Policy-conditioned Environment Models are More Generalizable

Ruifeng Chen [*1 2] Xiong-Hui Chen [*1 2] Yihao Sun [1] Siyuan Xiao [1] Minhui Li [1] Yang Yu [1 2]

Abstract

In reinforcement learning, it is crucial to have an accurate environment dynamics model to evaluate different policies' value in downstream tasks like offline policy optimization and policy evaluation. However, the learned model is known to be inaccurate in predictions when evaluating target policies different from data-collection policies. In this work, we found that utilizing policy representation for model learning, called policy-conditioned model (PCM) learning, is useful to mitigate the problem, especially when the offline dataset is collected from diversified behavior policies. The reason beyond that is in this case, PCM becomes a meta-dynamics model that is trained to be aware of and focus on the evaluation policies that on-the-fly adjust the model to be suitable to the evaluation policies' state-action distribution, thus improving the prediction accuracy. Based on that intuition, we propose an easy-to-implement yet effective algorithm of PCM for accurate model learning. We also give a theoretical analysis and experimental evidence to demonstrate the feasibility of reducing value gaps by adapting the dynamics model under different policies. Experiment results show that PCM outperforms the existing SOTA off-policy evaluation methods in the DOPE benchmark with *a large margin*, and derives significantly better policies in offline policy selection and model predictive control compared with the standard model learning method.

1. Introduction

Environment model learning, which learns a model to approximate state transitions and reward functions of the environment, has extensive applications in Offline Policy Evaluation (OPE) (Thomas et al., 2015; Doroudi et al., 2017) and offline Reinforcement Learning (offline RL) (Lange et al., 2012; Levine et al., 2020). In OPE, policy value is estimated by calculating the return of simulated trajectories from the learned model. In offline RL, approaches utilize the model for planning or optimizing policy to maximize the return. Model accuracy significantly affects the efficacy of these methodologies. However, a learned model is known to have a large value gap when used to evaluate a target policy different from data-collection policies (Xu et al., 2020; Clavera et al., 2018; Janner et al., 2019; Yu et al., 2020). Although the learned dynamics model achieves a small error measured under the offline data distribution, the target policy tends to visit a different state-action distribution, which may be unfamiliar to the dynamics model (Janner et al., 2019). Therefore, the model will generate unreliable transitions for the shifted distribution, resulting in erroneous value estimates (Xu et al., 2021). This issue hinders the adoption of models in more scenarios.

In this article, we provide a new offline model learning paradigm for accurate dynamics model learning, called policy-conditioned model (PCM) learning. PCM learning includes an extra context for model learning, representing the policy to be evaluated, and an extra loss to learn policy representation. Compared with the standard model learning paradigm, which fits the whole dataset with a unified model and rolls out whatever target policy within the model (Yu et al., 2020), in this way, PCM becomes a meta-dynamics model trained to be aware of the evaluation policies and make predictions by adapting to the policies' state-action distribution to improve prediction accuracy. Our major finding is that PCM will enjoy more extra generalization benefits from the adaptation mechanism when the offline dataset is collected from more diverse behavioral policies. In practice, we provide an easy-to-implement yet effective algorithm for PCM learning: we implement PCM via policy representation techniques (Duan et al., 2016; Chen et al., 2021; Nagabandi et al., 2019), which adopt an extra network module to on-the-fly encode policies' representation and input the policy representations as well as a state-action pair into the meta-dynamics model for next-state predictions. PCM produces different dynamics models given different policy representations. We also theoretically show that PCM can achieve a smaller value gap for a target policy compared

*Equal contribution [1]National Key Laboratory for Novel Software Technology, Nanjing University, China & School of Artificial Intelligence, Nanjing University, China [2]Polixir Technologies. Correspondence to: Yang Yu <yuy@nju.edu.cn>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

with standard policy-agnostic models under some assumptions.

Experiments are conducted based on MuJoCo (Todorov et al., 2012). We first conducted a proof-of-concept experiment, utilizing our custom-made dataset, which verified the effectiveness of the policy-aware mechanism for improving the model prediction accuracy. Then apply PCM in several downstream tasks. Results show that PCM improves the performance of off-policy evaluation in the DOPE benchmark with *a large margin*, and derives significantly better policies in offline policy selection and model predictive control compared with the standard model learning method.

2. Preliminaries

2.1. Markov Decision Process and Reinforcement Learning

We consider a Markov decision process (MDP) (Sutton & Barto, 2018) specified by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, T, \gamma, \rho_0)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r(s, a)$ is the reward function, $T(s'|s, a)$ is the transition function, $\gamma \in (0, 1)$ is the discount factor, and $\rho_0(s)$ is the initial state distribution. In reinforcement learning (RL), we are typically concerned with optimizing or estimating the value of a policy π in a policy space Π. Specifically, value is defined as:

$$V^\pi = \mathbb{E}_{s_0 \sim \rho_0, s_{1:\infty}, a_{0:\infty} \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]. \quad (1)$$

For a fixed policy π, the MDP becomes a Markow chain, and we define the occupancy measure $\rho^\pi(s, a) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}_\pi(s_t = s, a_t = a)$. Then the policy value can be rewritten as $V^\pi = \mathbb{E}_{s,a \sim \rho^\pi_{T^*}}[r(s, a)]$. When different dynamics are involved, we use an additional subscript to indicate the transition, e.g. $V^\pi_{T^*}$ and $V^\pi_{\hat{T}}$. For a learned dynamics model $\hat{T}$, we are interested in the value difference $|V^\pi_{T^*} - V^\pi_{\hat{T}}|$ for the policy π.

2.2. Off-policy Evaluation

Off-policy evaluation (OPE) (Le et al., 2019; Precup et al., 2000; Jiang & Li, 2016; Kostrikov & Nachum, 2020; Yang et al., 2020; Wen et al., 2020) aims at estimating the value V^π of a target policy π, based on a fixed dataset of transitions $\mathcal{D}$ collected from some *behavior policies* $\{\mu_i\}_{i=1}^n$ (or named *data-collection policies*). This problem is of great practical significance for several reasons, including providing high-confidence guarantees prior to deployment, performing policy improvement, and model selection. A major challenge in OPE is the distribution shift between the behavior policy and the target policy, which induces a large value gap between the estimated value and the true value.

3. Related Works

Off-policy Evaluation (OPE): OPE research is relevant to many practical domains such as recommendation systems (Li et al., 2011), health (Liao et al., 2019), and education (Mandel et al., 2014). There exists a large body of work on OPE, including methods based on fitted q-evaluation (Le et al., 2019; Hao et al., 2021) and importance sampling (Kostrikov & Nachum, 2020). Another class of OPE is model-based approachs (also referred to as the direct method), which is focused on in this paper. While model-based OPE has been considered by many previous works (Thomas & Brunskill, 2016; Hanna et al., 2017), they are confined to simple tasks and produce biased predictions due to the restricted range of state and action space in offline trajectories (Fu et al., 2021b). By contrast, our approach is applied to more intricate tasks and proves that model-based OPE can also do well in challenging continuous tasks.

Model Learning Should be Aware of Policies: There are some previous works in other fields also proposing the idea that the dynamics model should be aware of or focus on certain policies rather than all the policies. PAML (Abachi et al., 2020) proposes that model learning should incorporate the way the planner is going to use the model. PDML (Wang et al., 2022) dynamically adjusts the historical policy mixture distribution to ensure the learned model can continually adapt to the state-action visitation distribution of the evolving policy. However, in contrast to us, their policy awareness is aimed at facilitating online policy learning, adapting to the policy gradient estimate, or simply the most recently discovered policies. Recently, GALILEO (Chen et al., 2023c) adjust the weights of the data points in the offline model learning, where data is collected from one single and biased behavior policy, to make the model iteratively focus on the implicitly adversarial policies that the model cannot predict well at the moment. Our method considers accurate offline dynamics reconstruction utilizing an explicit policy representation module, enabling on-the-fly adaptation for target policy evaluation to improve local accuracy directly.

Model-based Offline RL: Model-based Offline RL (MBORL) algorithms also involve dynamics models for some downstream tasks. From the perspective of model usage, MBORL can generally be categorized into two groups: model predictive control (MPC) (Camacho & Alba, 2013) and Policy learning (PL). In MPC, Argenson & Dulac-Arnold (2021) directly performs planning in a learned dynamics model. In Policy Learning, a policy can be trained either in an in-support region by utilizing a conservative surrogate MDP (Yu et al., 2020; Kidambi et al., 2020; Yu et al., 2021; Sun et al., 2023; Chen et al., 2024a), or in out-of-policy regions by learning an adaptive policy (Chen et al., 2021; 2023b;a; Qin et al., 2023). Some works also utilize dynamics models with off-the-shelf model-free algorithms for better policy learning (Lyu et al., 2022; Wang et al., 2021). Recent studies (Rigter et al., 2022; Yang et al., 2022) also adopt an adversarial framework that alternates between dynamic-model training and policy learning. However, these works pay more attention to optimizing policy under a restricted dynamics model instead of directly learning a faithful model when using it, where the latter is what our work focuses on.

4. Value Gaps Formulation between True Dynamics and a Learned Model

This work focuses on the generalization of the dynamics model for the offline evaluation, which requires the model prediction to be as accurate as possible even beyond the support of the dataset to achieve a small evaluation error. [1].

We first give the problem formulation and the metric to evaluate the gap between true dynamics and a learned model. Offline dataset $D = \{\tau_m\}_{m=1}^M$ consists of previously collected trajectories $\tau_m = (s_0, a_0, r_0, s_1, \dots)$, each of which is generated by the interaction between one of the behavior policies $\Omega = \{\mu_i | i \in \mathcal{I}\}$ and the environment.

We follow the basic idea in OPE to define the performance metric of a dynamics model: in an MDP, a good dynamics model means for any target policy π, the gap between the value under true transition T^* and the value estimation under $\hat{T}$ is small, i.e., $|V^\pi_{T^*} - V^\pi_{\hat{T}}|$ is small. Inspired by a previous study (Janner et al., 2019), the value gaps between true dynamics and a learned model is bounded by

$$|V^\pi_{T^*} - V^\pi_{\hat{T}}| \le \frac{2\gamma R_{\max}}{(1-\gamma)^2} l(\pi, T^*, \hat{T}), \tag{2}$$

where $l(\pi, T^*, \hat{T}) = \mathbb{E}_{s,a\sim\rho^\pi} D_{\mathrm{TV}}(T^*(\cdot|s,a), \hat{T}(\cdot|s,a))$ is total variation between true and learned transitions under the state-action distribution of the target policy π to measure the model error. Eq. (2) implies that as long as we *reduce the model error* $l(\pi, T^*, \hat{T})$ *under the target policy's distribution* ρ^π, we can guarantee the reduction of the corresponding upper bound of the value gap. The full derivation is in App. A.1.

Remark 1 (Comparison to Janner's bound): The bound is a variant of the previous bounds in Janner et al. (2019); Xu et al. (2020; 2021), where we consider the generalization ability of learned models. We adapt the bound from Janner et al. (2019) to our multiple behavior policies setting for a more specific comparison:

$$|V^\pi_{T^*} - V^\pi_{\hat{T}}| \le \frac{2\gamma R_{\max}}{(1-\gamma)^2} l(\mathcal{D}, T^*, \hat{T}) + \frac{4R_{\max}}{(1-\gamma)^2} \sum_{i=1}^n w_i \max_s D_{TV}(\pi(\cdot|s), \mu_i(\cdot|s)), \tag{3}$$

where $l(\mathcal{D}, T^*, \hat{T}) = \mathbb{E}_{s,a\sim\mathcal{D}} D_{\mathrm{TV}}(T^*(\cdot|s,a), \hat{T}(\cdot|s,a))$ is the expected error under the training data distribution $\mathcal{D}$. Equation (3) bounds the value gap by the training model error and the additional terms of the divergence between the target policy and behavior policies μ_i, *which is in fact a relaxation of Equation* (2). Although it is still a valid bound, this result *only suggests minimizing the training model error* but ignores the dynamics model's generalization ability to the target policy since the unavoidable policy divergence term is irrelevant to the quality of learned dynamics models.

