \arg\min_{Q} \mathbb{E}_{s,a,s' \sim \mathcal{D}} \left[\left(\left(r(s,a) + \gamma \mathbb{E}_{a' \sim \widehat{\pi}^k(a' \mid s')} \left[\widehat{Q}^k(s', a') \right] \right) - Q(s,a) \right)^2 \right] \quad \text{(policy evaluation)} \tag{1}$$

$$\widehat{\pi}^{k+1} \leftarrow \arg\max_{\pi} \mathbb{E}_{s \sim \mathcal{D}, a \sim \pi(a \mid s)} \left[\widehat{Q}^{k+1}(s,a) \right] \quad \text{(policy improvement)} \tag{2}$$

- **Action distribution shift during training**:
  - π is trained to maximize Q-values $\Rightarrow$ maybe biased towards OoD actions with erroneously high Q-values.
  - because the target values for Bellman backups in policy evaluation use $a \sim \pi^k$, but the Q-function is trained only on $a \sim \pi_\beta(\cdot|s), s \sim \mathcal{D}$.
- **No state distribution shift issue during training**. However, the policy may suffer from **state distribution shift at test time**.

Outline

Policy constraints and penalities methods

- **Algorithmic framework for policy constraints**:

$$\widehat{Q}_{k+1}^{\pi} \leftarrow \arg\min_{Q} \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\left(Q(s,a) - \left(r(s,a) + \gamma \mathbb{E}_{a'\sim\pi_k(a'\mid s')}\left[\widehat{Q}_{k}^{\pi}\left(s',a'\right)\right]\right)\right)^2\right] \tag{3}$$

$$\pi_{k+1} \leftarrow \arg\max_{\pi} \mathbb{E}_{s\sim\mathcal{D}}\left[\mathbb{E}_{a\sim\pi(a\mid s)}\left[\widehat{Q}_{k+1}^{\pi}(s,a)\right]\right] \text{ s.t. } D\left(\pi,\pi_{\beta}\right) \leq \epsilon \tag{4}$$

- **Algorithmic framework for policy penalities**:
  - modified reward function $\bar{r}(s,a) = r(s,a) - \alpha D\left(\pi(\cdot\mid s), \pi_{\beta}(\cdot\mid s)\right)$

$$\widehat{Q}_{k+1}^{\pi} \leftarrow \arg\min_{Q} \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\left(Q(s,a) - \left(r(s,a) + \gamma \mathbb{E}_{a'\sim\pi_k(a'\mid s')}\left[\widehat{Q}_{k}^{\pi}\left(s',a'\right)\right] - \alpha\gamma D\left(\pi_k\left(\cdot\mid s'\right), \pi_{\beta}\left(\cdot\mid s'\right)\right)\right)\right)^2\right] \tag{5}$$

$$\pi_{k+1} \leftarrow \arg\max_{\pi} \mathbb{E}_{s\sim\mathcal{D}}\left[\mathbb{E}_{a\sim\pi(a\mid s)}\left[\widehat{Q}_{k+1}^{\pi}(s,a)\right] - \alpha D\left(\pi(\cdot\mid s), \pi_{\beta}(\cdot\mid s)\right)\right] \tag{6}$$

Different choice of divergence

- **Implicit f-divergence**: MARWIL [Wang, Xiong, Han, Liu, Zhang, et al., 2018], AWAC [Nair, Dalal, Gupta, and Levine, 2020].

Algorithm 1 Monotonic Advantage Re-Weighted Imitation Learning (MARWIL)

Require: Historical data $\mathcal{D}$ generated by π_β, hyper-parameter λ.

1: Performing the following maximization problem to obtain improved policy $\pi_{\theta_{\text{improved}}}$

$$\theta_{\text{improved}} = \arg\max_{\theta} \mathop{\mathbb{E}}_{s,a\sim\mathcal{D}} \left[\log \pi_\theta(a \mid s) \frac{1}{Z(s)} \exp\left(\frac{1}{\lambda} A^{\pi_\beta}(s,a) \right) \right] \tag{7}$$

- **Problem**: How to practically estimate $A^{\pi_\beta}(s,a)$?
- **Question**: How is MARWIL related to policy constraints methods?

Practical implementation of MARWIL: Single path estimation

- Suppose (s_t, a_t) belongs a trajectory $\tau \sim \mathcal{D}$ and

$$\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \cdots, s_t, a_t, r_t, \cdots)$$

- R_t is the single path cumulative reward starting from (s_t, a_t) on τ.
- Approximate value function of behavior policy using neural networks $V_\omega^{\pi_\beta}(s)$ using only offline data.
- In practice, good results can be achieved by simply using a **single path** estimation as $\widehat{A}(s_t, a_t) = R_t - V_\omega^{\pi_\beta}(s_t)$

What is MARWIL actually performing?

- First, solve the following policy optimization problem

$$\bar{\pi} = \arg\max_{\pi} \mathbb{E}_{a\sim\pi(\cdot|s)} \left[A^{\pi_\beta}(s,a)\right] - \lambda D_{\mathrm{KL}}\left(\pi(\cdot \mid s) \| \pi_\beta(\cdot \mid s)\right), \tag{8}$$

- which has a closed-form optimal solution obtained by enforcing the KKT conditions,

$$\bar{\pi}(a \mid s) = \frac{1}{Z(s)} \pi_\beta(a \mid s) \exp\left(\frac{1}{\lambda} A^{\pi_\beta}(s,a)\right) \tag{9}$$

- Then we project the closed-form 'phantom policy' into **the neural network policy class** using forward KL to avoid explicit behavior policy modeling.

$$\theta_{\text{improved}} \leftarrow \arg\min_{\theta} \mathbb{E}_{s\sim\beta}\left[D_{\mathrm{KL}}\left(\bar{\pi}, \pi_\theta\right)\right] \tag{10}$$

