Batch Reinforcement Learning with Hyperparameter Gradients

Byung-Jun Lee [*1] **Jongmin Lee** [*1] **Peter Vrancx** [2] **Dongho Kim** [2] **Kee-Eung Kim** [2 3]

Abstract

We consider the batch reinforcement learning problem where the agent needs to learn only from a fixed batch of data, without further interaction with the environment. In such a scenario, we want to prevent the optimized policy from deviating too much from the data collection policy since the estimation becomes highly unstable otherwise due to the off-policy nature of the problem. However, imposing this requirement too strongly will result in a policy that merely follows the data collection policy. Unlike prior work where this trade-off is controlled by hand-tuned hyperparameters, we propose a novel batch reinforcement learning approach, batch optimization of policy and hyperparameter (BOPAH), that uses a gradient-based optimization of the hyperparameter using held-out data. We show that BOPAH outperforms other batch reinforcement learning algorithms in tabular and continuous control tasks, by finding a good balance to the trade-off between adhering to the data collection policy and pursuing the possible policy improvement.

1. Introduction

In many real-world applications of reinforcement learning (RL), exploratory behavior can be very costly when the agent interacts with the environment. For example, it would be unreasonable to deploy an ϵ-greedy policy for autonomous vehicles, industrial plants, and clinical treatments, let alone many others. One of the common and straightforward practices in these scenarios is to build a simulator from collected data, train the agent with the simulated environment by allowing to make as much exploration as needed, and then deploy a fully optimized policy into the real environment. However, this approach requires a lot of human effort including domain expertise in building a faithful simulator that warrants a successful performance in the real environment. This paper concerns with such a scenario, often referred to as *batch RL*: optimize policy only from a fixed batch of data, without further interaction with the environment or a high-fidelity simulator.

Since the policy being optimized would be different from the policy used for data collection, batch RL algorithms are mostly founded on techniques in off-policy RL algorithms. They are designed to learn when the *behavior policy* (policy used to collect experience) differs from the *estimation policy* (policy we aim to learn). Recent off-policy policy optimization algorithms, e.g. (Lillicrap et al., 2016; Munos et al., 2016; Haarnoja et al., 2018), have shown to achieve remarkable sample efficiency in standard benchmark tasks. However, they still assume continuous interaction with the environment with the behavior policy being improved closely together with the estimation policy. On the other hand, the batch RL setting assumes the behavior policy being fixed throughout the policy optimization. This difference is considered a fundamental challenge for batch RL: as we optimize the estimation policy, it would differ more from the behavior policy, leading to severe *covariate shift* in the batch data. As the policy optimization frequently relies on function approximators trained on the data distributed by the behavior policy, the optimization could actually get worse due to the inevitable generalization error of the function approximators. Thus the challenge is about estimating how much we can trust the policy evaluation and safely optimize the policy, rather than about improving the sample efficiency as in standard off-policy RL scenarios, as exemplified by recent work on batch RL (Laroche et al., 2019; Fujimoto et al., 2019).

In this paper, we introduce a novel batch RL framework that uses the validation data to estimate the reliability of policy updates. Model selection and hyperparameter tuning with a held-out dataset is a standard practice in supervised learning, but has not been adapted to RL, to the best of our knowledge. The contribution of this paper is two-fold: first, we present a *generalized* KL-regularized RL framework that effectively constrains the distance of the estimation policy from the behavior policy differently per state and stabilizes the training process. Second, we present BOPAH (batch optimization of

[*] Equal contribution [1] School of Computing, KAIST, Daejeon, South Korea [2] PROWLER.io [3] Graduate School of AI, KAIST, Daejeon, South Korea. Correspondence to: Byung-Jun Lee <bjlee@ai.kaist.ac.kr>, Jongmin Lee <jmlee@ai.kaist.ac.kr>, Kee-Eung Kim <kekim@kaist.ac.kr>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

policy and hyperparameter) that optimizes the hyperparameter in the KL-regularized RL objective via the hypergradient (i.e. hyperparameter gradient) on the validation data. We present a model-based algorithm assuming tabular tasks as well as a model-free algorithm for continuous control tasks.

2. Related Works

The key concept in recent batch RL algorithms is to improve upon the baseline policy used for data collection. Petrik et al. (2016) consider the robust policy improvement over the baseline policy in the worst-case scenario. Laroche et al. (2019) presents SPIBB that bootstraps the estimation policy with the behavior policy in the state-action pairs that are not observed enough. While both algorithms are proven to be safe with their finite sample bounds, the hyperparameter values that guarantee a safe improvement with high probability can be too conservative to be used in practice.