In contrast to Janner's bound, Equation (2) requires focusing on *the model error under target distribution*, suggesting that an improved model generalization ability could further reduce the value gaps for target policies. This drives us to study model adaptation ability for diverse policies.

5. Policy-conditioned Dynamics Model Learning

In this section, we first give intuitions for policy-conditioned model (PCM) (Sec. 5.1). Then we give a formal formulation of PCM learning and a practical policy representation technique to achieve PCM learning (Sec. 5.2). Finally, we give an analysis of PCM in the generalization ability (Sec. 5.3).

5.1. Intuition for Policy-Conditioned Model Learning

In Fig. 1, we use an example to illustrate why policy-conditioned model (PCM) learning is superior to policy-agnostic model (PAM) learning.

Suppose we wish to learn an environment model where a biped robot is asked to move forward from an offline dataset including different locomotion patterns, such as walking, running, jumping, etc. Currently, the standard dynamics model, i.e., the policy-agnostic model (PAM), learns to predict all of the transitions coming from different locomotion patterns in a unified model. However, we notice that different locomotion patterns usually correspond to quite different transition patterns though these patterns can be regarded as a single task. For instance, jumping requires both legs to be folded and unfolded at the same time while running involves alternate flexion and extension of the legs. If we can utilize this nature, the learning complexity will be reduced.

Based on the above motivation, instead of learning a single model for the whole dataset, we propose to "divide" the dataset according to the data-collection policy and learn a

[1]It is noteworthy the notion of generalization here has subtle differences from that in online policy learning case, where policy divergence is much smaller and exploration also have impacts.

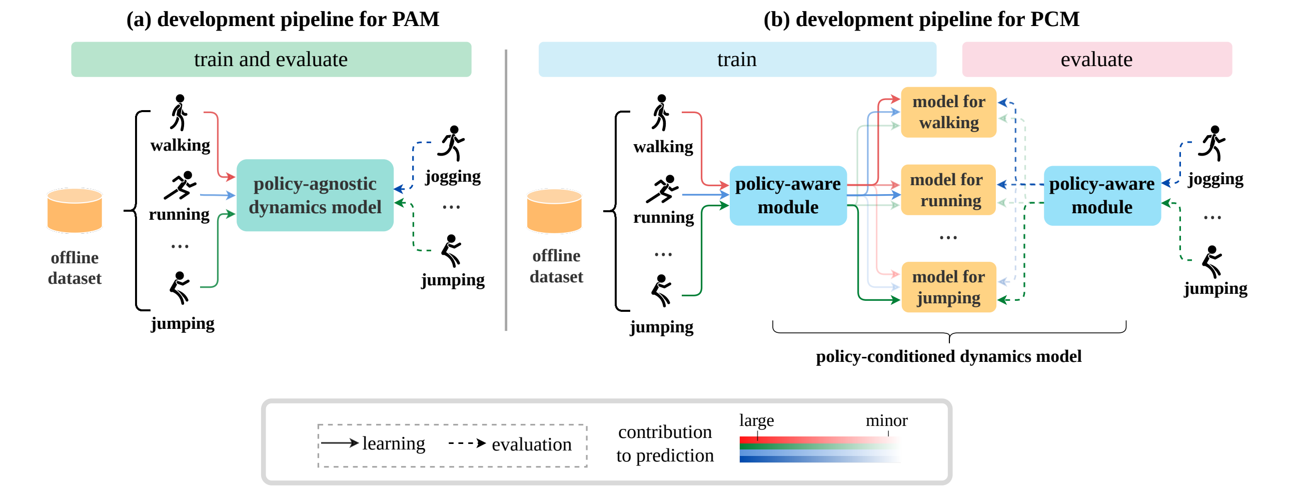


Figure 1: An illustration of the difference between the policy-agnostic model (left) and the policy-conditioned model (right). Suppose we wish to learn an environment where a biped robot is asked to move forward from an offline dataset including different locomotion patterns, such as jumping, walking, running, etc. Different locomotion patterns usually correspond to quite different transition patterns even though they can be regarded as a single task.

model for each subset. We regard each locomotion pattern as a subtask and respectively learn a model for each subtask. In this way, we can reduce the learning difficulty of each model, which is expected to obtain a more accurate model for each data-collection policy. The rationale behind this is that each data-collection policy only focuses on a relatively small subregion of the support set of the whole mixed state-action distribution, thus training the model under the state-action occupancy of each policy should be an easier task than the global model training and tends to obtain more accurate models. Moreover, if the target policy to be evaluated is unseen before in the dataset, e.g. jogging, which is a locomotion pattern between walking and running, it is hoped to yield a new model to adapt to the jogging policy by combining the walking model and the running model.

5.2. Achieve PCM Learning via Policy Representation Adaptation

For the dataset D collected by a set of behavior policies $\Omega = \{\mu_i | i \in \mathcal{I}\}$, the training data distribution $\mathcal{D}$ is a mixture of occupancy measures $\sum_{i\in\mathcal{I}} w_i \rho^{\mu_i}(s,a)$, where w_i is data proportion of policy μ_i. Conventional model learning fits a universal transition model directly under the whole mixed data distribution and rolls out whatever target policy in this *policy-agnostic model (PAM)*:

$$\hat{\psi} = \arg\min_{\psi\in\Psi} \sum_{\mu_i\in\Omega} w_i l(\mu_i, T^*, T_\psi), \tag{4}$$

where model T is parameterized by $\psi \in \Psi$.

The objective is sufficient for simple environments where the model capacity is rich enough to completely recover true transitions. However, in realistic large-scale tasks, the model's capacity is limited in comparison to true transition, resulting in a non-zero error, which will be further compounded during long-horizon rollout (Janner et al., 2019; Xu et al., 2020). With an adequate model capacity, it is possible to accurately fit true transition dynamics, which is the *unique* optimal model for any target policy. Nevertheless, the usually limited model capacity prevents perfect transition modeling and requires a proper allocation of the finite accuracy budget to facilitate the target policy rollout as much as possible. Since different policies perform distinct behaviors and access varied subregions of the state-action space, their optimal models within the model space are different, resulting in an optimal model *inconsistency*, i.e., there does not exist a unique model within the model space that is optimal for general target policies.

Our consequent idea is to select dynamics models adaptively for different policies, where each model is optimized specially for the occupancy measure of its corresponding policy. We name it *policy-conditioned model (PCM)*. This "model selection" procedure can be expressed through a mapping $F : \Pi \to \Psi$, where each policy π is associated with a model $T_{F(\pi)}$. Learning a PCM is therefore translated into finding an optimal F to minimize model error on the data distribution of each policy:

$$\hat{F} = \arg\min_{F\in\mathcal{F}} \sum_{\mu_i\in\Omega} w_i l(\mu_i, T^*, T_{F(\mu_i)}), \tag{5}$$

where $\mathcal{F}$ is function space of F. The difference is that PAM attempts to fit global transition dynamics, which is more difficult than the local transition modeling that PCM specializes in. Thus, for the behavior policies μ_i, model error $l(\pi, T^*, T_{\hat{F}(\mu_i)})$ can be reduced to achieve smaller value gaps compared to PAM. Sec. 6.1 empirically showcases the reduced training errors.

However, in real-world applications, the corresponding white-box policies are typically unknown. It is impractical to learn a mapping function $F(\pi)$ which directly takes policy π as the input. Inspired by many previous works (Duan et al., 2016; Chen et al., 2021; Nagabandi et al., 2019) which have successfully utilized RNN as an extra representation extractor module to map the interaction trajectories to some task-specific meta-parameters, we use similar RNN structure to learn and infer policy representations from given interaction trajectories, and a policy-representation-conditioned dynamics model is learned to adapt its predictions based on the input policy representation. Formally, let $\tau_{0:t} = (s_0, a_0, s_1, a_1, ..., s_t, a_t)$ be a trajectory generated by a data-collection policy up to timestep $0 \le t \le H-1$ (H is the horizon of the MDP) and the offline dataset is a set of N trajectories $\mathcal{D} = \{\tau^{(j)}\}_{j=1}^N$. For any timestep t, trajectories $\tau_{0:t-1}$ will be fed into a RNN $q_\phi(\tau_{0:t-1})$ to obtain an embedding z_t. After that, an adaptive dynamics model $T_\psi(s_{t+1}|s_t, a_t, z_t)$ is learned to adapt its predictions of s_{t+1} based on z_t. Recall that we expect to get a policy representation, the embedding z_t should encode salient information about the policy. To this end, a policy representation regularizer $\mathcal{R}_\pi$ is jointly optimized to reconstruct the specified policy. The overall learning objective of PCM is:

$$\min_{\phi,\theta,\psi} \mathbb{E}_{t\sim[0,H-2],\tau^{(j)}_{0:t+1}\sim\mathcal{D}}[-\log T_\psi(s_{t+1}|s_t, a_t, q_\phi(\tau^{(j)}_{0:t-1})) \\ - \lambda\mathcal{R}_\pi(q_\phi(\tau^{(j)}_{0:t-1}), \pi^{(j)}, \theta)], \tag{6}$$

where λ is a hyperparameter for policy-representation regularization. Since $\pi^{(j)}$ unknown in prior, in this work, we directly use the policy reconstruction term in implementation, i.e., $\mathcal{R}_\pi(q_\phi(\tau^{(j)}_{0:t-1}), \pi^{(j)}, \theta) = \log p_\theta(a_t|s_t, q_\phi(\tau^{(j)}_{0:t-1}))$, where $p_\theta(a_t|s_t, z_t)$ is the jointed optimized policy decoder. Note that the gradients would be backpropagated from T_ψ and p_θ to z if optimal models' or policies' parameters in different trajectories are inconsistent but have the same representation of z, then the parameters of ϕ will be updated automatically to distinguish them. Our pseudo-code of the overall PCM learning is shown in Alg. 1. Besides, since the input trajectories can last for hundreds of steps in most RL tasks, which has proven to raise challenges for the RNN training due to gradient vanishing (Le & Zuidema, 2016), we incorporate the residual connection into the policy encoder and dynamics model, referring to the previous success of ResNet (He et al., 2016). Two tricks for modern autoregressive model training (Vaswani et al., 2017; Janner et al., 2021), layer normalization and dropout, are also incorporated to improve RNN representation stability.

5.3. Analysis of PCM learning and its implementation

In this section, we show the adaptation effect from PCM can provide additional generalization benefits when the learned model extrapolates to the data distribution of new target policies absent from training dataset. We introduce an assumption on the smoothness of well-trained models:

Assumption 5.1. For the learned model T, the point-wise model error $D_{\mathrm{TV}}(T^*(\cdot|s,a), T(\cdot|s,a))$ is L-Lipschitz with respect to the state-action pairs, i.e.,

$$\Big|D_{\mathrm{TV}}(T^*, T)(s_1, a_1) - D_{\mathrm{TV}}(T^*, T)(s_2, a_2)\Big| \\ \le L \cdot D\big((s_1, a_1), (s_2, a_2)\big), \tag{7}$$

where $D(\cdot,\cdot)$ is some kind of distance defined on the state-action space $\mathcal{S} \times \mathcal{A}$.