- **Solving MARWIL optimization problem 7 is equivalent to solving 8 and 10.**

Extensions of MARWIL

- AWAC [Nair, Dalal, Gupta, and Levine, 2020] extends MARWIL to multiple policy improvement step $k = 1, 2, \cdots$,

$$\theta_{k+1} = \arg\max_{\theta} \mathop{\mathbb{E}}_{s,a\sim\beta} \left[\log \pi_\theta(a \mid s) \frac{1}{Z(s)} \exp\left(\frac{1}{\lambda} A^{\pi_k}(s,a) \right) \right] \quad \text{where} \quad \pi_k = \pi_{\theta_k}, \beta = \mathcal{D},$$

which is equivalent to

$$\bar{\pi}_{k+1}(a \mid s) \leftarrow \frac{1}{Z(s)} \pi_\beta(a \mid s) \exp\left(\frac{1}{\lambda} A^{\pi_k}(s,a) \right) \tag{11}$$

$$\theta_{k+1} \leftarrow \arg\min_{\theta} \mathbb{E}_{s\sim\beta}\left[D_{\mathrm{KL}}\left(\bar{\pi}_{k+1}, \pi_\theta \right) \right] \tag{12}$$

Remark: **MARWIL** estimates A^{π_β} instead of A^{π_k}; **AWAC** uses a Q-critic network to evaluate Q^{π_k} via optimization problem 1 via transition triplet sampled from offline dataset and obtain estimate of A^{π_k}.

Different choice of divergence

- **Explicit f-divergence**: E.g. KL-divergence, DAPO [Wang, Li, Xiong, and Zhang, 2019] (details see Page 60), BRAC [Wu et al., 2019]
- **Integral probability metrics (IPMs)**:
  - BEAR[Kumar et al., 2019] used **(finite sample) MMD and justified as resembling a support constraining metric (a good tradeoff between sub-optimality and policy constraints)**,
  - Wasserstein distance in BRAC [Wu et al., 2019]
- Above methods require that the behavior policy is known or estimated well.

Drawbacks of policy constraint algorithms

- Intuitively, this algorithm can be understood as performing imitation learning, but permitting minor deviations.
- Constraining the policy to be near-in distribution to the empirical policy can fail to take advantage of highly-visited states which are reached via many trajectories.
- The policies which differ substantially in the conditional distribution can still produce very similar state visitation distributions.
- In fact, in the limit of infinite data, even spanning **full support** of state-action visitation distribution, policy constraint algorithms are not guaranteed to converge to the optimal policy.
  - For policy support matching algorithms, no guarantee that action support conditioned every state has full support on action space.
  - For other policy constraint, too restricted.

Outline

Naive model based approach can be arbitrarily bad

- The work of Ross and Bagnell [2012] theoretically studied the performance of model-based reinforcement learning in the offline batch setting.
- In particular, the algorithm they analyzed involves
  (1) learning a transition dynamics model using the offline dataset,
  (2) and subsequently planning in the learned model without any additional safeguards.
- Their theoretical results are largely negative for this algorithm, suggesting that in the worst case, this algorithm could have arbitrarily large sub-optimality gap.
- In addition, their sub-optimality bounds become pathologically loose when the data logging distribution does not share support with the distribution of the optimal policy.

Naive model based approach ‘Batch’ can be arbitrarily bad

- Let $\mathcal{T}$ denote the class of transition models considered, and ν a state-action exploration distribution we can sample the system from.
- ‘Batch’ first finds the best model $\widehat{T} \in \mathcal{T}$ of observed transitions, and solves (potentially approximately) the optimal control (OC) problem with $\widehat{T}$ and known cost function C to return a policy $\widehat{\pi}$ for test execution.
- Here the author consider OC as a minimization problem.
- **Question:** if ‘Batch’ learns a model $\widehat{T}$ with small error on training data, and solves the OC problem well, what guarantees does it provide on control performance of $\widehat{\pi}$?
  - Ross and Bagnell [2012] illustrate the drawbacks of a purely ‘batch’ method due to the mismatch in train-test distribution.

Analysis of 'Batch' methods in tabular setting

- The quality of the OC problem's solution:
  - $\widehat{V}^{\widehat{\pi}}$ and $\widehat{V}^{\pi'}$ are the value functions of $\widehat{\pi}$ and π' under learned model $\widehat{T}$ respectively)
  - For any policy π', let
    $$\epsilon_{\mathrm{oc}}^{\pi'} = \mathbb{E}_{s\sim\mu}\left[\widehat{V}^{\widehat{\pi}}(s) - \widehat{V}^{\pi'}(s)\right]$$
    denote how much better π' is compared to $\widehat{\pi}$ on model $\widehat{T}$
  - If $\widehat{\pi}$ is an ϵ-optimal policy on $\widehat{T}$ within some class of policies Π, then $\epsilon_{\mathrm{oc}}^{\pi'} \leq \epsilon$ for all $\pi' \in \Pi$
- A natural measure of model error that arises from the analysis is in terms of L_1 distance between the predicted and true next state's distributions.
  - the predictive error of $\widehat{T}$, measured in L_1 distance, under the training distribution ν.

$$\epsilon_{\mathrm{prd}}^{\mathrm{L1}} = \mathbb{E}_{(s,a)\sim\nu}\left[\left\|T_{sa} - \widehat{T}_{sa}\right\|_1\right]$$

- In general, we can use any loss minimizable from samples that upper bounds $\epsilon_{\mathrm{prd}}^{\mathrm{L1}}$ for models in the class.