In the case of continuous state and action spaces, Fujimoto et al. (2019) propose BCQ where the actor can only perturb the behavior policy by a limited amount. Kumar et al. (2019) provide a bound on suboptimality of an estimation policy concerning its support, and present BEAR, which imposes an MMD constraint between the estimation policy and the behavior policy. Siegel et al. (2020) use KL constraint and propose ABM that imposes advantage-weighting when estimating behavior policy from the batch data, which filters out trajectories that would lead to worse performance than the current policy. They have empirically shown to robustly improve over the baseline policy in continuous control tasks. However, the hyperparameters that control the risk play a crucial role in the performance, which must be hand-tuned in practice.

Reinforcement learning with KL regularization (Todorov, 2007; Kappen et al., 2012; Schulman et al., 2017; Fox et al., 2016; Galashov et al., 2019) or KL constraint (Schulman et al., 2015; Achiam et al., 2017; Sun et al., 2018) have been extensively studied. Theoretical analyses similar to the bounds we propose have also been presented in a model-based RL context (Sun et al., 2018; Janner et al., 2019). State-independent KL regularization has also been employed in the context of batch RL (Jaques et al., 2019; Wu et al., 2019) with the hand-tuned hyperparameter. Nevertheless, our generalization of KL-regularized RL to state-dependent regularization provides a unique insight on how they apply to batch RL.

Agarwal et al. (2020) demonstrate that recent off-policy deep RL algorithms, without correcting distribution mismatch, trained on sufficiently large and diverse offline datasets can result in high quality policies. In contrast, the critical assumption we make in this paper is that the data collection policy is determined by domain-specific requirements, rather than selected freely to mitigate problems related to batch RL. In this situation, addressing the distribution mismatch is essential.

Lastly, we will adopt the gradient-based hyperparameter optimization approach in supervised learning. When it is possible to obtain the gradient of the model selection criterion with respect to hyperparameters, gradient-based optimization of hyperparameter (Bengio, 2000; Maclaurin et al., 2015) is able to efficiently optimize a large number of hyperparameters, outperforming Bayesian optimization models (Pedregosa, 2016). We will show how hypergradient computations can be formulated in terms of policy evaluation in the RL context.

3. Preliminaries

We consider the environment modeled as a Markov decision process (MDP), $M = \langle \mathcal{S}, \mathcal{A}, P, R, d_0, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s_{t+1}|s_t, a_t)$ is the transition probability, $r_t = R(s_t, a_t) \in \mathbb{R}$ is the immediate reward function, $d_0(s) = p(s_0 = s)$ is the initial state distribution, and $\gamma \in (0, 1)$ is the discount factor. We denote $d^\pi_M(s) = (1-\gamma) \sum_{t=0}^\infty \gamma^t p(s_t = s|\pi, M)$ the discounted state marginal of policy π in the MDP M.

The state and the action value functions of policy π on MDP M are denoted by $V^\pi_M(s)$ and $Q^\pi_M(s, a)$ respectively. We adopt the discounted return objective, i.e. $V^\pi_M(s) = \mathbb{E}_{\pi, M|s}[\sum_{t=0}^\infty \gamma^t r_t]$ and $Q^\pi_M(s, a) = \mathbb{E}_{\pi, M|s,a}[\sum_{t=0}^\infty \gamma^t r_t]$. We measure the performance of a policy π on MDP M by the expectation of the value function under the initial state distribution, $\rho_M(\pi) = \mathbb{E}_{s_0 \sim d_0}[V^\pi_M(s_0)]$. The objective of policy optimization is to find the optimal policy π^* that maximizes the expected discounted return, $\pi^* = \arg\max_\pi \rho_M(\pi)$.

When the model M is available, the value of policy π can be computed by iteratively applying the Bellman backup operator $\mathcal{T}^\pi_M$:

$$\mathcal{T}^\pi_M Q(s, a) = R(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[V(s')],$$
$$\text{where } V(s) = \mathbb{E}_{a \sim \pi(\cdot|s)}[Q(s, a)],$$

which is a contraction mapping and has a unique fixed point solution Q^π_M.