Assump. 5.1 measures the local generalization ability of a learned model, common in previous works (Tang et al., 2022). Generally speaking, if we say the learned model $T_{\hat{\psi}}$ generalizes well w.r.t. the state-action inputs, we mean that for some unseen (s_2, a_2) deviating from a training data (s_1, a_1), the point-wise model error will not increase much, reflected by a bounded L. Based on this assumption, we find that the expected model error of PAM under the target policy data distribution can be controlled:

Proposition 5.2. *Under Assump. 5.1, for any policy $\pi \in \Pi$, the model error of PAM $T_{\hat{\psi}}$ under target policy-induced distribution can be bounded:*

$$l(\pi, T^*, T_{\hat{\psi}}) \le \min_{\mu_i \in \Omega} \big\{ l(\mu_i, T^*, T_{\hat{\psi}}) + L \cdot W_1(\rho^\pi, \rho^{\mu_i}) \big\}, \tag{8}$$

where W_1 is Wasserstein-1 distance defined on state-action distribution space $\mathcal{P}(\mathcal{S} \times \mathcal{A})$ with underlying metric D.

Prop. 5.2 shows that the generalization of PAMs to the target policy solely relies on their point-level smoothness over the state-action inputs. PCMs instead take a further step to reduce the test model error via the policy adaptation:

Proposition 5.3. *Under Assump. 5.1, for any policy $\pi \in \Pi$, model error of PCM $T_{\hat{F}(\pi)}$ is bounded:*

$$l(\pi, T^*, T_{\hat{F}(\pi)}) \le \\ \min_{\mu_i \in \Omega} \Big\{ \underbrace{l(\mu_i, T^*, T_{\hat{F}(\mu_i)})}_{\text{training error}} + \underbrace{L \cdot W_1(\rho^\pi, \rho^{\mu_i}) - C(\pi, \mu_i)}_{\text{generalization error}} \Big\}, \tag{9}$$

where adaptation gain $C(\pi, \mu_i) := l(\pi, T^*, T_{\hat{F}(\mu_i)}) - l(\pi, T^*, T_{\hat{F}(\pi)})$ summarizes the adaptation effect.

Prop. 5.3 explains the PCM's advantages over PAM: 1) a smaller model error on the training dataset (as discussed in Sec. 5.2); 2) a positive adaptation gain $C(\pi, \mu_i)$, where depict the benefit of the policy adaptation effect based on the insight that when testing on a new policy π within some effective region, the model $T_{\hat{F}(\pi)}$ customized for π should have a smaller model error under the target distribution ρ^π than any $T_{\hat{F}(\mu_i)}$. PAM does not include the policy-conditioned mechanism and the adaptation gain is always zero. Note that fine-tuning the PAM parameters for a new policy is not practical because it requires the interaction of the new policy with the environment to collect the target domain experiences, which is prohibitive in general. In contrast, the policy representation serves as an additional covariate in PCM, which enables an extra adaptation ability to target policies and hence a non-zero adaptation gain term with no need for the real experiences in the target domain. Therefore, Prop. 5.3 shows that the model error of PCM for a new target policy π is reduced by the adaptation gain C, if $C > 0$, compared with PAM (refer to Prop. A.2).

However, it is still difficult in general to rigorously analyze the adaptation gain $C(\pi, \mu_i)$ due to the complexity of neural networks and the optimization process. We discuss several cases of adaptation effects and extrapolation errors in App. A.3. Empirically, as the target policy π gradually diverges from Ω, the adaptation gain will increase from zero and partially reduce the extrapolation error within an effective adaptation region. When π leaves far enough from Ω, C will reach the maximum and then start to decrease. This trend exhibits the efficacy of policy adaptation to a reasonable degree. We also provide experimental evidence in Sec. 6.1.2, which aligns with the intuition.

Remark 2 (Model complexity): In our implementation in Sec. 5.2, we introduce additional model complexity by using the RNN module compared to the model space of PAM, which is a gap between the analysis and the practical algorithm. One may suspect that it is the increased model capacity helps the dynamics model learning, instead of the policy-conditioned mechanism. We eliminate the concerns through experiments. In particular, we find that simply increasing the model capacity by using a larger network without mechanism changes cannot bring significant improvement (as shown in Table 8 in Appendix). The additional module works mainly because it allows an adaptive and therefore effective utilization of the limited capacity for different target policies, which reduces the in-distribution model error and also brings generalization benefits for new target policies as we show in the next subsection.

6. Experiment

In this section, we justify the efficacy of the policy adaptation mechanism for model learning via a proof-of-concept experiment (Sec. 6.1). In Sec. 6.1.2, we conduct experimental studies to verify PCM enjoys smaller value gaps as analyzed in Sec. 5.3. Then we evaluate PCM on specific downstream tasks including off-policy evaluation (OPE), offline policy selection (OPS), and model predictive control (MPC) (Sec 6.2). Finally, we conduct ablation studies in Sec. 6.3 and verify whether PCM learns reasonable policy representation via visualization (Sec 6.4) [2].

6.1. Proof-of-Concept Verification

Prop. 5.3 indicates that the value gap of PCM for an unseen policy π can be reduced by 1) a smaller model error on the training dataset; 2) a positive adaptation gain C. In this section, we conduct several proof-of-concept experiments to verify these statements.

6.1.1. Reducing Training Errors

We consider a simplified setting that does not involve generalization to unseen policies to justify the idea of the policy adaptation mechanism for model learning. We collect a dataset sampled by 10 different policies in HalfCheetah and solely choose one of the 10 policies for evaluation. Since there is no need for generalization, we can use a simple policy representation scheme called *vector policy embedding*, $F(\mu_i)$. Specifically, we employ a $n \times m$ matrix to represent the policies, where n is the number of policies in the dataset and m is the dimension of the policy representation. The matrix can be updated by backpropagation. We compared the performance of the model with and without embedding. Fig. 2(a) and 2(b) show even with such a simple policy representation scheme, PCM can significantly outperform PAM on the model error as well as the value gap.

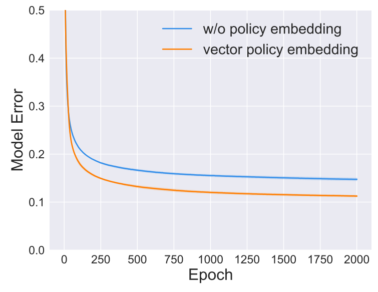
Furthermore, we show that the vector policy embedding indeed helps the model adapt to a specific policy. We first train and obtain an embedding for each policy, After training, we have 10 different vector policy embeddings for these 10 policies, respectively. Then we evaluate each policy under models given different vector embeddings and record the value gap under each case. The results are shown in the mismatch heatmap below. Fig. 2(c) shows that the model performs better under policy with better-matched embedding (for any two policies, the closer their numbers are, the more similar they are), indicating that the vector policy embedding helps the model adapt to a specific policy.

6.1.2. Having Positive Adapatation Gain

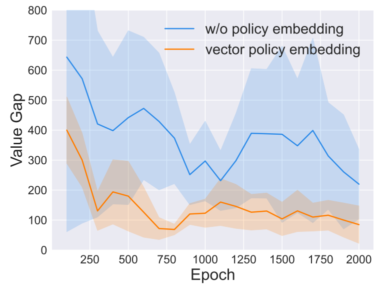
We now further verify that PCM indeed has adaptation gain and smaller value gaps in unseen policies. All experiments in this section are conducted in HalfCheetah.

We analyze the adaptation gain quantitatively by fixing a data-collection policy μ_i and computing $C(\pi, \mu_i)$ for different policies π. We refer to Appx. E.2 for more details. As illustrated in Fig. 3(a), the gain gradually increases with the policy divergence, reaches a maximum, and decreases as the policy divergence continues increasing. This confirms

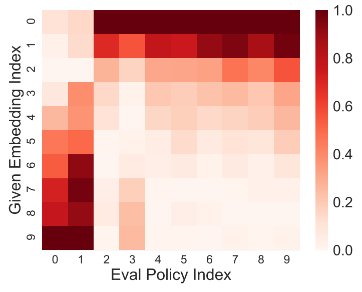
[2] code: https://github.com/xionghuichen/policy-conditioned-model.git



(a) Model error



(b) Value gap

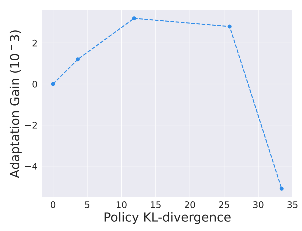


(c) Heatmap

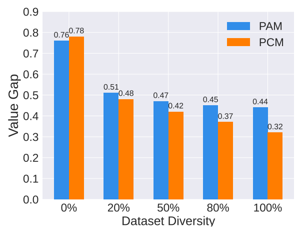
Figure 2: Illustration of model error, value gap of policy learned in with or w/o policy embedding model, and the heatmap about the performance of evaluating policy with different policy embedding.

the analyzed cases in Sec. 5.3.

Finally, we directly compare the value gaps of PAM and PCM and also investigate the influence of different levels of dataset diversity on them. To do so, we construct datasets with varying levels of diversity (0%, 20%, 50%, 80%, 100%), where the percentage indicates that the dataset is created from the replay buffer of SAC (Haarnoja et al., 2018) until the policy reaches the specific level of performance.Appx. E.1 presents details of the data collection process. We train PAM and PCM on each dataset and test them on other 11 policies provided by the DOPE benchmark (Fu et al., 2021a), which were unseen before in the datasets. Fig. 3(b) depicts value gaps of each model trained on each dataset, demonstrating that PCM achieves smaller value gaps. Moreover, the results show that as the diversity of the dataset increases, both PAM and PCM achieve smaller value gaps, with PCM exhibiting a more substantial advantage.



(a) Adaptation gain



(b) Value gaps

Figure 3: Illustrations of PCM having smaller value gaps. The left shows the adaptation gain of PCM for different unseen policies π, relative to a data-collection policy μ_i. The right shows normalized value gaps of PAM and PCM trained on datasets with different levels of diversity when testing on 11 target unseen policies.

6.2. Evaluation on Downstream Tasks

6.2.1. Off-policy Evaluation

We compare PCM with several OPE methods, including: **Fitted Q-Evaluation (FQE)** (Le et al., 2019), that estimates the policy value via iteratively performing Bellman update, **Doubly Robust (DR)** (Jiang & Li, 2016), that combines the importance sampling technique with a value estimator for variance reduction, **Importance Sampling (IS)** (Kostrikov & Nachum, 2020), that performs importance sampling with a learned behavior policy, **DICE** (Yang et al., 2020), that uses a saddle-point objective to estimate marginalized importance weights $d^{\pi}(s,a)/d^{\pi_B}(s,a)$, **Variational Power Method (VPM)** (Wen et al., 2020), that runs a variational power iteration algorithm to estimate the importance weights without the knowledge of the behavior policy, **Policy-Agnostic Model (PAM)**, that removes the policy representation module in PCM and serves as the ablation method. We evaluate these approaches on a variety of tasks from DOPE-D4RL and DOPE-RL-Unplugged benchmarks (Fu et al., 2021a), where the data in these tasks is collected by diverse policies.

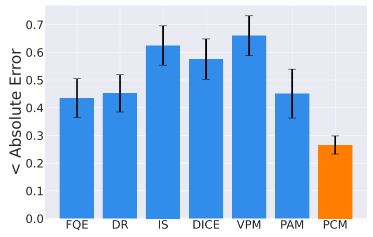
Fig. 4 shows the performance of PCM and other methods in three metrics (details of the metrics and results separated by tasks are in App. C). We find that PCM outperforms other methods by a large margin. Specifically, the results of the absolute error provide direct evidence that PCM can reduce the value gap effectively. Besides, PCM obtains a higher rank correlation and lower regret, indicating that PCM can not only perform accurate evaluation but also select the competitive policies among the policies to be evaluated.