Analysis of 'Batch' methods in tabular setting

- The mismatch between the state-action exploration distribution ν and distribution induced by executing another policy π starting in μ, denoted

$$c_\nu^\pi = \sup_{s,a} \frac{D_{\mu,\pi}(s,a)}{\nu(s,a)}$$

- Assume the costs $C(s,a) \in [C_{\min}, C_{\max}]$. Let $C_{\mathrm{rng}} = C_{\max} - C_{\min}$ and $H = \frac{\gamma C_{\mathrm{rng}}}{(1-\gamma)^2}$. H is a scaling factor that relates model error to error in total cost predictions.
- **Theorem.** The policy $\widehat{\pi}$ is s.t. for any policy π' (infinite data regime):

$$J_\mu(\widehat{\pi}) \leq J_\mu(\pi') + \epsilon_{oc}^{\pi'} + \frac{c_\nu^{\widehat{\pi}} + c_\nu^{\pi'}}{2} H \epsilon_{prd}^{\mathrm{L1}}$$

Drawbacks of ‘Batch’ methods in tabular setting

- $c_\nu^{\pi'}$ measures how well ν explores state actions visited by the policy π' we compare to.
  - This factor is inevitable: we cannot hope to compete against policies that spend most of their time where we rarely explore.
- $c_\nu^{\widehat{\pi}}$ measures the mismatch in train-test distribution. Its presence is the major drawback of ‘Batch’.
  - As $\widehat{\pi}$ cannot be known in advance, we can only bound $c_\nu^{\widehat{\pi}}$ by considering all policies we could learn: $\sup_{\pi \in \Pi} c_\nu^{\pi}$.
  - This worst case is likely to be realized in practice: if ν rarely explores some state-action regions, the model could be bad for these and significantly underestimate their cost. The learned policy is thus encouraged to visit these low-cost regions where few data were collected.

Drawbacks of 'Batch' methods in tabular setting

- To minimize $\sup_{\pi \in \Pi} c_\nu^\pi$, the best ν for Batch is often a uniform distribution, when possible. This introduces a dependency on the number of states and actions (or state-action space volume) (i.e..$c_\nu^{\widehat{\pi}} + c_\nu^{\pi'}$ is $O(|S||A|)$) multiplying the modeling error.
- Sampling from a uniform distribution often requires access to a generative model.
- If we only have access to a RL forward model and a base policy π_0 inducing ν when executed in the system, then $c_\nu^{\widehat{\pi}}$ could be arbitrarily large (e.g.if $\widehat{\pi}$ leads to 0 probability states under π_0), and $\widehat{\pi}$ arbitrarily worse than π_0.

Model based Offline Reinforcement Learning (MOReL)

- In contrast, MoRel [Kidambi, Rajeswaran, Netrapalli, and Joachims, 2020] present a novel algorithmic framework that constructs a pessimistic MDP, and show that this is crucial for better empirical results and sharper theoretical analysis.

MOReL Framework

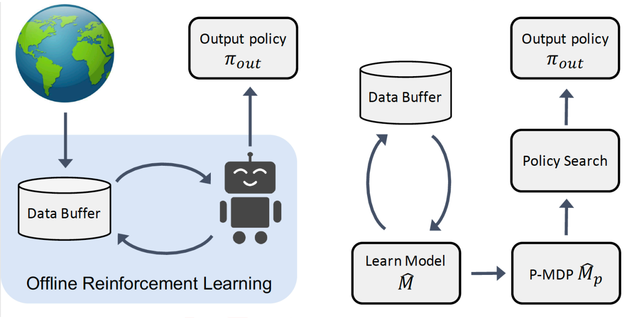
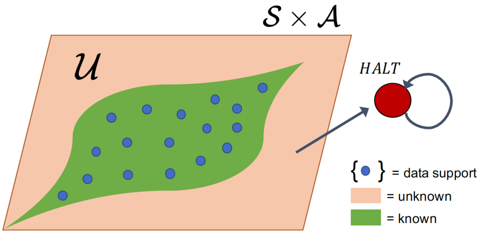


Figure: Illustration of MOReL framework which learns a pessimistic MDP (P-MDP) from the dataset and uses it for policy search.

MOReL Unknown state-action detector

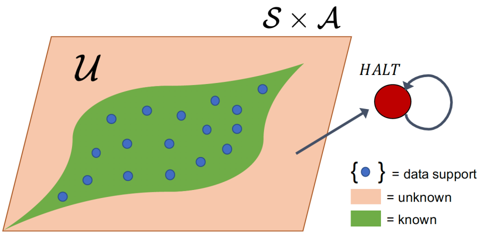


Unknown state-action detector (USAD): We partition the state-action space into known and unknown regions based on the accuracy of learned model as follows.

Definition 1. *(α-USAD) Given a state-action pair (s, a), define an unknown state action detector as:*

$$U^{\alpha}(s,a) = \begin{cases} \textit{FALSE} \ \ (\textit{i.e. Known}) & \text{if } D_{TV}\left(\hat{P}(\cdot|s,a), P(\cdot|s,a)\right) \leq \alpha \ \ \textit{can be guaranteed} \\ \textit{TRUE} \ \ (\textit{i.e. Unknown}) & \textit{otherwise} \end{cases} \tag{2}$$

MOReL Pessimistic MDP construction



Definition 2. *The (α, κ)-pessimistic MDP is described by $\hat{\mathcal{M}}_p := \{S \cup HALT, A, r_p, \hat{P}_p, \hat{\rho}_0, \gamma\}$. Here, S and A are states and actions in the MDP $\mathcal{M}$. HALT is an additional absorbing state we introduce into the state space of $\hat{\mathcal{M}}_p$. $\hat{\rho}_0$ is the initial state distribution learned from the dataset $\mathcal{D}$. γ is the discount factor (same as $\mathcal{M}$). The modified reward and transition dynamics are given by:*

$$\hat{P}_p(s'|s,a) = \begin{cases} \delta(s' = \text{HALT}) & \text{if } U^{\alpha}(s,a) = \text{TRUE} \\ & \text{or } \ s = \text{HALT} \\ \hat{P}(s'|s,a) & \textit{otherwise} \end{cases} \qquad r_p(s,a) = \begin{cases} -\kappa & \text{if } s = \text{HALT} \\ r(s,a) & \textit{otherwise} \end{cases}$$

MOReL algorithm

Algorithm 1 `MOReL`: Model Based Offline Reinforcement Learning

1: **Require** Dataset $\mathcal{D}$
2: Learn approximate dynamics model $\hat{P} : S \times A \to S$ using $\mathcal{D}$.
3: Construct α-USAD, $U^\alpha : S \times A \to \{\text{TRUE}, \text{FALSE}\}$ using $\mathcal{D}$ (see Definition 1).
4: Construct the *pessimistic* MDP $\hat{\mathcal{M}}_p = \{S \cup \text{HALT}, A, r_p, \hat{P}_p, \hat{\rho}_0, \gamma\}$ (see Definition 2).
5: (OPTIONAL) Use a behavior cloning approach to estimate the behavior policy $\hat{\pi}_b$.
6: $\pi_{\text{out}} \leftarrow \text{PLANNER}(\hat{\mathcal{M}}_p, \pi_{\text{init}} = \hat{\pi}_b)$
7: **Return** π_{out}.