In batch RL, the agent learns from the fixed dataset of experiences $\mathcal{D} = \{(s_i, a_i, s'_i, r_i)\}_{i=0}^N$ without direct interaction with the environment. We will refer to the policy μ used for the data collection as the *behavior policy* and the policy π being optimized as the *estimation policy*, borrowing the terminology in off-policy RL. If we only use samples in $\mathcal{D}$ to compute the expectation of the Bellman backup operator $\mathcal{T}^\pi_M$, the resulting approximate operator $\mathcal{T}^\pi_{\widehat{M}}$ has a unique fixed point solution $Q^\pi_{\widehat{M}}$, which is a state-action value function on a Maximum Likelihood Estimation (MLE) MDP

$\widehat{M} = \langle \mathcal{S}, \mathcal{A}, \widehat{P}, R, d_0, \gamma \rangle$, where $\widehat{P}$ is the maximum likelihood estimate of P by $\mathcal{D}$. We also use the notation $d^\pi_{\widehat{M}}$ to denote the discounted state marginal of a policy π under the MLE MDP.

4. Generalized KL-Regularization for Batch Reinforcement Learning

In a naive approach to batch RL, we could build an MDP model $\widehat{M}$ from the batch data and optimize the policy using the model. However, the resulting policy may fail to produce any improvement over the behavior policy or even perform severely worse (Petrik et al., 2016; Laroche et al., 2019). For safe RL with batch data, we will first derive a policy improvement bound for model-based RL, which naturally yields a regularization function for batch RL.

Our derivation starts with bounding the policy evaluation error incurred by the model error and the distribution shift of the policy.

Theorem 4.1. *Let*

$$\epsilon^\pi_M = \mathbb{E}_{s\sim d^\pi_M}\left[\mathbb{TV}^{\pi,\mu}_s\right],\ \epsilon^\pi_{\widehat{M}} = \mathbb{E}_{s\sim d^\pi_{\widehat{M}}}\left[\mathbb{TV}^{\pi,\mu}_s\right],$$
$$\epsilon^P_M = \mathbb{E}_{\substack{s\sim d^\mu_M \\ a\sim\mu}}\left[\mathbb{TV}^{P,\widehat{P}}_{s,a}\right]$$

where $\mathbb{TV}^{p,q}_x$ denotes the total variation distance between $p(\cdot|x)$ and $q(\cdot|x)$. Suppose the reward function is bounded $|R(s,a)| \leq R_{\max}$ for all s, a and known to the agent for simplicity. For any policies π, μ, and an estimated MLE MDP $\widehat{M}$, the difference of policy evaluation is bounded by:

$$|\rho_M(\pi) - \rho_{\widehat{M}}(\pi)| \leq c_1\left(\epsilon^\pi_M + \epsilon^\pi_{\widehat{M}}\right) + c_2\epsilon^P_M \qquad (1)$$

where $c_1 = \frac{2R_{\max}}{(1-\gamma)^2}$ and $c_2 = \frac{2\gamma R_{\max}}{(1-\gamma)^2}$.

Since the error ϵ^P_M only depends on the dataset and is not directly controllable during batch RL policy optimization, we can gather only relevant terms and formulate the following constrained optimization similar to (Schulman et al., 2015; Achiam et al., 2017):

$$\pi^*_\delta = \arg\max_\pi \rho_{\widehat{M}}(\pi) \qquad (2)$$
$$\text{s.t. } \mathbb{E}_{s\sim d^\pi_M}[\mathbb{KL}^{\pi,\mu}_s] \leq \delta, \mathbb{E}_{s\sim d^\pi_{\widehat{M}}}[\mathbb{KL}^{\pi,\mu}_s] \leq \delta,$$

where $\mathbb{KL}^{p,q}_x$ denotes the KL-divergence between $p(\cdot|x)$ and $q(\cdot|x)$. Then, we can provide a policy improvement bound for π^*_δ in Eq. (2), which can be derived from Theorem 4.1:

Corollary 4.1. *The negative baseline regret (Petrik et al., 2016) of π^*_δ, which is the performance improvement by adopting π^*_δ instead of the baseline policy μ on the true environment M, is lower bounded by:*

$$\rho_M(\pi^*_\delta) - \rho_M(\mu) \geq \rho_{\widehat{M}}(\pi^*_\delta) - \rho_{\widehat{M}}(\mu) - c_1\sqrt{2\delta} - 2c_2\epsilon^P_M$$

with c_1 and c_2 defined in Theorem 4.1.

This lower bound has two competing factors with respect to δ: increasing δ would enlarge the feasibility region and thus increase the objective $\rho_{\widehat{M}}(\pi)$, but this will also make the estimation of expected return under $\widehat{M}$ unreliable since the estimation error will increase. Most of the off-policy RL and batch RL algorithms have this trade-off captured by *hyperparameters* (Schulman et al., 2015; Petrik et al., 2016; Laroche et al., 2019).