6.2.2. Offline Policy Selection

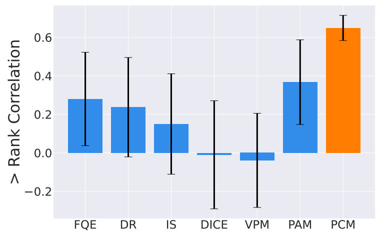
In this section, we explore the efficacy of using PCM on offline policy selection (OPS) for a practical offline RL algorithm. Specifically, we use the implementation in OfflineRL-Kit (Sun, 2023) to train MOPO (Yu et al., 2020) for 1000 epochs and record policy snapshots at the latest 20 epochs for OPS. MOPO tends to give policies with different performance even near the end of the training as shown in Tab. 10, suitable for OPS tasks and comparing different methods. We compare our method against PAM and FQE as well as directly selecting the last-epoch policy. Tab. 1 shows the performance gains by different methods, computed by $\frac{(V_{\text{selected}}-\bar{V})}{V_{\max}-\bar{V}} \times 100\%$, where V_{selected} is the value of the se-

Table 1: Performance gain of offline policy selection for MOPO (Yu et al., 2020) by different methods.

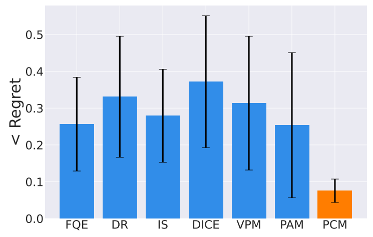
Task Name	Last Epoch	FQE	IS	DICE	PAM	PCM (Ours)
halfcheetah-medium-replay	39.3%	23.0%	87.8%	1.6%	1.6%	**98.4%**
hopper-medium-replay	56.0%	34.1%	56.0%	19.8%	47.3%	**64.8%**
walker2d-medium-replay	-4.6%	4.6%	34.3%	13.0%	-30.6%	**51.9%**
Average	30.2%	20.6%	59.4%	39.3%	11.5%	**71.7%**



(a) Absolute error



(b) Rank correlation



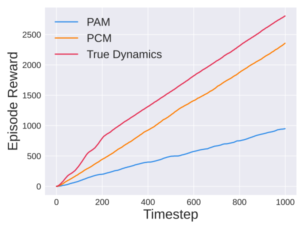
(c) Regret

Figure 4: The performance of OPE in three metrics. To aggregate across tasks, we normalize the real policy values and evaluate policy values to range between 0 and 1. The error bars denote the standard errors among the tasks with three seeds.

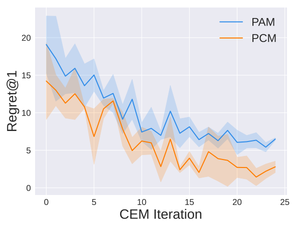
lected policy and $\bar{V}, V_{\max}$ are the average and max values of the evaluated policies, respectively. It is noteworthy that the gains of FQE and PAM are even lower than directly selecting the last-epoch policy, also indicated in (Qin et al., 2022). In contrast, our approach shows a brilliant performance, implying that it reliably chooses a better policy for an offline RL algorithm to deploy. The results of PAM and PCM on more tasks with more repetitions can be found in Tab. 12 in Appendix.

6.2.3. Model Predictive Control

An accurate model can also be expected to perform effective model predictive control (MPC). We therefore compare our proposed PCM against PAM and the true dynamics (using MuJoCo simulator itself as true dynamics). Following Chua et al. (2018), we use the cross-entropy method (CEM) as the optimization technique in MPC, which samples actions form a distribution closer to previous action samples yielding high rewards. Details on MPC and CEM are discussed in App. F.



(a) Performance



(b) Regret

Figure 5: Left shows cumulative rewards within an episode in HalfCheetah. Right shows regrets of PAM and PCM during CEM, obtained by tracking several planning processes.

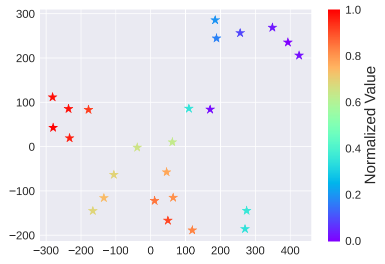
Fig. 5(a) shows the cumulative rewards of the three methods during an episode, from which we can see that PCM performs similarly to the true dynamics and significantly outperforms PAM. To further explore why our approach works better, we calculate regret values of the evaluation of action sequences for PCM and PAM respectively. We track several planning processes and compute regret $\sum_{i=t}^{t+T} \mathbb{E}_{T^*}[r(s_i, a_i^*)] - \sum_{i=t}^{t+T} \mathbb{E}_{T^*}[r(s_i, \hat{a}_i)]$ for both PAM and PCM, where $\hat{a}_{t:t+T}$ and $a^*_{t:t+T}$ are the optimal action sequences selected by the learned model and true dynamics respectively. Regret is the difference between the real value of the action sequence selected by the model and the value of the optimal action sequence. Results in Fig. 5(b) shows that PCM has lower regret than PAM, meaning that our approach tends to pick out actions that are closer to the optimal policy.

6.3. Ablation Studies

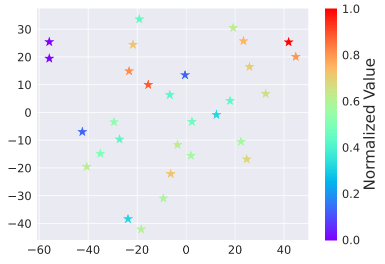
We do ablation studies to analyze the effect of the model architecture and implementation tricks mentioned in Sec. 5.2. The results are shown in Tab. 2. Specifically, we investigate the effect of residual connection (PCM-res) and two related tricks borrowed from transformer (Janner et al., 2021; Vaswani et al., 2017): layer normalization (PCM-layer) and dropout (PCM-drop), removing them from the dynamics model. We find that removing the layer normalization and dropout leads to no significant performance change. The residual connection is relatively more important mainly because it relieves the gradient vanishing for RNN-based policy encoder training. Removing the reconstruction loss for policy representations (PCM-repr) presents significant per-

Table 2: Normalized absolute error for ablations of implementation components in PCM.

Task Name	**PCM**	**PCM-layer**	**PCM-drop**	**PCM-res**	**PCM-repr**	**PAM+GRU**	**PAM**
halfcheetah-medium-replay	0.31±0.08	0.37±0.05	0.32±0.04	0.43±0.03	0.47±0.06	0.48±0.12	0.53±0.04
hopper-medium-replay	0.08±0.02	0.12±0.01	0.10±0.02	0.14±0.01	0.22±0.04	0.15±0.05	0.34±0.04
walker2d-medium-replay	0.25±0.05	0.21±0.05	0.29±0.02	0.30±0.04	0.39±0.07	0.42±0.03	0.66±0.16
Average	**0.21**	0.23	0.24	0.29	0.36	0.37	0.51



(a) with representation loss



(b) without representation loss

Figure 6: Visualization for policy representations of different policies learned by PCM in HalfCheetah. Points are colored according to the normalized value.

formance degradation, which demonstrates the efficacy of PCM mainly comes from our policy-embedded mechanism for the dynamics model learning instead of just the architecture advantages. Replacing the MLP in PAM with GRU cells used in the PCM implementations (PAM+GRU) still underperforms our PCM, though the network architecture indeed benefits.

6.4. Analysis of Learned Policy Representation

In this section, we conduct a study to verify whether the PCM can learn reasonable policy representations. We select several policies with different performance and feed the trajectories generated by these policies into the policy encoder module of PCM. We visualize the outputted policy representations via the t-SNE (van der Maaten & Hinton, 2008) technique in Fig. 6(a). We find that the policies with similar performance have similar policy representations since there is a degree of resemblance between their performed actions, while the representations of policies with widely different performance are far apart due to their quite different behavior. In contrast, the representations without the policy reconstruction loss are randomly distributed as shown in Fig. 6(b). This result demonstrates that PCM can effectively identify similar policies and distinguish different policies. We provide results on more tasks in Appx. D.

7. Discussion

This paper handles the challenge that learned dynamics models tend to have large value gaps when evaluating a target policy different from the behavior policies if the offline dataset is collected by diverse behavior policies. We propose training a Policy-Conditioned Model (PCM) that generates distinct dynamics models based on different target policies. We demonstrate that PCM can achieve smaller value gaps by reducing training errors and better generalization to out-of-distribution data. Empirical results across domains and algorithms validate the superiority of our approach.

It should be noted that several possible ways exist to implement the policy-conditioned mechanism, and the RNN-based policy encoding employed in this work is just one of them. A limitation is that we analyze the generalization of PAM and PCM based on the infinite sample assumption for each behavior policy. However, for realistic situations where only finite samples are available, the data from each policy are limited, and additional estimation errors occur in the model learning, which requires further analysis to compare PAM with PCM. In the future, we aim to find a more efficient policy representation scheme to enhance the model's generalization ability.

Another limitation is that we mainly focus on the model generalization ability in the offline policy evaluation scenario, and it has subtle differences from that in online and reinforcement learning scenarios. The divergence between the evaluation policies and the behavior policies within the static offline dataset is usually much larger than that in online policy learning case, where the experience data are continually collected by evolving policies, and therefore the our-of-distribution generalization is more essential in offline scenarios. Besides, the exploration effect probably makes some kinds of inaccurate models more beneficial in long-term policy optimization than an accurate one, which is more complicated and beyond our consideration.

Acknowledgements

This work is supported by National Science Foundation of China (61921006). We thank the anonymous reviewers for their helpful discussions and suggestions on improving the paper.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Abachi, R., Ghavamzadeh, M., and Farahmand, A. Policy-aware model learning for policy gradient methods. *CoRR*, abs/2003.00030, 2020.

Argenson, A. and Dulac-Arnold, G. Model-based offline planning. In *9th International Conference on Learning Representations (ICLR'21)*, virtual event, 2021.

Camacho, E. and Alba, C. *Model Predictive Control*. Advanced Textbooks in Control and Signal Processing. Springer London, 2013. ISBN 9780857293985.

Chen, R., Jia, C., Huang, Z., Liu, T.-S., Liu, X.-H., and Yu, Y. Offline transition modeling via contrastive energy learning. In *Forty-first International Conference on Machine Learning*, 2024a. URL https://openreview.net/forum?id=dqpg8jdA2w.

Chen, R., Liu, X.-H., Liu, T.-S., Jiang, S., Xu, F., and Yu, Y. Foresight distribution adjustment for off-policy reinforcement learning. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS'24)*, Auckland, New Zealand, 2024b.

Chen, X., Yu, Y., Li, Q., Luo, F., Qin, Z. T., Shang, W., and Ye, J. Offline model-based adaptable policy learning. In *Advances in Neural Information Processing Systems 34 (NeurIPS'21)*, virtual event, 2021.

Chen, X., He, B., Yu, Y., Li, Q., Qin, Z. T., Shang, W., Ye, J., and Ma, C. Sim2rec: A simulator-based decision-making approach to optimize real-world long-term user engagement in sequential recommender systems. In *39th IEEE International Conference on Data Engineering (ICDE'23)*, Anaheim, CA, 2023a.

Chen, X., Luo, F., Yu, Y., Li, Q., Qin, Z., Shang, W., and Ye, J. Offline model-based adaptable policy learning for decision-making in out-of-support regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45 (12):15260–15274, 2023b.

Chen, X., Yu, Y., Zhu, Z., Yu, Z., Chen, Z., Wang, C., Wu, Y., Qin, R., Wu, H., Ding, R., and Huang, F. Adversarial counterfactual environment model learning. In *Advances in Neural Information Processing Systems 36 (NeurIPS'23)*, New Orleans, LA, 2023c.