Practical implementation of MoReL

- **Dynamics model learning**: Gaussian dynamics models $\hat{P}(\cdot \mid s, a) \equiv \mathcal{N}\left(f_\phi(s, a), \Sigma\right)$, with mean $f_\phi(s, a) = s + \sigma_\Delta \operatorname{MLP}_\phi\left(\left(s-\mu_s\right) / \sigma_s, \left(a-\mu_a\right) / \sigma_a\right)$, where $\mu_s, \sigma_s, \mu_a, \sigma_a$ are the mean and standard deviations of states/actions in $\mathcal{D}$; σ_Δ is the standard deviation of state differences, i.e. $\Delta = s' - s, \left(s, s'\right) \in \mathcal{D}$;
- **Unknown state-action detector (USAD)**: Track uncertainty using the predictions of **ensembles of models**. Learn multiple models $\left\{f_{\phi_1}, f_{\phi_2}, \ldots\right\}$ where each model uses a different weight initialization and are optimized with different mini-batch sequences. **Ensemble discrepancy**: $\operatorname{disc}(s, a) = \max_{i, j}\left\|f_{\phi_i}(s, a) - f_{\phi_j}(s, a)\right\|_2$, With this, we implement **USAD** as below:

$$U_{\text{practical}}(s, a) = \begin{cases} \text{FALSE (i.e. Known)} & \text{if } \operatorname{disc}(s, a) \leq \text{ threshold} \\ \text{TRUE (i.e. Unknown)} & \text{if } \operatorname{disc}(s, a) > \text{ threshold} \end{cases}$$

Questions to be answered

1. **Comparison to prior work**: How does MOReL compare to prior SOTA offline RL algorithms in commonly studied benchmark tasks?
2. **Quality of logging policy**: How does the quality (value) of the data logging (behavior) policy, and by extension the dataset, impact the quality of the policy learned by MOReL?
3. **Importance of pessimistic MDP**: How does MOReL compare against a naïve model-based RL approach that directly plans in a learned model without any safeguards?
4. **Transfer from pessimistic MDP to environment**: Does learning progress in the P-MDP, which we use for policy learning, effectively translate or transfer to learning progress in the environment?

Logging offline data (I)

Figure: Four continuous control tasks in Gym environment.

- First, partially train a policy (π_b) to obtain values around 1000, 4000, 1000, and 1000 respectively for the four environments using baseline policy optimization algorithm for continuous action space.
- Prepare an untrained random gaussian policy π_r.

Logging offline dataset (II)

($\mathcal{E}1$) Pure: The entire dataset is collected with the data logging (behavioral) policy π_b.

($\mathcal{E}2$) Eps-1: 40% of the dataset is collected with π_b, another 40% collected with $\pi_b^u(0.1)$, and the final 20% is collected with a random policy π_r.

($\mathcal{E}3$) Eps-3: 40% of the dataset is collected with π_b, another 40% collected with $\pi_b^u(0.3)$, and the final 20% is collected with a random policy π_r.

($\mathcal{E}4$) Gauss-1: 40% of the dataset is collected with π_b, another 40% collected with $\pi_b^g(0.1)$, and the final 20% is collected with a random policy π_r.

($\mathcal{E}5$) Gauss-3: 40% of the dataset is collected with π_b, another 40% collected with $\pi_b^g(0.3)$, and the final 20% is collected with a random policy π_r.

MOReL Performance (I) - Compared to baselines

Table 1: Results in various environment-exploration combinations. Baselines are reproduced from Wu et al. [18]. Prior work does not provide error bars. For MOReL results, error bars indicate the standard deviation across 5 different random seeds. We choose SOTA result based on the average performance.

Environment: Ant-v2					
Algorithm	BCQ [15]	BEAR [16]	BRAC [18]	Best Baseline	MOReL (Ours)
Pure	1921	2100	2839	2839	**3663±247**
Eps-1	1864	1897	2672	2672	**3305±413**
Eps-3	1504	2008	2602	2602	**3008±231**
Gauss-1	1731	2054	2667	2667	**3329±270**
Gauss-3	1887	2018	2640	2661	**3693±33**

Environment: Hopper-v2					
Algorithm	BCQ [15]	BEAR [16]	BRAC [18]	Best Baseline	MOReL (Ours)
Pure	1543	0	2291	2774	**3642±54**
Eps-1	1652	1620	2282	2360	**3724±46**
Eps-3	1632	2213	1892	2892	**3535±91**
Gauss-1	1599	1825	2255	2255	**3653±52**
Gauss-3	1590	1720	1458	2097	**3648±148**

Environment: HalfCheetah-v2					
Algorithm	BCQ [15]	BEAR [16]	BRAC [18]	Best Baseline	MOReL (Ours)
Pure	5064	5325	6207	**6209**	**6028±192**
Eps-1	5693	5435	**6307**	**6307**	5861±192
Eps-3	5588	5149	6263	**6359**	5869±139
Gauss-1	5614	5394	**6323**	**6323**	6026±74
Gauss-3	5837	5329	**6400**	**6400**	5892±128

Environment: Walker-v2					
Algorithm	BCQ [15]	BEAR [16]	BRAC [18]	Best Baseline	MOReL (Ours)
Pure	2095	2646	2694	2907	**3709±159**
Eps-1	1921	2695	3241	**3490**	2899±588
Eps-3	1953	2608	**3255**	**3255**	**3186±92**
Gauss-1	2094	2539	2893	3193	**4027±314**
Gauss-3	1734	2194	**3368**	**3368**	**2828±589**

MOReL Performance (II) - Impact from the quality of logging policy

- Pure-random dataset from untrained random Gaussian policy π_r.
- Pure-partial dataset is the $\mathcal{E}1$ dataset.