Now, instead of using KL-divergence as constraints in the policy optimization, we reformulate the problem as an unconstrained optimization where the KL-divergence acts as a regularization:

$$\widetilde{\rho}_{\widehat{M}}(\pi) = \rho_{\widehat{M}}(\pi) - \alpha\Big(\mathbb{E}_{s\sim d^\pi_M}[\mathbb{KL}^{\pi,\mu}_s] + \mathbb{E}_{s\sim d^\pi_{\widehat{M}}}[\mathbb{KL}^{\pi,\mu}_s]\Big)$$
$$= \mathbb{E}_{\pi,\widehat{M}}\left[\sum_{t=0}^{\infty}\gamma^t\left(r_t - \alpha\left(\frac{d^\pi_M(s_t)}{d^\pi_{\widehat{M}}(s_t)} + 1\right)\mathbb{KL}^{\pi,\mu}_{s_t}\right)\right]$$

This objective is essentially a variation of KL-regularized RL (Todorov, 2007; Kappen et al., 2012; Schulman et al., 2017; Fox et al., 2016; Galashov et al., 2019), which considers the expected KL-divergence both in the true MDP and the estimated MDP. Note that the term $(d^\pi_M(s_t)/d^\pi_{\widehat{M}}(s_t)+1)$ is very hard to estimate without access to the true model M. We thus work with the objective

$$\max_\pi \mathbb{E}_{\pi,\widehat{P}}\left[\sum_{t=0}^{\infty}\gamma^t\left(r_t - \alpha\beta(s_t)\mathbb{KL}^{\pi,\mu}_{s_t}\right)\right] \qquad (3)$$

where α is the state-independent hyperparameter and $\beta(s)$'s are the state-dependent hyperparameters. We will denote $\alpha(s) = \alpha\beta(s)$ for brevity. In the later part of the paper, we will automatically tune $\alpha(s)$ using held-out validation set.

4.1. KL-Regularized Policy Iteration

In this section, we derive batch policy iteration with the *generalized* KL-regularization that alternates between policy evaluation and policy improvement, a planning algorithm for Eq. (3) with fixed yet arbitrary hyperparameters $\alpha(s)$.

We first present the iterative policy evaluation method by defining KL-regularized Bellman operator $\mathcal{T}^\pi_{\mathbb{KL}}$. For fixed policy π,

$$\mathcal{T}^\pi_{\mathbb{KL}}\widetilde{Q}(s,a) = R(s,a) + \mathbb{E}_{s'}\left[\widetilde{V}(s')\right] \qquad (4)$$
$$\text{where } \widetilde{V}(s) = -\alpha(s)\mathbb{KL}^{\pi,\mu}_s + \mathbb{E}_{a\sim\pi}\left[\widetilde{Q}(s,a)\right]$$

Lemma 4.1. (KL-Regularized Policy Evaluation) *For a fixed π with $\mathbb{KL}^{\pi,\mu}_s < \infty\ \forall s$, the backup operator $\mathcal{T}^\pi_{\mathbb{KL}}$ is a contraction mapping and has an unique fixed point solution $\mathcal{T}^\pi_{\mathbb{KL}}\widetilde{Q}^\pi = \widetilde{Q}^\pi$. In other words, for any $\widetilde{Q}^0 : \mathcal{S}\times\mathcal{A}\to\mathbb{R}$, define $\widetilde{Q}^{k+1} = \mathcal{T}^\pi_{\mathbb{KL}}\widetilde{Q}^k$. Then the sequence $\widetilde{Q}^k$ converges to KL-regularized Q-value function of π as $k\to\infty$.*

KL-regularized value functions have the following interpretations:

$$\widetilde{V}^{\pi}(s) = \mathbb{E}_{\pi|s}\left[\sum_{t=0}^{\infty}\gamma^t\big(r_t - \alpha(s_t)\mathbb{KL}_{s_t}^{\pi,\mu}\big)\right]$$

$$\widetilde{Q}^{\pi}(s,a) = \mathbb{E}_{\pi|s,a}\left[r_0 + \sum_{t=1}^{\infty}\gamma^t\big(r_t - \alpha(s_t)\mathbb{KL}_{s_t}^{\pi,\mu}\big)\right].$$

In the policy improvement step, we can compute a improved policy by weighting exponential $\widetilde{Q}^{\pi}$ and μ. However, when dealing with continuous state and action encountered in the later section, we may want to optimize our policy only within a set of tractable distributions Π (e.g. a set of Gaussian distributions), which requires a projection of the updated policy distribution onto Π. One of the simple ways is to adopt the information projection that minimizes the KL-divergence to the target distribution as in (Haarnoja et al., 2018):

$$\pi^{\text{new}}(\cdot|s) = \arg\min_{\pi'\in\Pi}\mathbb{KL}\left(\pi'(\cdot|s)\middle\|\frac{\exp\left(\frac{\widetilde{Q}^{\pi}(s,\cdot)}{\alpha(s)}\right)\mu(\cdot|s)}{Z^{\pi}(s)}\right) \tag{5}$$

where $Z^{\pi}(s)$ is the normalization constant. The following lemma shows that π^{new} computed by Eq. (5) always improves the value over π.