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems 31 (NeurIPS'18)*, Montréal, Canada, 2018.

Clavera, I., Rothfuss, J., Schulman, J., Fujita, Y., Asfour, T., and Abbeel, P. Model-based reinforcement learning via meta-policy optimization. In *Proceedings of The 2nd Conference on Robot Learning (CoRL'18)*, Zürich, Switzerland, 2018.

Doroudi, S., Thomas, P. S., and Brunskill, E. Importance sampling for fair policy selection. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence (UAI'17)*, Sydney, Australia, 2017.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. RL^2: Fast reinforcement learning via slow reinforcement learning. *CoRR*, abs/1611.02779, 2016.

Fu, J., Norouzi, M., Nachum, O., Tucker, G., Wang, Z., Novikov, A., Yang, M., Zhang, M. R., Chen, Y., Kumar, A., Paduraru, C., Levine, S., and Paine, T. Benchmarks for deep off-policy evaluation. In *9th International Conference on Learning Representations (ICLR'21)*, virtual event, 2021a.

Fu, J., Norouzi, M., Nachum, O., Tucker, G., ziyu wang, Novikov, A., Yang, M., Zhang, M. R., Chen, Y., Kumar, A., Paduraru, C., Levine, S., and Paine, T. Benchmarks for deep off-policy evaluation. In *9th International Conference on Learning Representations (ICLR'21)*, virtual event, 2021b.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning (ICML'18)*, Stockholmsmässan, Sweden, 2018.

Hanna, J. P., Stone, P., and Niekum, S. Bootstrapping with models: Confidence intervals for off-policy evaluation. In Singh, S. and Markovitch, S. (eds.), *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI'17)*, San Francisco, CA, 2017.

Hao, B., Ji, X., Duan, Y., Lu, H., Szepesvari, C., and Wang, M. Bootstrapping fitted q-evaluation for off-policy inference. In *Proceedings of the 38th International Conference on Machine Learning (ICML'21)*, virtual event, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*, Las Vegas, Nevada, 2016.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems 32 (NeurIPS'19)*, Vancouver, Canada, 2019.

Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems 34 (NeurIPS'21)*, virtual event, 2021.

Jiang, N. and Li, L. Doubly robust off-policy value evaluation for reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning (ICML'16)*, New York City, NY, 2016.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. Morel: Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems 33 (NeurIPS'20)*, virtual event, 2020.

Kostrikov, I. and Nachum, O. Statistical bootstrapping for uncertainty estimation in off-policy evaluation. *CoRR*, abs/2007.13609, 2020.

Lange, S., Gabel, T., and Riedmiller, M. A. Batch reinforcement learning. In *Reinforcement Learning*, volume 12, pp. 45–73. 2012.

Le, H. M., Voloshin, C., and Yue, Y. Batch policy learning under constraints. *CoRR*, abs/1903.08738, 2019.

Le, P. and Zuidema, W. Quantifying the vanishing gradient and long distance dependency problem in recursive neural networks and recursive lstms. *CoRR*, abs/1603.00423, 2016.

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

Li, L., Chu, W., Langford, J., and Wang, X. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the 4th International Conference on Web Search and Web Data Mining (WSDM'11)*, Hong Kong, China, 2011.

Liao, P., Klasnja, P. V., and Murphy, S. A. Off-policy estimation of long-term average outcomes with applications to mobile health. *CoRR*, abs/1912.13088, 2019.

Liu, X.-H., Xu, F., Zhang, X., Liu, T., Jiang, S., Chen, R., Zhang, Z., and Yu, Y. How to guide your learner: Imitation learning with active adaptive expert involvement. In *Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS'23)*, London, UK, 2023.

Lyu, J., Li, X., and Lu, Z. Double check your state before trusting it: Confidence-aware bidirectional offline model-based imagination. In *Advances in Neural Information Processing Systems 35 (NeurIPS'22)*, New Orleans, LA, 2022.

Mandel, T., Liu, Y., Levine, S., Brunskill, E., and Popovic, Z. Offline policy evaluation across representations with applications to educational games. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'14)*, Paris, France, 2014.

Nagabandi, A., Clavera, I., Liu, S., Fearing, R. S., Abbeel, P., Levine, S., and Finn, C. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *7th International Conference on Learning Representations (ICLR'19)*, New Orleans, LA, 2019.

Precup, D., Sutton, R. S., and Singh, S. Eligibility traces for off-policy policy evaluation. In *Proceedings of the 17th International Conference on Machine Learning (ICML'00)*, Stanford, CA, 2000.

Qin, R.-J., Zhang, X., Gao, S., Chen, X.-H., Li, Z., Zhang, W., and Yu, Y. NeoRL: A near real-world benchmark for offline reinforcement learning. In *Advances in Neural Information Processing Systems 35 (NeurIPS'22, Datasets and Benchmarks)*, New Orleans, LA, 2022.

Qin, T., Wang, T., and Zhou, Z. Rehearsal learning for avoiding undesired future. In *Advances in Neural Information Processing Systems 36 (NeurIPS'23)*, New Orleans, LA, 2023.

Rigter, M., Lacerda, B., and Hawes, N. Rambo-rl: Robust adversarial model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems 35 (NeurIPS'22)*, New Orleans, LA, 2022.

Sun, Y. Offlinerl-kit: An elegant pytorch offline reinforcement learning library. https://github.com/yihaosun1124/OfflineRL-Kit, 2023.

Sun, Y., Zhang, J., Jia, C., Lin, H., Ye, J., and Yu, Y. Model-bellman inconsistency for model-based offline reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning (ICML'23)*, Honolulu, HI, 2023.

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

Tang, H., Meng, Z., Hao, J., Chen, C., Graves, D., Li, D., Yu, C., Mao, H., Liu, W., Yang, Y., et al. What about inputting policy in value function: Policy representation and policy-extended value function approximator. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI'22)*, virtual event, 2022.

Thomas, P. S. and Brunskill, E. Data-efficient off-policy policy evaluation for reinforcement learning. *CoRR*, abs/1604.00923, 2016.

Thomas, P. S., Theocharous, G., and Ghavamzadeh, M. High-confidence off-policy evaluation. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15)*, Austin, TX, 2015.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'12)*, Algarve, Portugal, 2012.

van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86): 2579–2605, 2008.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems 30 (NeurIPS'17)*, Long Beach, CA, 2017.

Wang, J., Li, W., Jiang, H., Zhu, G., Li, S., and Zhang, C. Offline reinforcement learning with reverse model-based imagination. In *Advances in Neural Information Processing Systems 34 (NeurIPS'21)*, virtual event, 2021.

Wang, X., Wongkamjan, W., and Huang, F. Live in the moment: Learning dynamics model adapted to evolving policy. *CoRR*, abs/2207.12141, 2022.

Wen, J., Dai, B., Li, L., and Schuurmans, D. Batch stationary distribution estimation. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*, virtual event, 2020.

Xu, T., Li, Z., and Yu, Y. Error bounds of imitating policies and environments. In *Advances in Neural Information Processing Systems 33 (NeurIPS'20)*, virtual event, 2020.

Xu, T., Li, Z., and Yu, Y. Error bounds of imitating policies and environments for reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):6968–6980, 2021.

Yang, M., Nachum, O., Dai, B., Li, L., and Schuurmans, D. Off-policy evaluation via the regularized lagrangian. In *Advances in Neural Information Processing Systems 33 (NeurIPS'20)*, virtual event, 2020.

Yang, S., Zhang, S., Feng, Y., and Zhou, M. A unified framework for alternating offline model training and policy learning. In *Advances in Neural Information Processing Systems 35 (NeurIPS'22)*, New Orleans, LA, 2022.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. MOPO: model-based offline policy optimization. In *Advances in Neural Information Processing Systems 33 (NeurIPS'20)*, virtual event, 2020.

Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., and Finn, C. COMBO: conservative offline model-based policy optimization. In *Advances in Neural Information Processing Systems 34 (NeurIPS'21)*, virtual event, 2021.

Appendix

A. Proofs

A.1. Value Gap Bounds

First, we prove the bound 2 of value gap:

$$|V^{\pi}_{T^*} - V^{\pi}_{\hat{T}}| \leq \frac{2R_{\max}\gamma}{(1-\gamma)^2}\mathbb{E}_{s,a\sim\rho^{\pi}} D_{\mathrm{TV}}(T^*(\cdot|s,a), \hat{T}(\cdot|s,a)). \tag{10}$$

Proof. In the proofs in this subsection, we use $\rho^{\pi}_{\hat{T}}$ to denote the occupancy measure under the transition model $\hat{T}$, and still use ρ^{π} to denote the occupancy measure in the real dynamics.

$$\begin{aligned}
|V^{\pi}_{T^*} - V^{\pi}_{\hat{T}}| &= \frac{1}{1-\gamma}|\sum_{s,a}(\rho^{\pi}(s,a) - \rho^{\pi}_{\hat{T}}(s,a))r(s,a)| && (11)\\
&\overset{(a)}{\leq} \frac{2R_{\max}}{1-\gamma}D_{\mathrm{TV}}(\rho^{\pi}, \rho^{\pi}_{\hat{T}}) && (12)\\
&\overset{(b)}{\leq} \frac{2R_{\max}\gamma}{(1-\gamma)^2}\mathbb{E}_{s,a\sim\rho^{\pi}} D_{\mathrm{TV}}(T^*(\cdot|s,a), \hat{T}(\cdot|s,a)), && (13)
\end{aligned}$$

where (a) holds because

$$\begin{aligned}
|\sum_{s,a}(\rho^{\pi}(s,a) - \rho^{\pi}_{\hat{T}}(s,a))r(s,a)| &\leq \sum_{s,a}|\rho^{\pi}(s,a) - \rho^{\pi}_{\hat{T}}(s,a)||r(s,a)| && (14)\\
&\leq R_{\max}\sum_{s,a}|\rho^{\pi}(s,a) - \rho^{\pi}_{\hat{T}}(s,a)| && (15)\\
&= 2R_{\max}D_{\mathrm{TV}}(\rho^{\pi}, \rho^{\pi}_{\hat{T}}), && (16)
\end{aligned}$$

and (b) holds because

$$D_{\mathrm{TV}}(\rho^{\pi}, \rho^{\pi}_{\hat{T}}) \leq \frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim\rho^{\pi}} D_{\mathrm{TV}}(T^*(\cdot|s,a), \hat{T}(\cdot|s,a)), \tag{17}$$

which follows the proof of Lemma 11 in (Xu et al., 2021). □

For comparison, we apply the traditional bound (Janner et al., 2019; Xu et al., 2020) to our mixture training data distribution setting:

$$\begin{aligned}
|V^{\pi}_{T^*} - V^{\pi}_{\hat{T}}| \leq & \frac{2R_{\max}\gamma}{(1-\gamma)^2}\mathbb{E}_{s,a\sim\rho^{\mathrm{mix}}} D_{TV}(T^*(\cdot|s,a), \hat{T}(\cdot|s,a)) && (18)\\
&+\frac{4R_{\max}}{(1-\gamma)^2}\sum_{i=1}^{n} w_i \max_s D_{TV}(\pi(\cdot|s), \mu_i(\cdot|s)), && (19)
\end{aligned}$$

which consists of the model error under the training data distribution and additional unavoidable policy divergence terms. *This result only suggests minimizing the training model error but ignores the generalization ability to the target policy.*

Proof.