Table 2: Value of the policy learned by MOReL (5 random seeds) when working with a dataset collected with a random (untrained) policy (Pure-random) and a partially trained policy (Pure-partial).

Environment	Pure-random	Pure-partial
Hopper-v2	2354 ± 443	3642 ± 54
HalfCheetah-v2	2698 ± 230	6028 ± 192
Walker2d-v2	1290 ± 325	3709 ± 159
Ant-v2	1001 ± 3	3663 ± 247

MOReL Performance (III) - Importance of Pessimistic MDP

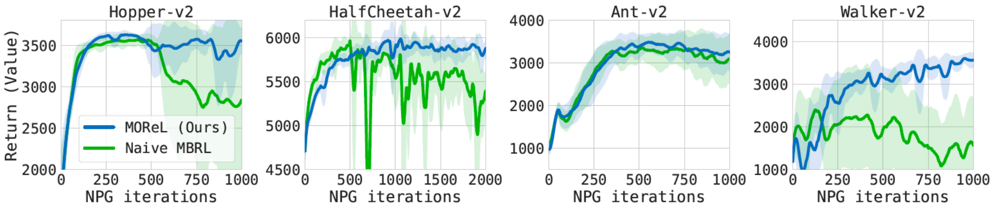


Figure 3: MOReL and Naive MBRL learning curves. The x-axis plots the number of model-based NPG iterations, while y axis plots the return (value) in the real environment. The naive MBRL algorithm is highly unstable while MOReL leads to stable and near-monotonic learning. Notice however that even naive MBRL learns a policy that performs often as well as the best model-free offline RL algorithms.

MOReL Performance (IV) - Transfer from P-MDP to environment

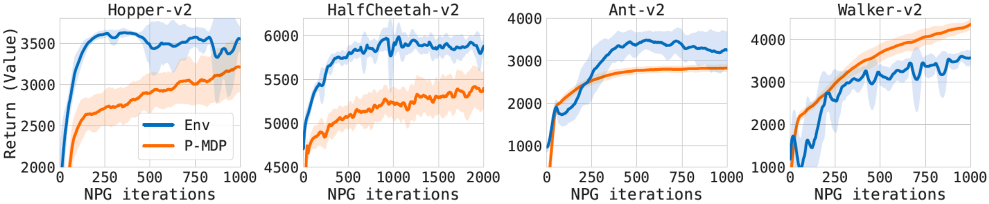


Figure 4: Learning curve using the `Pure-partial` dataset, see paper text for details. The policy is learned using the pessimistic MDP (P-MDP), and we plot the performance in both the P-MDP and the real environment over the course of learning. We observe that the performance in the P-MDP closely tracks the true performance and never substantially exceeds it, as predicted in section 4.1. This shows that the policy value in the P-MDP serves as a good surrogate for the purposes of offline policy evaluation and learning.

MOReL Main theorem

Instance dependent quantity.

Definition 3. *(Hitting time) Given an MDP $\mathcal{M}$, starting state distribution ρ_0, state-action pair (s, a) and a policy π, the* hitting time $T^{\pi}_{(s,a)}$ *is defined as the random variable denoting the first time action a is taken at state s by π on $\mathcal{M}$, and is equal to ∞ if a is never taken by π from state s. For a set of state-action pairs $\mathcal{S} \subseteq S \times A$, we define $T^{\pi}_{\mathcal{S}} \stackrel{def}{=} \min_{(s,a)\in\mathcal{S}} T^{\pi}_{(s,a)}$.*

Theorem 1. *(Policy value with pessimism) The value of any policy π on the original MDP $\mathcal{M}$ and its $(\alpha, R_{\max})$-pessimistic MDP $\hat{\mathcal{M}}_p$ satisfies:*

$$J_{\hat{\rho}_0}(\pi, \hat{\mathcal{M}}_p) \geq J_{\rho_0}(\pi, \mathcal{M}) - \frac{2R_{max}}{1-\gamma} \cdot D_{TV}(\rho_0, \hat{\rho}_0) - \frac{2\gamma R_{max}}{(1-\gamma)^2} \cdot \alpha - \frac{2R_{max}}{1-\gamma} \cdot \mathbb{E}\left[\gamma^{T^{\pi}_{\mathcal{U}}}\right], \text{ and}$$

$$J_{\hat{\rho}_0}(\pi, \hat{\mathcal{M}}_p) \leq J_{\rho_0}(\pi, \mathcal{M}) + \frac{2R_{max}}{1-\gamma} \cdot D_{TV}(\rho_0, \hat{\rho}_0) + \frac{2\gamma R_{max}}{(1-\gamma)^2} \cdot \alpha,$$

where $T^{\pi}_{\mathcal{U}}$ denotes the hitting time of unknown states $\mathcal{U} \stackrel{def}{=} \{(s, a) : U^{\alpha}(s, a) = TRUE\}$ by π on $\mathcal{M}$.

MOReL Upper bound

Corollary 2. *Suppose PLANNER in Algorithm 1 returns an ϵ_π sub-optimal policy. Then, we have*

$$J_{\rho_0}(\pi^*, \mathcal{M}) - J_{\rho_0}(\pi_{out}, \mathcal{M}) \leq \epsilon_\pi + \frac{4R_{max}}{1-\gamma} \cdot D_{TV}(\rho_0, \hat{\rho}_0) + \frac{4\gamma R_{max}}{(1-\gamma)^2} \cdot \alpha + \frac{2R_{max}}{1-\gamma} \cdot \mathbb{E}\left[\gamma^{T_{\mathcal{U}}^{\pi^*}}\right].$$

MOReL Upper bound

Lemma 5. *(Hitting time and visitation distribution) For any set $\mathcal{S} \subseteq S \times A$, and any policy π, we have $\mathbb{E}\left[\gamma^{T^{\pi}_{\mathcal{S}}}\right] \leq \frac{1}{1-\gamma} \cdot d^{\pi,\mathcal{M}}(\mathcal{S})$.*