Lemma 4.2. (KL-Regularized Policy Improvement) *Given a policy $\pi \in \Pi$ and its value function $\widetilde{Q}^{\pi}$, if we update the new policy π^{new} by Eq. (5), then $\widetilde{Q}^{\pi^{\text{new}}}(s,a) \geq \widetilde{Q}^{\pi}(s,a)\ \forall s,a$.*

Lemma 4.1 and 4.2 suggest a full algorithm: *the KL-regularized policy iteration* alternates between the KL-regularized policy evaluation of Eq. (4) and the KL-regularized policy improvement of Eq. (5), and it is guaranteed to converge to the optimal policy π^* within the set of Π.

Theorem 4.2. (KL-Regularized Policy Iteration) *Suppose that $|R(s,a)| \leq R_{\max}$ and $\mathbb{KL}_s^{\pi,\mu} < \infty$. Starting from any $\pi_0 \in \Pi$, the sequence of the value functions $\widetilde{Q}^{\pi_k}$ and the improved policies π_{k+1} converge to the optimal value function and the optimal policy $\pi^* \in \Pi$, i.e. $\lim_{k\to\infty}\widetilde{Q}^{\pi_k}(s,a) \geq \widetilde{Q}^{\pi}(s,a)$ for any $\pi \in \Pi$, $s \in \mathcal{S}$, and $a \in \mathcal{A}$.*

Our theoretical result can be seen as an extension of those in entropy-regularized RL (Haarnoja et al., 2018) to KL-regularized RL, where the regularization parameter is arbitrarily given per state.

5. Batch Optimization of Policy and Hyperparameter (BOPAH)

In supervised learning, we commonly adopt regularization to select a model that generalizes well to unseen data. This is typically captured by a set of hyperparameters that balances between *approximation* and *generalization* (i.e. overfitting vs. underfitting). Commonly, the model parameters θ are optimized using the following objective function with training data $\mathcal{D}_{\text{train}}$:

$$\theta_\alpha^* = \arg\max_{\theta}\left[\mathcal{L}(\theta,\mathcal{D}_{\text{train}}) - E_\alpha(\theta)\right] \tag{6}$$

where $\mathcal{L}$ measures how well the model performs on $\mathcal{D}_{\text{train}}$ (e.g. log-likelihood $\mathcal{L}(\theta,\mathcal{D}_{\text{train}}) = \log p(\mathcal{D}_{\text{train}}|\theta)$ if probabilistic model), and E_α is a regularization term that penalizes complex models to prevent overfitting (e.g. L2 regularization $E_\alpha(\theta) = \alpha\|\theta\|_2^2$) where α is the regularization parameter that balances between approximation and generalization. Then, the regularization parameter α is treated as a hyperparemter and optimized using the held-out validation dataset $\mathcal{D}_{\text{valid}}$ which is mutually exclusive to $\mathcal{D}_{\text{train}}$:

$$\alpha^* = \arg\max_{\alpha}\mathcal{L}(\theta_\alpha^*,\mathcal{D}_{\text{valid}}) \tag{7}$$

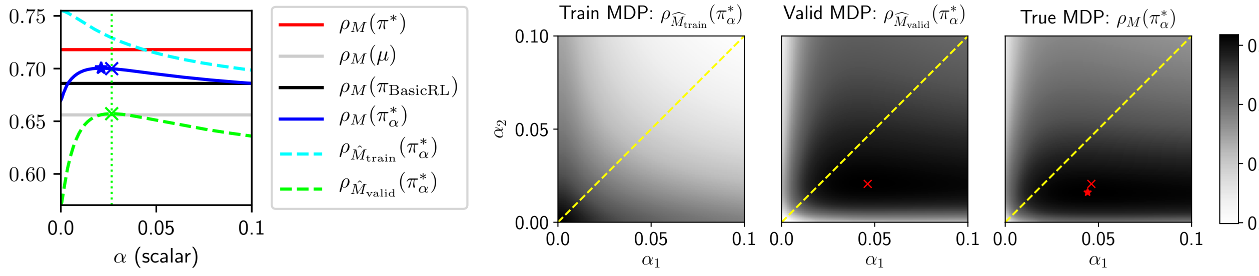
Optimizing the hyperparameter is often done using simple grid or random search (Bergstra & Bengio, 2012), but these simple methods scale poorly with the number of hyperparameters. Instead, our approach adopts the gradient-based hyperparameter optimization method (Bengio, 2000; Maclaurin et al., 2015; Pedregosa, 2016), known to be capable of many hyperparameters by using local information about the objective function, assuming that Eq. (7) is differentiable.