$$\begin{aligned}
|V^{\pi}_{T^*} - V^{\pi}_{\hat{T}}| &\le |V^{\pi}_{T^*} - \sum_i w_i V^{\mu_i}_{T^*}| + |\sum_i w_i V^{\mu_i}_{T^*} - \sum_i w_i V^{\mu_i}_{\hat{T}}| + |\sum_i w_i V^{\mu_i}_{\hat{T}} - V^{\pi}_{\hat{T}}| \\
&\le \sum_i w_i (|V^{\pi}_{T^*} - V^{\mu_i}_{T^*}| + |V^{\pi}_{\hat{T}} - V^{\mu_i}_{\hat{T}}|) + |\sum_{s,a} (\rho^{\text{mix}}(s,a) - \rho^{\text{mix}}_{\hat{T}}(s,a)) r(s,a)| \\
&\overset{(*)}{\le} \frac{2R_{\max}\gamma}{(1-\gamma)^2} \mathbb{E}_{s,a\sim\rho^{\text{mix}}} D_{TV}(T^*(\cdot|s,a), \hat{T}(\cdot|s,a)) \\
&+ \frac{4R_{\max}}{(1-\gamma)^2} \sum_i w_i \max_s D_{TV}(\pi(\cdot|s), \mu_i(\cdot|s)),
\end{aligned}$$

where $(*)$ holds because

$$|V^{\pi}_{T^*} - V^{\mu_i}_{T^*}| \le \frac{2R_{\max}}{(1-\gamma)^2} \max_s D_{TV}(\pi(\cdot|s), \mu_i(\cdot|s)), \tag{20}$$

$$|V^{\pi}_{\hat{T}} - V^{\mu_i}_{\hat{T}}| \le \frac{2R_{\max}}{(1-\gamma)^2} \max_s D_{TV}(\pi(\cdot|s), \mu_i(\cdot|s)), \tag{21}$$

and

$$\begin{aligned}
|\sum_{s,a} (\rho^{\text{mix}}(s,a) - \rho^{\text{mix}}_{\hat{T}}(s,a)) r(s,a)| &\le \frac{2R_{\max}}{1-\gamma} D_{\text{TV}}(\rho^{\text{mix}}, \rho^{\text{mix}}_{\hat{T}}) && (22) \\
&\le \frac{2R_{\max}}{1-\gamma} \sum_i w_i D_{\text{TV}}(\rho^{\mu_i}, \rho^{\mu_i}_{\hat{T}}) && (23) \\
&\le \frac{2R_{\max}\gamma}{(1-\gamma)^2} \sum_i w_i \mathbb{E}_{s,a\sim\rho^{\mu_i}} D_{\text{TV}}(T^*(\cdot|s,a), \hat{T}(\cdot|s,a)) && (24) \\
&\le \frac{2R_{\max}\gamma}{(1-\gamma)^2} \mathbb{E}_{s,a\sim\rho^{\text{mix}}} D_{\text{TV}}(T^*(\cdot|s,a), \hat{T}(\cdot|s,a)). && (25)
\end{aligned}$$

□

A.2. Proofs in Section 4

Assumption A.1. For the learned model T, the point-wise model error $D_{\text{TV}}(T^*(\cdot|s,a), T(\cdot|s,a))$ is L-Lipschitz with respect to the state-action pairs, i.e.,

$$\Big| D_{\text{TV}}(T^*, T)(s_1, a_1) - D_{\text{TV}}(T^*, T)(s_2, a_2) \Big| \le L \cdot D\big((s_1, a_1), (s_2, a_2)\big), \tag{26}$$

where $D(\cdot, \cdot)$ is some kind of distance defined on the state-action space $\mathcal{S} \times \mathcal{A}$.

Proposition A.2. *Under Assump. A.1, for any policy $\pi \in \Pi$, the model error of PAM $T_{\hat{\psi}}$ can be bounded:*

$$l(\pi, T^*, T_{\hat{\psi}}) \le \min_{\mu_i \in \Omega} \big\{ l(\mu_i, T^*, T_{\hat{\psi}}) + L \cdot W_1(\rho^{\pi}, \rho^{\mu_i}) \big\}, \tag{27}$$

where $W_1(\cdot, \cdot)$ is the Wasserstein-1 distance defined on the state-action distribution space $\mathcal{P}(\mathcal{S} \times \mathcal{A})$ with the underlying metric D.

Proof. The Wasserstein-1 metric has a dual representation:

$$W_1(p, q) = \sup_{\|f\|_{Lip} \le 1} \int f(x) \mathrm{d}p(x) - f(y) \mathrm{d}q(y), \tag{28}$$

where $\|f\|_{Lip}$ is the Lipschitz constant of function f. Therefore we have

$$\begin{aligned}
|l(\pi, T^*, T_{\hat{\psi}}) - l(\mu_i, T^*, T_{\hat{\psi}})| &= |\mathbb{E}_{s,a\sim\rho^{\pi}} D_{\text{TV}}(T^*, T_{\hat{\psi}})(s,a) - \mathbb{E}_{s,a\sim\rho^{\mu_i}} D_{\text{TV}}(T^*, T_{\hat{\psi}})(s,a)| \\
&\le \sup_{\|f\|_{Lip} \le L} \int f(s,a) \mathrm{d}\rho^{\pi}(s,a) - f(s,a) \mathrm{d}\rho^{\mu_i}(s,a) && (29) \\
&= L \cdot W_1(\rho^{\pi}, \rho^{\mu_i}). && (30)
\end{aligned}$$

This holds for all $\mu_i \in \Omega$, therefore

$$l(\pi, T^*, T_{\hat{\psi}}) \leq \min_{\mu_i \in \Omega} \{ l(\mu_i, T^*, T_{\hat{\psi}}) + L \cdot W_1(\rho^\pi, \rho^{\mu_i}) \}. \tag{31}$$

□

Definition A.3. For a data-collection policy μ_i and a target policy π, we can define the adaptation gain of a learned policy-conditioned model:

$$C(\pi, \mu_i) := l(\pi, T^*, T_{\hat{F}(\mu_i)}) - l(\pi, T^*, T_{\hat{F}(\pi)}). \tag{32}$$

Proposition A.4. *Under Assump. A.1, for any policy $\pi \in \Pi$, the model error of PCM $T_{\hat{F}(\pi)}$ is bounded:*

$$l(\pi, T^*, T_{\hat{F}(\pi)}) \leq \min_{\mu_i \in \Omega} \{ \underbrace{l(\mu_i, T^*, T_{\hat{F}(\mu_i)})}_{\text{training error}} + \underbrace{L \cdot W_1(\rho^\pi, \rho^{\mu_i}) - C(\pi, \mu_i)}_{\text{generalization error}} \}. \tag{33}$$

Proof. Similar to the proof in Prop. A.2, for some behavior policy μ_i, the inaccuracy of μ_i-adapted model on the test distribution $\rho^\pi_{T^*}$ can be bounded by that on the training distribution $\rho^\mu_{T^*}$ and the generalization error caused by the distribution extrapolation

$$l(\pi, T^*, T_{\hat{F}(\mu_i)}) \leq l(\mu_i, T^*, T_{\hat{F}(\mu_i)}) + L \cdot W_1(\rho^\pi, \rho^{\mu_i}). \tag{34}$$

The π-adapted model $T_{\hat{F}(\pi)}$ enjoys an adaptation gain compared to $T_{\hat{F}(\mu_i)}$, compensating the extrapolation effect

$$l(\pi, T^*, T_{\hat{F}(\pi)}) = l(\pi, T^*, T_{\hat{F}(\mu_i)}) - C(\pi, \mu_i). \tag{35}$$

Therefore the generalization error of the π-adapted model $T_{\hat{F}(\pi)}$ on the test distribution ρ^π is reduced

$$l(\pi, T^*, T_{\hat{F}(\pi)}) \leq l(\mu_i, T^*, T_{\hat{F}(\mu_i)}) + L \cdot W_1(\rho^\pi, \rho^{\mu_i}) - C(\pi, \mu_i). \tag{36}$$

This holds for all $\mu_i \in \Omega$, therefore

$$l(\pi, T^*, T_{\hat{F}(\pi)}) \leq \min_{\mu_i \in \Omega} \{ l(\mu_i, T^*, T_{\hat{F}(\pi)}) + L \cdot W_1(\rho^\pi, \rho^{\mu_i}) - C(\pi, \mu_i) \}. \tag{37}$$

□

A.3. Analysis on the intermediate adaptation

It is hard in general to rigorously analyze the adaptation gain $C(\pi, \mu_i)$ because of the complexity of neural networks and the optimization process. To provide more concrete intuitions, in the following, we show some special cases for PCM to explain the benefit of such adaptation effects.

Case 1: Direct match. If the $T_{\hat{F}(\pi)}$ simply equals to $T_{\hat{F}(\mu_i)}$ for some training policies μ_i, e.g., equals to the one has the smallest occupancy divergence with respect to the target π, then the adaptation gain is exactly zero, which means no adaptation effect is enabled. Hence in this case the advantage compared to PAM solely comes from the reduced error in training policy distributions.

Case 2: Imaginary retraining. Another example corresponds to the (probably unachievable) perfect generalization case, where $\hat{F}(\pi) = \arg\min_{\psi \in \Psi} l(\pi, T^*, T_\psi)$. In this case, we can imaginary ρ^π based on π and search the optimal model parameter directly. Therefore the adaptation gain completely cancels out the extrapolation error and the optimal model accuracy under the capacity limitation. This case can be regarded as a ceiling of model accuracy for any practical PCM algorithms.

Case 3: Intermediate adaptation. In general, we argue that the generalization ability of a well-trained PCM will fall between the two extreme cases, where the adaptation gain is greater than zero and able to partially reduce the extrapolation error within a certain region as π diverges from Ω gradually. When π is far enough from Ω, C will reach the maximum and then may start to decrease and will be less than zero finally.

We argue that the generalization ability of PCM will more resemble Case 3, in which C will first increase and gradually decrease after reaching the maximum. We provide some empirical evidence in Sec. 6.1.2, which can support our intuition.

We provide a formulation of the intermediate adaptation effect in the following.

Assumption A.5. Assume that the adaptation gain $C(\pi, \mu_i)$ of a well-trained policy-conditioned model satisfies the following properties:

1. if $\mu_i = \pi$, the adaptation gain $C(\pi, \mu_i)$ equals zero, since the adaptation effect is not activated;
2. as π diverges from μ_i gradually, the adaptation effect becomes significant, an therefore $C(\pi, \mu_i)$ increases from zero;
3. when π is far enough from μ_i, $C(\pi, \mu_i)$ reaches the maximum and then may start to decrease due to the finite samples and bounded generalization, so it leaves the effective adaptation region.

Under Assump. A.5, there exists an $L_i > 0$ such that $C(\pi, \mu_i) \geq L_i \cdot W_1(\rho^{\pi}, \rho^{\mu_i})$ for a data-collection policy μ_i and a target policy π within the effective adaptation region, which can be justified by the empirical result in Fig. 3(a). Then we have

Proposition A.6. *Under the assumption of the adaptation gain, the generalization error of the learned policy-conditioned model can be bounded:*

$$l(\pi, T^*, T_{\hat{F}(\pi)}) \leq \min_{i \in I_\pi} \left\{ l(\mu_i, T^*, T_{\hat{F}(\mu_i)}) + (L - L_i) W_1(\rho^{\pi}, \rho^{\mu_i}) \right\}, \tag{38}$$

where I_π is the index set of the training policies in whose effective adaptation region the target policy π lies.