Proof of Lemma 5. The proof is rather straightforward. We have

$$\mathbb{E}\left[\gamma^{T^{\pi}_{\mathcal{U}}}\right] \leq \sum_{(s',a')\in\mathcal{U}} \mathbb{E}\left[\gamma^{T^{\pi}_{(s',a')}}\right] \leq \sum_{(s',a')\in\mathcal{U}} \sum_{t=0}^{\infty} \gamma^t P(s_t = s', a_t = a' | s_0 \sim \rho_0, \pi, \mathcal{M})$$
$$= \frac{1}{1-\gamma} \sum_{(s',a')\in\mathcal{U}} d^{\pi,\mathcal{M}}(s',a') = \frac{1}{1-\gamma} \cdot d^{\pi,\mathcal{M}}(\mathcal{U}).$$

□

Corollary 3. *(Upper bound) Suppose the dataset $\mathcal{D}$ is large enough so that $\alpha = D_{TV}(\rho_0, \hat{\rho}_0) = 0$. Then, the output π_{out} of Algorithm 1 satisfies:*

$$J_{\rho_0}(\pi^*, \mathcal{M}) - J_{\rho_0}(\pi_{out}, \mathcal{M}) \leq \epsilon_\pi + \frac{2R_{max}}{1-\gamma} \cdot \mathbb{E}\left[\gamma^{T^{\pi^*}_{\mathcal{U}}}\right] \leq \epsilon_\pi + \frac{2R_{\max}}{(1-\gamma)^2} \cdot d^{\pi^*,\mathcal{M}}(\mathcal{U})$$

MOReL policy compared to behavior policy

- Finally, we note that as the size of dataset $\mathcal{D}$ increases to ∞, Theorem 1 and the optimality of PLANNER together imply that $J_{\rho_0}(\pi_{\text{out}}, \mathcal{M}) \geq J_{\rho_0}(\pi_b, \mathcal{M})$ since $\mathbb{E}\left[\gamma^{T_{\mathcal{U}}^{\pi}}\right]$ goes to 0.

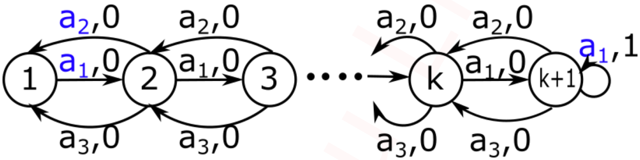
Theorem 1. *(Policy value with pessimism) The value of any policy π on the original MDP $\mathcal{M}$ and its $(\alpha, R_{\max})$-pessimistic MDP $\hat{\mathcal{M}}_p$ satisfies:*

$$J_{\hat{\rho}_0}(\pi, \hat{\mathcal{M}}_p) \geq J_{\rho_0}(\pi, \mathcal{M}) - \frac{2R_{max}}{1-\gamma} \cdot D_{TV}(\rho_0, \hat{\rho}_0) - \frac{2\gamma R_{max}}{(1-\gamma)^2} \cdot \alpha - \frac{2R_{max}}{1-\gamma} \cdot \mathbb{E}\left[\gamma^{T_{\mathcal{U}}^{\pi}}\right], \text{ and}$$

$$J_{\hat{\rho}_0}(\pi, \hat{\mathcal{M}}_p) \leq J_{\rho_0}(\pi, \mathcal{M}) + \frac{2R_{max}}{1-\gamma} \cdot D_{TV}(\rho_0, \hat{\rho}_0) + \frac{2\gamma R_{max}}{(1-\gamma)^2} \cdot \alpha,$$

where $T_{\mathcal{U}}^{\pi}$ denotes the hitting time of unknown states $\mathcal{U} \overset{def}{=} \{(s,a) : U^{\alpha}(s,a) = TRUE\}$ by π on $\mathcal{M}$.

MOReL Lower bound



Proposition 4. *(Lower bound) For any discount factor* $\gamma \in [0.95, 1)$, *support mismatch* $\epsilon \in \left(0, \frac{1-\gamma}{\log \frac{1}{1-\gamma}}\right]$ *and reward range* $[-R_{max}, R_{max}]$, *there is an MDP* $\mathcal{M}$, *starting state distribution* ρ_0, *optimal policy* π^* *and a dataset collection policy* π_b *such that i)* $d^{\pi^*,\mathcal{M}}(\mathcal{U}_D) \leq \epsilon$, *and ii) any policy* $\hat{\pi}$ *that is learned solely using the dataset collected with* π_b *satisfies:*

$$J_{\rho_0}(\pi^*, \mathcal{M}) - J_{\rho_0}(\hat{\pi}, \mathcal{M}) \geq \frac{R_{max}}{4(1-\gamma)^2} \cdot \frac{\epsilon}{\log \frac{1}{1-\gamma}},$$

where $\mathcal{U}_D \stackrel{def}{=} \{(s, a) : (s, a, r, s') \notin \mathcal{D} \text{ for any } r, s'\}$ *denotes state action pairs not in the dataset* $\mathcal{D}$.

MOReL Lower bound proof

- We set $k = 10 \log \frac{1}{1-\gamma}$.
- The MDP has $k+1$ states, with three actions a_1, a_2 and a_3 at each state.
- The rewards (shown on the transition arrows) are all 0 except for the action a_1 taken in state $k+1$, in which case it is 1.
- Note that the rewards can be scaled by $R_{\max}$ but for simplicity, we consider the setting with $R_{\max} = 1$.
- It is clear that the optimal policy π^* is to take the action a_1 in all the states.
- The starting state distribution ρ_0 is state 1 with probability $p_0 \stackrel{\text{def}}{=} \frac{\epsilon}{(1-\gamma)\log\frac{1}{1-\gamma}}$ and state $k+1$ with probability $1 - p_0$.
- The actions taken by the data collection policy are shown in blue. since the dataset consists only of (state, action, reward, next state) pairs $(1, a_1, 0, 2), (2, a_2, 0, 1)$ and $(k+1, a_1, 1, k+1)$ we see that $\mathcal{U}_D = (S \times A) \backslash \{(1, a_1), (2, a_2), (k+1, a_1)\}$ and $d^{\pi^*, \mathcal{M}}(\mathcal{U}_D) = (1-\gamma) \cdot \sum_{t=1}^{k-1} \gamma^t \cdot p_0 \leq (1-\gamma) \cdot (k-1) \cdot p_0 \leq \epsilon$ proving the first claim.