In this section, we introduce Batch Optimization of Policy and Hyperparameter (BOPAH), a method for batch reinforcement learning that aims to achieve the best possible performance improvement on the (not fully known) true environment. BOPAH adopts KL-regularization in policy optimization.

5.1. Model-based BOPAH

BOPAH starts by dividing the entire batch data $\mathcal{D} = \{(s_i,a_i,s_i',r_i)\}_{i=1}^N$ into two mutually exclusive sets $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{valid}}$. Each of the split datasets constructs the train (MLE) MDP $\widehat{M}_{\text{train}}$ and the valid (MLE) MDP $\widehat{M}_{\text{valid}}$ respectively. The policy is then optimized on the train MDP $\widehat{M}_{\text{train}}$ with the state-dependent KL-regularization whose introduction was justified in Section 4:

$$\pi_\alpha^* = \arg\max_{\pi}\mathbb{E}_{\pi,\widehat{M}_{\text{train}}}\left[\sum_{t=0}^{\infty}\gamma^t\Big(r_t - \alpha(s_t)\mathbb{KL}_{s_t}^{\pi,\mu}\Big)\right] \tag{8}$$



(a) Single hyperparameter: $\alpha(s) = \alpha \; \forall s$.

(b) Two hyperparameters: $\alpha(s^{\text{odd}}) = \alpha_1$ and $\alpha(s^{\text{even}}) = \alpha_2$.

Figure 1. Experimental result on a random MDP, where $\mathcal{S} = \{s^1, \dots s^{20}\}$, $|\mathcal{A}| = 4$, and α controls the degree of regularization. All the results are obtained by averaging over 300 trials. The symbol $\times$ denotes the value of α that performs the best in the valid MDP. The symbol $\star$ denotes the optimal value of α for the true MDP.

which can be solved by KL-regularized policy iteration. In Eq. (8), if the hyperparameters $\alpha(s)$'s are close to zero, the resulting policy becomes the optimal policy of the unregularized train MDP $\widehat{M}_{\text{train}}$, which would be *overfitting* to $\mathcal{D}_{\text{train}}$. In contrast, if $\alpha(s)$'s become too large, the policy is reduced to the behavior policy μ, which corresponds to *underfitting*. Our goal is to find the optimal hyperparameters of $\alpha(s)$ that balances between two extremes.

To this end, we optimize the hyperparameters on the valid MDP $\widehat{M}_{\text{valid}}$:

$$\alpha^* = \arg\max_{\alpha} \mathbb{E}_{\pi^*_\alpha, \widehat{M}_{\text{valid}}} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] \tag{9}$$

and deploy $\pi^*_{\alpha^*}$ to the real environment. Note the similarity between our framework for batch RL Eq. (8-9), and the hyperparameter optimization in supervised learning Eq. (6-7).

Illustrative Example Figure 1 highlights our approach for batch RL using a synthetic example. We constructed a single instance of random MDP M with $|\mathcal{S}| = 20$ and $|\mathcal{A}| = 4$. We collected batch data $\mathcal{D}$ consisting of 100 episodes with maximum time step 50 using the behavior policy $\mu = 0.7\pi^* + 0.3\pi_{\text{unif}}$, where π^* is the optimal policy of M and π_{unif} is the uniform random policy. As a baseline, π_{BasicRL} is obtained by computing the optimal policy of the MLE MDP using the entire data in $\mathcal{D}$. We divide $\mathcal{D}$ into two mutually exclusive sets $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{valid}}$ of same number of trajectories, and π^*_α is computed via Eq. (8) on the $\widehat{M}_{\text{train}}$, the MLE MDP using the data in $\mathcal{D}_{\text{train}}$. Finally, $\rho_{\mathcal{M}}(\pi)$ denotes the (reward) performance of policy π on MDP $\mathcal{M} \in \{M, \widehat{M}_{\text{train}}, \widehat{M}_{\text{valid}}\}$, i.e. $\rho_{\mathcal{M}}(\pi) = \sum_{\pi,\mathcal{M}} [\sum_{t=0}^{\infty} \gamma^t r_t]$.