Proof. Substituting the assumption on the adaptation gain into the consequence of Prop. A.4 yields

$$\begin{aligned} l(\pi, T^*, T_{\hat{F}(\pi)}) &\leq \min_{\mu_i \in \Omega} \left\{ l(\mu_i, T^*, T_{\hat{F}(\mu_i)}) + L \cdot W_1(\rho^{\pi}, \rho^{\mu_i}) - C(\pi, \mu_i) \right\} & (39) \\ &\leq \min_{i \in I_\pi} \left\{ l(\mu_i, T^*, T_{\hat{F}(\mu_i)}) + (L - L_i) W_1(\rho^{\pi}, \rho^{\mu_i}) \right\}, & (40) \end{aligned}$$

where we replace the whole data-collection policy set Ω with a local data-collection policy index set I_π because the adaptation gain is locally assumed. $\square$

Prop. A.6 shows that the coefficient of extrapolation term is reduced from L to $L - L_i$ thanks to the adaptation effect, implying better generalization to unseen policies within a reasonable effective region. Note here we do not argue that policy-conditioned models can generalize better to any target policy since here we only make local assumptions on the adaptation effect, and for those policies quite distinct from all the data-collection policies we do not expect a high fidelity no matter for policy-agnostic or policy-conditioned models.

B. Implementation Details

B.1. Pseudocode

The pseudocode of PCM via policy representation is listed in Alg. 1

Algorithm 1 Policy-conditioned Model Learning

```
Require: Offline dataset $\mathcal{D} = \{\tau^{(j)}\}_{j=1}^{N}$ with $N$ trajectories, policy encoder $q_\phi$, policy decoder $p_\theta$, policy-conditioned dynamics model $T_\psi$.
for $i = 1$ to $n_{\text{iter}}$ do
    sample a trajectory $\tau \sim \mathcal{D}$
    sample a timestep $t \sim [0, \text{len}(\tau)]$
    use the trajectory $\tau_{0:t+1}$ to optimize $\phi, \theta, \psi$ according to Eq. (6).
end for
```

B.2. Implementation Details of Policy-Agnostic Model (PAM)

PAM is a 4-layer feedforward neural network with 200 hidden units. In addition, we borrow the design of blocks in Transformer. We employ a residual connection around each of the two layers, followed by layer normalization. That is, the output of each layer is LayerNorm$(x + \text{Layer}(x))$, where Layer(x) $=$ Dropout(Activation(Linear(x))).

B.3. Implementation Details of Policy-Conditioned Model (PCM)

PCM follows the same architecture as PAM except for an additional policy encoder and a policy decoder. The policy encoder is a 3-layer GRU with 128 hidden units and outputs a 128-dim embedding. Then the outputted embedding will be concatenated with the first layer's output of the dynamics model. The policy decoder takes a state and an embedding as input and outputs an action distribution. It is also constructed by a 4-layer MLP with 200 hidden units.

We present hyperparameters for model training in Tab. 3, which are shared by PAM and PCM (except λ since it is a hyperparameter unique to PCM).

Table 3: Training hyperparameters of PAM and PCM.

Hyperparameters	**Value**	**Description**
Batch size	32	Batch size for gradient descent.
Optimizer	Adam	Optimizer.
Learning rate	1e-4	Learning rate for gradient descent.
Dropout rate	0.1	Dropout rate.
λ	0.01	Weight of policy representation loss.

C. Details of OPE

C.1. Off-policy Evaluation with PCM

Since we argue that our proposed PCM tends to have a smaller value gap, an obvious application that can make full use of its superiority is off-policy evaluation (OPE). OPE via a learned dynamics model is straightforward, which only needs to compute the return using simulated trajectories generated by the evaluated policy under the learned dynamics model. Due to the stochasticity in the model and the policy, we estimate the return for a policy with Monte-Carlo sampling. See Alg. 2 for pseudocode.

In practical evaluation, we choose $\gamma = 0.995$ and $N = 10$.

C.2. Metrics

The metrics we use in our paper are defined as follows:

Algorithm 2 Off-policy Evaluation with PCM

```
**Require:** Policy-conditioned dynamics model $(q_\phi, p_\theta, T_\psi)$ learned on $\mathcal{D}$, evaluated policy $\pi$, number of rollouts $N$. set of initial states $\mathcal{S}_0$, discount factor $\gamma$, horizon length $H$.
for $i = 1$ to $N$ do
    $R_i = 0$
    Sample initial state $s_0 \sim \mathcal{S}_0$
    Initialize $\tau_{-1} = \mathbf{0}$
    for $t = 0$ to $H - 1$ do
        $z_t = q_\phi(\tau_{t-1})$
        $a_t \sim \pi(\cdot|s_t)$
        $s_{t+1}, r_t \sim T_\psi(\cdot|s_t, a_t, z_t)$
        $R_i = R_i + \gamma^t r_t$
        $\tau_t = (\tau_{t-1}, s_t, a_t)$
    end for
end for
return $\frac{1}{N}\sum_{i=1}^{N} R_i$
```

Absolute Error The absolute error is defined as the difference between the value and estimated value of a policy:

$$\text{AbsErr} = |V^\pi - \hat{V}^\pi|, \tag{41}$$

where V^π is the true value of the policy and $\hat{V}^\pi$ is the estimated value of the policy.

Rank correlation Rank correlation measures the correlation between the ordinal rankings of the value estimates and the true values, which can be written as:

$$\text{RankCorr} = \frac{\text{Cov}(V^\pi_{1:N}, \hat{V}^\pi_{1:N})}{\sigma(V^\pi_{1:N})\sigma(\hat{V}^\pi_{1:N})}, \tag{42}$$

where $1:N$ denotes the indices of the evaluated policies.

Regret@k Regret@k is the difference between the value of the best policy in the entire set, and the value of the best policy in the top-k set (where the top-k set is chosen by estimated values). It can be defined as:

$$\text{Regret @k} = \max_{i \in 1:N} V^\pi_i - \max_{j \in \text{topk}(1:N)} V^\pi_j, \tag{43}$$

where topk$(1:N)$ denotes the indices of the top K policies as measured by estimated values $\hat{V}^\pi$.

C.3. Detailed Results

Detailed results tables are presented here (averaged over 3 random seeds).

Table 4: Raw absolute error for each algorithm on D4RL and RL Unplugged tasks.

Task Name	**FQE**	**DR**	**IS**	**DICE**	**VPM**	**PAM**	**PCM (Ours)**
halfcheetah-medium-replay	1003±132	1001±129	1409±154	1440±158	1384±148	1009±76	**622±160**
hopper-medium-replay	234±71	267±60	375±54	364±49	392±44	185±22	**44±11**
walker2d-medium-replay	313±73	296±54	427±60	347±51	424±64	358±85	**140±28**
ant-medium-replay	**410±79**	421±72	603±101	583±110	612±105	558±11	529±23
halfcheetah-medium-expert	1014±101	1015±103	1400±146	1078±132	1427±111	1184±421	**935±41**
hopper-medium-expert	282±76	426±99	106±29	259±54	442±43	313±130	**114±5**
walker2d-medium-expert	233±42	217±46	436±62	322±60	425±611	446±59	**95±34**
ant-medium-expert	**319±67**	326±66	604±102	471±100	604±106	524±11	564±81
cartpole-swingup	19±1	24±3	69±2	23±2	38±4	22.4±7.8	**10±1**
cheetah-run	48±2	40±2	44±2	23±11	62±4	14±4	**8±1**
fish-swim	20±2	20±2	35±2	59±2	31±1	13±0	**11±0**

Table 5: Normalized absolute error for each algorithm on D4RL and RL Unplugged tasks.

Task Name	**FQE**	**DR**	**IS**	**DICE**	**VPM**	**PAM**	**PCM (Ours)**
halfcheetah-medium-replay	0.50±0.06	0.50±0.06	0.75±0.08	0.72±0.08	0.69±0.07	0.53±0.04	**0.31±0.08**
hopper-medium-replay	0.43±0.13	0.49±0.11	0.69±0.10	0.67±0.07	0.72±0.08	0.34±0.04	**0.08±0.02**
walker2d-medium-replay	0.56±0.13	0.53±0.10	0.76±0.11	0.67±0.09	0.76±0.11	0.66±0.16	**0.25±0.05**
ant-medium-replay	**0.36±0.07**	0.37±0.06	0.53±0.09	0.51±0.10	0.54±0.09	0.49±0.01	0.46±0.02
halfcheetah-medium-expert	0.51±0.05	0.51±0.05	0.70±0.07	0.54±0.07	0.71±0.06	0.59±0.21	**0.46±0.02**
hopper-medium-expert	0.43±0.12	0.65±0.15	0.16±0.04	0.39±0.08	0.67±0.07	0.57±0.24	**0.21±0.01**
walker2d-medium-expert	0.42±0.08	0.39±0.08	0.78±0.11	0.58±0.11	0.76±0.11	0.82±0.11	**0.17±0.06**
ant-medium-expert	**0.28±0.06**	0.29±0.06	0.53±0.09	0.41±0.09	0.53±0.09	0.46±0.01	0.49±0.07
cartpole-swingup	0.17±0.01	0.22±0.02	0.57±0.02	0.19±0.01	0.31±0.03	0.20±0.07	**0.09±0.01**
cheetah-run	0.68±0.03	0.57±0.03	0.63±0.03	0.33±0.05	0.88±0.06	0.22±0.07	**0.15±0.01**
fish-swim	0.44±0.03	0.45±0.04	0.77±0.04	1.32±0.05	0.69±0.02	0.28±0.01	**0.25±0.01**

Table 6: Rank correlation for each algorithm on D4RL and RL Unplugged tasks.

Task Name	**FQE**	**DR**	**IS**	**DICE**	**VPM**	**PAM**	**PCM (Ours)**
halfcheetah-medium-replay	0.26±0.37	0.32±0.37	0.59±0.26	-0.15±0.41	-0.07±0.36	0.71±0.13	**0.86±0.06**
hopper-medium-replay	0.17±0.15	0.24±0.25	0.41±0.32	0.21±0.34	-0.11±0.22	0.91±0.03	**0.94±0.02**
walker2d-medium-replay	-0.19±0.36	-0.37±0.39	0.65±0.24	0.55±0.23	-0.52±0.25	0.14±0.42	**0.71±0.16**
ant-medium-replay	**0.57±0.28**	0.45±0.32	0.07±0.39	-0.24±0.39	-0.26±0.29	-0.05±0.12	0.06±0.11
halfcheetah-medium-expert	0.62±0.27	0.62±0.27	-0.06±0.37	-0.08±0.35	-0.47±0.29	0.44±0.46	**0.84±0.02**
hopper-medium-expert	-0.33±0.30	-0.41±0.27	0.37±0.27	-0.08±0.32	0.21±0.32	0.54±0.32	**0.74±0.03**
walker2d-medium-expert	0.25±0.32	0.19±0.33	0.24±0.33	-0.34±0.34	0.49±0.37	0.21±0.13	**0.88±0.13**
ant-medium-expert	**0.37±0.35**	0.35±0.35	-0.21±0.35	-0.33±0.40	-0.28±0.28	-0.35±0.49	0.02±0.12
cartpole-swingup	0.70±0.07	0.55±0.09	-0.23±0.11	-0.16±0.11	0.01±0.11	0.63±0.13	**0.90±0.01**
cheetah-run	0.56±0.08	0.56±0.08	-0.01±0.12	0.07±0.11	0.01±0.12	0.64±0.08	**0.74±0.03**
fish-swim	0.10±0.12	0.11±0.12	-0.17±0.11	0.44±0.09	**0.56±0.08**	0.20±0.13	0.45±0.03

Table 7: Regret for each algorithm on D4RL (Regret@1) and RL Unplugged (Regret@5) tasks.