MOReL Lower bound proof

- Since none of the states and actions in $\mathcal{U}_D$ are seen in the dataset, after permuting the actions if necessary, the expected time taken by any policy learned from the dataset, to reach state $k+1$ starting from state 1 is at least $\exp(k/5) \geq (1-\gamma)^{-2}$.
- So, the value of any policy $\widehat{\pi}$ learned from the dataset is at most $\frac{1-p_0}{1-\gamma} + \frac{p_0 \cdot \gamma^{(1-\gamma)^{-2}}}{1-\gamma} = \frac{1}{1-\gamma} - p_0 \cdot \frac{1-\gamma^{(1-\gamma)^{-2}}}{1-\gamma} \leq \frac{1}{1-\gamma} - \frac{3p_0}{4(1-\gamma)}$, where we used $\gamma \in [0.95, 1)$ in the last step.
- On the other hand, the value of π^* is at least $\frac{1-p_0}{1-\gamma} + p_0 \cdot \left(\frac{1}{1-\gamma} - k\right)$. So the suboptimality of any learned policy is at least $p_0 \cdot \left(\frac{3}{4(1-\gamma)} - k\right) = p_0 \cdot \left(\frac{3}{4(1-\gamma)} - 10\log\frac{1}{1-\gamma}\right) \geq \frac{p_0}{4(1-\gamma)}$, where we again used $\gamma \in [0.95, 1)$ in the last step. Substituting the value of p_0 proves the proposition.

MOPO: Model based Offline Policy Optimization

- Another simultaneous work called MOPO [Yu, Thomas, Yu, Ermon, Zou, Levine, Finn, and Ma, 2020] is derived in a similar way.

Lemma 1 (Simulation/Telescoping lemma (Refer to Page 9 in Lecture 8)).
Let M and $\widehat{M}$ be two MDPs with the same reward function r, but different dynamics T and $\widehat{T}$ respectively. Let $G^{\pi}_{\widehat{M}}(s,a) := \mathbb{E}_{s' \sim \widehat{T}(s,a)}\left[V^{\pi}_{M}(s')\right] - \mathbb{E}_{s' \sim T(s,a)}\left[V^{\pi}_{M}(s')\right]$. Then,

$$\eta_{\widehat{M}}(\pi) - \eta_M(\pi) = \gamma \mathbb{E}_{(s,a)\sim \rho^{\pi}_{\widehat{T}}}\left[G^{\pi}_{\widehat{M}}(s,a)\right] \tag{13}$$

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

$$|G^{\pi}_{\widehat{M}}(s,a)| \le \sup_{f \in \mathcal{F}} \left| \mathbb{E}_{s' \sim \widehat{T}(s,a)}\left[f(s')\right] - \mathbb{E}_{s' \sim T(s,a)}\left[f(s')\right] \right| =: d_{\mathcal{F}}(\widehat{T}(s,a), T(s,a)), \tag{14}$$

Assumptions

Assumption 1.

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

As a direct corollary of Assumption 1 and equation equation 14, we have

$$|G^\pi_{\widehat{M}}(s,a)| \leq c d_{\mathcal{F}}(\widehat{T}(s,a), T(s,a)). \tag{15}$$

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 $\widehat{T}$ if $d_{\mathcal{F}}(\widehat{T}(s,a), T(s,a)) \leq u(s,a)$ for all $s \in \mathcal{S}, a \in \mathcal{A}$.

Penalized virtual MDP construction

- Given an admissible error estimator, we define the *uncertainty-penalized reward* $\tilde{r}(s,a) := r(s,a) - \lambda u(s,a)$ where $\lambda := \gamma c$, and the *uncertainty-penalized MDP* $\widetilde{M} = (\mathcal{S}, \mathcal{A}, \widehat{T}, \tilde{r}, \mu_0, \gamma)$.
- We observe that $\widetilde{M}$ is conservative in that the return under it bounds from below the true return:

$$\begin{aligned} \eta_M(\pi) &\geq \mathbb{E}_{(s,a)\sim\rho^{\pi}_{\widehat{T}}}\left[r(s,a) - \gamma |G^{\pi}_{\widehat{M}}(s,a)|\right] \geq \mathbb{E}_{(s,a)\sim\rho^{\pi}_{\widehat{T}}}\left[r(s,a) - \lambda u(s,a)\right] \\ &\geq \mathbb{E}_{(s,a)\sim\rho^{\pi}_{\widehat{T}}}\left[\tilde{r}(s,a)\right] = \eta_{\widetilde{M}}(\pi) \end{aligned} \tag{16}$$

Model based Offline policy optimization

Algorithm 2 Framework for Model-based Offline Policy Optimization (MOPO) with Reward Penalty

Require: Dynamics model $\widehat{T}$ with admissible error estimator $u(s,a)$; constant λ.

1: Define $\tilde{r}(s,a) = r(s,a) - \lambda u(s,a)$. Let $\widetilde{M}$ be the MDP with dynamics $\widehat{T}$ and reward $\tilde{r}$.