Figure 1a visualizes the result when we used a global scalar hyperparameter, i.e. $\alpha(s) = \alpha \; \forall s$. As α increases, the performance of π^*_α in $\widehat{M}_{\text{train}}$ monotonically decreases (cyan) since $\alpha = 0$ yields the optimal policy for the $\widehat{M}_{\text{train}}$. In contrast, the performance of π^*_α in $\widehat{M}_{\text{valid}}$ (green) shows an expected trend, where too large or too small value of α deteriorates the performance. Note that the performance trend in $\widehat{M}_{\text{valid}}$ is strongly correlated with the trend in the true model M. This justifies the use of $\widehat{M}_{\text{valid}}$ for selecting α in order to perform well on (unknown) true model M.

On the other hand, note that π^*_α underperforms π_{BasicRL} in the true model when $\alpha = 0$. This is expected, since π^*_α is obtained from $\widehat{M}_{\text{train}}$ using less amount of data. However, when $\alpha^* = \arg\max_\alpha \rho_{\widehat{M}_{\text{valid}}}(\pi^*_\alpha)$ is used, $\pi^*_{\alpha^*}$ significantly outperforms π_{BasicRL}. This result supports that batch RL can benefit from hyperparameter tuning approach, a common practice in supervised learning.

We further extended the experiment setting by allowing state-dependent hyperparameters $\alpha(s)$. Figure 1b demonstrates the result when using two hyperparameters instead of one. We set $\alpha(s^{\text{odd}}) = \alpha_1$ to the states of odd index and $\alpha(s^{\text{even}}) = \alpha_2$ to the states of even index. Note that there is a strong overall correlation between $\rho_{\widehat{M}_{\text{valid}}}(\pi^*_\alpha)$ and $\rho_M(\pi^*_\alpha)$, and the optimal hyperparameter does not lie on the dotted yellow line that corresponds to the case of the global scalar hyperparameter. This result supports that the state-dependent hyperparameters can be advantageous over the global scalar hyperparameter.

5.2. Gradient-based Hyperparameter Optimization

Tuning the state-dependent hyperparameter $\alpha(s)$ via black-box optimization (e.g. grid search or random search) is intractable due to the curse of dimensionality even for a handful of states. Thus, BOPAH adopts gradient-based hyperparameter optimization, whose analytical form is provided as follows:

Theorem 5.1. *Suppose that the state-dependent function $\alpha_\xi(s)$ for Eq. (8) is parameterized by ξ* [1]*. Then, the hy-*

[1] For example, if $\forall s \; \alpha_\xi(s) = \xi$ such that $\nabla_\xi \alpha_\xi(s) = 1$, this reduces to single hyperparameter α. On the other hand, if $\alpha_\xi(s) =$

pergradient $\nabla_\xi \rho_{\widehat{M}_{\text{valid}}}(\pi_\xi^*) = \nabla_\xi \mathbb{E}_{\pi_\xi^*, \widehat{M}_{\text{valid}}}\left[\sum_{t=0}^\infty \gamma^t r_t\right]$ *is given by:*

$$\nabla_\xi \rho_{\widehat{M}_{\text{valid}}}(\pi_\xi^*) = \mathbb{E}_{\pi_\xi^*, \widehat{M}_{\text{valid}}}\left[\sum_{t=0}^{\infty} \gamma^t \boldsymbol{\chi}_\xi(s_t)\right] \tag{10}$$

s.t.

$$\begin{aligned}
&\pi_\xi^* \triangleq \arg\max_\pi \mathbb{E}_{\pi, \widehat{M}_{\text{train}}}\left[\sum_{t=0}^\infty \gamma^t \Big(r_t - \alpha_\xi(s_t)\mathbb{KL}_{s_t}^{\pi,\mu}\Big)\right] \\
&\nabla_\xi \widetilde{Q}_{\widehat{M}_{\text{train}}}^{\pi_\xi^*}(s,a) \triangleq \mathbb{E}_{\pi_\xi^*, \widehat{M}_{\text{train}}}\left[\sum_{t=1}^\infty \gamma^t \Big(-\nabla_\xi \alpha_\xi(s_t)\mathbb{KL}_{s_t}^{\pi_\xi^*,\mu}\Big)\right] \\
&\widetilde{Q}_{\widehat{M}_{\text{train}}}^{\pi_\xi^*}(s,a) \triangleq \mathbb{E}_{\pi_\xi^*, \widehat{M}_{\text{train}}}\left[\sum_{t=0}^\infty \gamma^t \Big(r_t - \alpha_\xi(s_t)\mathbb{KL}_{s_t}^{\pi,\mu}\Big)\right] \\
&Q_{\widehat{M}_{\text{valid}}}^{\pi_\xi^*}(s,a) \triangleq \mathbb{E}_{\pi_\xi^*, \widehat{M}_{\text{valid}}}\left[\sum_{t=0}^\infty \gamma^t r_t\right] \\
&\boldsymbol{\chi}_\xi(s) \triangleq \frac{1}{\alpha_\xi(s)^2}\text{cov}_{a\sim\pi_\xi^*}\Big[\alpha_\xi(s)\nabla_\xi \widetilde{Q}_{\widehat{M}_{\text{train}}}^{\pi_\xi^*}(s,a) \\
&\qquad\qquad -\nabla_\xi \alpha_\xi(s)\widetilde{Q}_{\widehat{M}_{\text{train}}}^{\pi_\xi^*}(s,a), Q_{\widehat{M}_{\text{valid}}}^{\pi_\xi^*}(s,a)\Big]
\end{aligned}$$