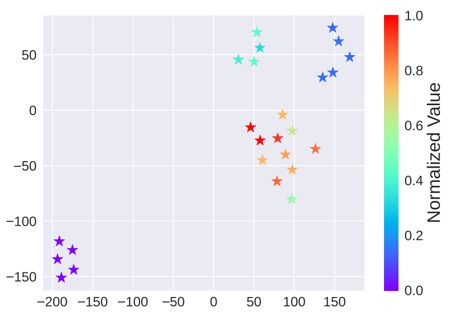
Task Name	**FQE**	**DR**	**IS**	**DICE**	**VPM**	**PAM**	**PCM (Ours)**
halfcheetah-medium-replay	0.36±0.16	0.33±0.18	**0.13±0.10**	0.30±0.07	0.25±0.09	0.23±0.12	0.15±0.04
hopper-medium-replay	0.31±0.18	0.33±0.20	0.11±0.06	0.26±0.10	0.33±0.23	0.16±0.12	**0.08±0.04**
walker2d-medium-replay	0.24±0.20	0.68±0.23	**0.02±0.05**	0.18±0.12	0.46±0.31	0.21±0.12	0.05±0.01
ant-medium-replay	**0.05±0.19**	0.17±0.31	0.18±0.06	0.09±0.10	0.03±0.08	0.21±0.10	0.08±0.03
halfcheetah-medium-expert	0.14±0.07	0.14±0.07	0.73±0.42	0.38±0.37	0.80±0.34	0.36±0.45	**0.12±0.05**
hopper-medium-expert	0.41±0.20	0.34±0.35	0.06±0.03	0.20±0.08	0.13±0.10	0.14±0.14	**0.08±0.04**
walker2d-medium-expert	0.22±0.14	0.30±0.12	0.13±0.07	0.78±0.27	0.24±0.42	0.43±0.46	**0.10±0.04**
ant-medium-expert	0.36±0.14	0.37±0.13	0.46±0.18	0.60±0.16	0.32±0.24	0.59±0.28	**0.06±0.04**
cartpole-swingup	0.06±0.04	0.28±0.05	0.73±0.16	0.68±0.41	0.50±0.13	0.04±0.06	**0.00±0.00**
cheetah-run	0.17±0.05	0.09±0.05	0.40±0.21	0.27±0.05	0.37±0.04	0.24±0.18	**0.00±0.00**
fish-swim	0.50±0.03	0.61±0.12	0.12±0.05	0.35±0.24	**0.02±0.02**	0.18±0.14	0.11±0.06

Table 8: Ablation of the model capacity.

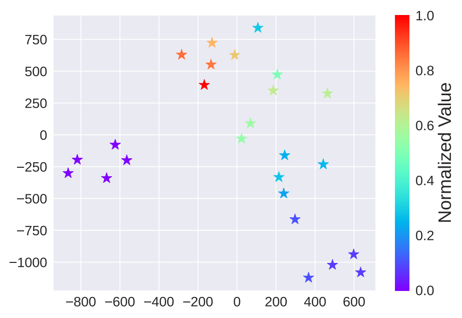
Task Name	**absolute error**	**rank correlation**	**regret**
PAM	0.51 ± 0.04	0.47 ± 0.10	0.22 ± 0.07
PAM (larger)	0.26 ± 0.05	0.61 ± 0.07	0.12 ± 0.08
PCM	0.19 ± 0.03	0.77 ± 0.05	0.07 ± 0.03

D. Visualizations for Policy Representations

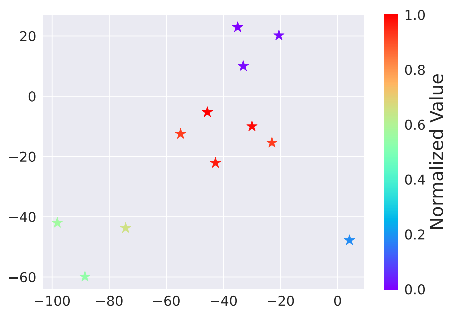
We provide the visualization of policy representations in more tasks as shown in Fig. 7. The results shows that PCM can effectively identify similar policies and distinguish different policies.



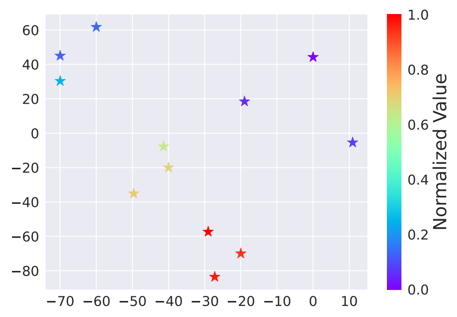
(a) Hopper



(b) Walker2d



(c) Carpole-swingup



(d) Fish-swim

Figure 7: Illustrations of the t-SNE visualization for policy representations of different policies learned by PCM in Hopper, Walker2d, Carpole-swingup, and Fish-swim. For each task, several policies (denoted by different markers) are plotted, which are colored according to the normalized performance.

Table 9: Hyperparameters of MPC.

Hyperparameters	Value	Description
Planning horizon	30	Length of the planning horizon.
Num candidates	500	Number of the sampled sequence actions.
Num elites	50	Number of elites.
α	0.1	How much of the previous mean is used for the next CEM iteration.
CEM iteration	25	Number of the CEM iterations.

E. Experiment Details of the Empirical Evidence Versification

We now introduce some experimental details in Sec 6.1.2.

E.1. Details of Data Collection

Data collection for evaluating model error and adaptation gain. We train SAC (Haarnoja et al., 2018) for 1000 epochs (each epoch contains 1k gradient steps) in HalfCheetah environment. Then we record policy snapshots at 10, 20,..., 100 epochs and use each policy to sample 20 trajectories for data collection. When evaluating adaptation gain, we fix a data-collection policy μ_i and select various policies during SAC training as target policies.

Data collection for evaluating value gap. We construct datasets with varying levels of diversity (20%, 50%, 80%), where the percentage indicates that the dataset is created from the replay buffer of SAC (Haarnoja et al., 2018) until the policy reaches the specific level of performance. We train PAM and PCM on each dataset and test them on the other 11 policies provided by the DOPE benchmark (Fu et al., 2021a), which were unseen before in the datasets.

E.2. Details of Adaptation Gain

Recall that the adaptation gain is defined as

$$C(\pi, \mu_i) := l(\pi, T^*, T_{\hat{F}(\mu_i)}) - l(\pi, T^*, T_{\hat{F}(\pi)}),$$

where l is the TV divergence between true and learned dynamics. However, directly computing C is intractable since the true transition T^* is unknown. Therefore, we instead use the mean squared error $\frac{1}{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{S}|}\left(\hat{s}^{(i)} - s^{(i)}\right)^2$ to compute inconsistency between true and learned dynamics, where $\hat{s} \sim T_\psi$ and $s \sim T^*$.

F. Details of model predictive control

Once a model is learned, we can use it for control by predicting the future outcomes of candidate policies or actions and then selecting the particular candidate that is predicted to result in the highest reward. A classical planning method is model predictive control (MPC) (Camacho & Alba, 2013) which plans for a sequence of actions. We choose MPC here for several reasons, including implementation simplicity and lower computational burden.

Given the state of the system s_t at time t, the planning horizon H, and an action sequence $\boldsymbol{a}_{t:t+H} = \{a_t, ..., a_{t+H}\}$, the dynamics model T_ψ will predict a state trajectory $\boldsymbol{s}_{t:t+H}$. At each time step t, the MPC controller applies the first action a_t of the sequence of the optimized actions $\arg\max_{\boldsymbol{a}_{t:t+H}} \sum_{i=t}^{H} r(s_i, a_i)$. A common way to generate candidate action sequences is random shooting, which simply generates N independent random action sequences. This approach has been shown to achieve success on many control tasks, but it has significant drawbacks: it scales poorly with the dimension of both the action space and the planning horizon. As suggested in Chua et al. (2018), we use CEM instead of random shooting, which samples actions from a distribution closer to previous action samples that yielded high rewards.

All the experiments in Sec. 6.2.3 share the same hyperparameters as shown in Tab. 9.

G. Details of Offline Policy Selection (OPS)

We train MOPO (Yu et al., 2020) for 1000 epochs and record policy snapshots at the latest 20 epochs for OPS. We report these policies' performance in Tab. 10, used for our OPS tasks, and compare them with the maximum performance within the dataset. The results show that MOPO usually gives different policies even near the end of training, suitable for OPS tasks and comparing different methods. Besides, for the halfcheetah and hopper task, the policies learned by MOPO have better performance than the behavior policies within the dataset, indeed showing a generalization beyond the dataset.

We compare our method against FQE, IS, DICE, PAM as well as directly selecting the last-epoch policy. The raw performance of policies selected by each OPS approach on each task is listed in Tab. 11. Tab. 1 shows the performance gains by different methods. The performance gain is computed by $\frac{(V_{\text{selected}}-\bar{V})}{V_{\max}-\bar{V}} \times 100\%$, where V_{selected} represents the value of the selected policy and $\bar{V}, V_{\max}$ are the average and max values of the evaluated policies, respectively. It is noteworthy that the gains of FQE and PAM are even lower than directly selecting the last-epoch policy, which is also indicated in another work (Qin et al., 2022). In contrast, our approach shows a brilliant performance, implying that it can reliably choose a better policy for an offline RL algorithm to deploy.

Table 10: Statistics of the policy performance used for OPS tasks.

Task	Min	Max	Mean	Max within dataset
halfcheetah-medium-replay	59.4	75.7	70.6 ± 4.3	42.4
hopper-medium-replay	42.7	106.0	96.0 ± 14.8	98.6
walker2d-medium-replay	18.5	88.4	79.5 ± 14.8	89.9

Table 11: Raw performance of offline policy selection for MOPO (Yu et al., 2020) via different approaches.

Task Name	Last Epoch	FQE	IS	DICE	PAM	PCM (Ours)
halfcheetah-medium-replay	72.3	70.0	74.8	71.3	71.3	75.9
hopper-medium-replay	102.0	100.0	102.0	98.7	101.2	102.8
walker2d-medium-replay	79.3	80.3	83.5	81.2	76.5	85.4
Average	84.5	83.4	86.8	83.7	83.0	88.0

Following an anonymous reviewer's suggestion, we also conducted the OPS experiments on more tasks and tried more repetitions. The results of PAM and PCM averaged over three seeds are reported in Tab 12.

Table 12: Performance of offline policy selection for MOPO (Yu et al., 2020) of PAM and PCM methods on more tasks.

Task Name	PAM	PCM
halfcheetah-medium-replay	-2.1%±13.3%	78.9%±14.1%
hopper-medium-replay	43.1%±3.1%	54.7%±10.5%
walker2d-medium-replay	-4.5%±23.4%	82.8%±21.9%
halfcheetah-medium-expert	35.7%±13.8%	50.3%±21.3%
hopper-medium-expert	-35.3%±46.8%	56.7%±13.3%
walker2d-medium-expert	1.6%±43.9%	20.4%±38.7%
Average	6.42%±24.05%	57.3%±19.97%

Appendix References

Fu, J., Norouzi, M., Nachum, O., Tucker, G., Wang, Z., Novikov, A., Yang, M., Zhang, M. R., Chen, Y., Kumar, A., Paduraru, C., Levine, S., and Paine, T. Benchmarks for deep off-policy evaluation. In *9th International Conference on Learning Representations (ICLR'21)*, virtual event, 2021.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. In *Advances in neural information processing systems 32 (NeurIPS'19)*, Vancouver, BC, Canada, 2019.

Xu, T., Li, Z., and Yu, Y. Error bounds of imitating policies and environments. In *Advances in Neural Information Processing Systems 33 (NeurIPS'20)*, virtual event, 2020.

Xu, T., Li, Z., and Yu, Y. Error bounds of imitating policies and environments for reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):6968–6980, 2021.