2: Run any RL algorithm on $\widetilde{M}$ until convergence to obtain

$$\widehat{\pi} = \text{argmax}_{\pi} \eta_{\widetilde{M}}(\pi) \tag{17}$$

MOPO Practical Implementation

Algorithm 3 MOPO instantiation with regularized probabilistic dynamics and ensemble uncertainty

Require: reward penalty coefficient λ rollout horizon h, rollout batch size b.

```
Train on batch data $\mathcal{D}_{\text{env}}$ an ensemble of $N$ probabilistic dynamics $\{\widehat{T}^i(s', r \mid s, a) = \mathcal{N}(\mu^i(s,a), \Sigma^i(s,a))\}_{i=1}^N$.
Initialize policy $\pi$ and empty replay buffer $\mathcal{D}_{\text{model}} \leftarrow \varnothing$.
for epoch $1, 2, \ldots$ do    ▷ This for-loop is essentially one outer iteration of MBPO
    for $1, 2, \ldots, b$ (in parallel) do
        Sample state $s_1$ from $\mathcal{D}_{\text{env}}$ for the initialization of the rollout.
        for $j = 1, 2, \ldots, h$ do
            Sample an action $a_j \sim \pi(s_j)$.
            Randomly pick dynamics $\widehat{T}$ from $\{\widehat{T}^i\}_{i=1}^N$ and sample $s_{j+1}, r_j \sim \widehat{T}(s_j, a_j)$.
            Compute $\tilde{r}_j = r_j - \lambda \max_{i=1}^N \|\Sigma^i(s_j, a_j)\|_{\mathsf{F}}$.
            Add sample $(s_j, a_j, \tilde{r}_j, s_{j+1})$ to $\mathcal{D}_{\text{model}}$.
    Drawing samples from $\mathcal{D}_{\text{env}} \cup \mathcal{D}_{\text{model}}$, use SAC to update $\pi$.
```

MOPO Theoretical justification (I)

- Let $\pi^\star$ be the optimal policy on M and π^{B} be the policy that generates the batch data. Define $\epsilon_u(\pi)$ as

$$\epsilon_u(\pi) := \underset{(s,a)\sim\rho^{\pi}_{\widehat{T}}}{\mathbb{E}}[u(s,a)]$$

- For $\delta \geq \delta_{\min} := \min_\pi \epsilon_u(\pi)$, let π^δ be the best policy among those incurring model error at most δ :

$$\pi^\delta := \underset{\pi:\epsilon_u(\pi)\leq\delta}{\arg\max}\, \eta_M(\pi)$$

MOPO Theoretical justification (II)

Theorem 2.
Under Assumption 1 and 2, the learned policy $\widehat{\pi}$ in MOPO (Algorithm 2) satisfies

$$\eta_M(\widehat{\pi}) \geq \sup_{\pi}\{\eta_M(\pi) - 2\lambda\epsilon_u(\pi)\} \tag{18}$$

In particular, for all $\delta \geq \delta_{\min}$, $\eta_M(\widehat{\pi}) \geq \eta_M(\pi^{\delta}) - 2\lambda\delta$

- Consequence 1: for behavior policy π_B, $\epsilon_u\left(\pi^{\mathrm{B}}\right)$ is expected to be small. $\eta_M(\hat{\pi}) \geq \eta_M\left(\pi^{\mathrm{B}}\right) - 2\lambda\epsilon_u\left(\pi^{\mathrm{B}}\right) \approx \eta_M(\pi^{\mathrm{B}})$.
- Consequence 2: (18) tells us that the learned policy $\hat{\pi}$ can be as good as any policy π with $\epsilon_u(\pi) \leq \delta$, or in other words, any policy that visits states with sufficiently small uncertainty as measured by $u(s, a)$.
- Consequence 3: by varying the choice of δ to maximize the RHS of (18), we trade off the risk and the return.

Appendix: Interpretation of DAPO via the pseudo rewards

- Policy optimization over the pseudo reward

$$r(s,a) - \frac{1}{\eta}\log\frac{\pi(a|s)}{\pi_t(a|s)} \text{ (DAPO)} \quad \text{or} \quad r(s,a) - \frac{1}{\eta}\log\frac{\mu_\pi(s,a)}{\mu_t(s,a)} \text{ (Hard to implement)}$$

  can be interpreted as **trading off between high return and taking the risk of escaping off-policy data coverage**

  - encouraging visitation by dynamically adding positive bonus rewards in the state-action region s.t. $\mu_\pi(s,a) < \mu_t(s,a)$ or $\pi(a|s) < \pi_t(a|s)$
  - and discourage visitation by adding negative bonus rewards in the state-action region s.t. $\mu_\pi(s,a) > \mu_t(s,a)$ or $\pi(a|s) > \pi_t(a|s)$

- Similar to DAPO[Wang, Li, Xiong, and Zhang, 2019], BRAC [Wu et al., 2019] apply multi-step divergence penalization in the offline setting.

References I

S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. In International Conference on Machine Learning, pages 2052–2062, 2019.

S. K. S. Ghasemipour, D. Schuurmans, and S. S. Gu. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl. arXiv preprint arXiv:2007.11091, 2020.

R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims. Morel: Model-based offline reinforcement learning. arXiv preprint arXiv:2005.05951, 2020.

A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In Advances in Neural Information Processing Systems, pages 11761–11771, 2019.

A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. arXiv preprint arXiv:2006.04779, 2020.

References II

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

Y. Liu, A. Swaminathan, A. Agarwal, and E. Brunskill. Provably good batch reinforcement learning without great exploration. arXiv preprint arXiv:2007.08202, 2020.

Y. Luo, H. Xu, and T. Ma. Learning self-correctable policies and value functions from demonstrations with negative sampling. arXiv preprint arXiv:1907.05634, 2019.

A. Nair, M. Dalal, A. Gupta, and S. Levine. Accelerating online reinforcement learning with offline datasets. arXiv preprint arXiv:2006.09359, 2020.

S. Ross and J. A. Bagnell. Agnostic system identification for model-based reinforcement learning. In Proceedings of the 29th International Coference on International Conference on Machine Learning, pages 1905–1912, 2012.

References III

Q. Wang, J. Xiong, L. Han, H. Liu, T. Zhang, et al. Exponentially weighted imitation learning for batched historical data. In Advances in Neural Information Processing Systems, pages 6288–6297, 2018.

Q. Wang, Y. Li, J. Xiong, and T. Zhang. Divergence-augmented policy optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32, pages 6097–6108. Curran Associates, Inc., 2019. URL `http://papers.nips.cc/paper/8842-divergence-augmented-policy-optimization.pdf`.

Y. Wu, G. Tucker, and O. Nachum. Behavior regularized offline reinforcement learning, 2019.

T. Yu, G. Thomas, L. Yu, S. Ermon, J. Zou, S. Levine, C. Finn, and T. Ma. Mopo: Model-based offline policy optimization. arXiv preprint arXiv:2005.13239, 2020.