where $\text{cov}_a[\mathbf{f}(a), g(a)]_i = \mathbb{E}[f_i(a)g(a)] - \mathbb{E}[f_i(a)]\mathbb{E}[g(a)]$ *denotes element-wise covariance between the vector* $\mathbf{f}(\cdot)$ *and the scalar* $g(\cdot)$.

In the above, π_ξ^* can be obtained by KL-regularized policy iteration, and $\nabla_\xi \widetilde{Q}_{\widehat{M}_{\text{train}}}^{\pi_\xi^*}(s,a)$ and $\nabla_\xi \rho_{\widehat{M}_{\text{valid}}}(\pi_\xi^*)$ can be computed by a standard policy evaluation technique with auxiliary reward functions $R_1(s,a) \triangleq -\nabla_\xi \alpha_\xi(s)\mathbb{KL}_s^{\pi_\xi^*,\mu}$ and $R_2(s,a) \triangleq \boldsymbol{\chi}(s)$, respectively. Finally, we can optimize the hyperparameters via the hypergradient by iterating the following until convergence: $\xi \leftarrow \xi + \eta \nabla_\xi \rho_{\widehat{M}_{\text{valid}}}(\pi_\xi^*)$ where ξ is a parameter for the state-dependent function $\alpha_\xi(s)$, and η is a learning rate.

This completes the description of BOPAH, which iteratively alternates between policy optimization and gradient-based hyperparameter optimization.

Remark While the definition of $\boldsymbol{\chi}(s)$ from Theorem 5.1 is hard to interpret as is, it can be reduced to simple expression when $\alpha_\xi(s)$ is state-independent and provides an additional intuition on the behavior of the hypergradient. When $\alpha_\xi(s)$ is state-independent, i.e. $\alpha_\xi(s) = \xi$, the auxiliary value function $\nabla_\xi \widetilde{Q}_{\widehat{M}_{\text{train}}}^{\pi_\xi^*}(s,a)$ becomes a simple discounted sum of KL-divergences:

$$\nabla_\xi \widetilde{Q}_{\widehat{M}_{\text{train}}}^{\pi_\xi^*}(s,a) = -\mathbb{E}_{\pi_\xi^*, \widehat{M}_{\text{train}}}\left[\sum_{t=1}^\infty \gamma^t \mathbb{KL}_{s_t}^{\pi_\xi^*,\mu}\right] \tag{11}$$

which further reduces $\boldsymbol{\chi}(s)$ by canceling out the regularization term of the KL-regularized value function to become:

$$\boldsymbol{\chi}_\xi(s) = -\frac{1}{\alpha_\xi(s)}\text{cov}_{a\sim\pi_\xi^*}\left[Q_{\widehat{M}_{\text{train}}}^{\pi_\xi^*}(s,a), Q_{\widehat{M}_{\text{valid}}}^{\pi_\xi^*}(s,a)\right] \tag{12}$$

where $Q_{\widehat{M}_{\text{train}}}^{\pi_\xi^*}(s,a) = \mathbb{E}_{\pi_\xi^*, \widehat{M}_{\text{train}}}\left[\sum_{t=0}^\infty \gamma^t r_t\right]$, the unregularized value function under train MDP. The auxiliary reward function $\boldsymbol{\chi}(s)$ is now a negative covariance between unregularized value functions of train and valid MDPs.

Consider a scenario where a learned policy only exploits reliable state-action pairs (i.e. state-action pairs that appear frequently in the dataset). In this case, two unregularized value functions $Q_{\widehat{M}_{\text{train}}}^{\pi_\xi^*}(s,a)$ and $Q_{\widehat{M}_{\text{valid}}}^{\pi_\xi^*}(s,a)$ should be similar under the policy distribution, resulting in positive covariance. Then, the hypergradient, which is a discounted sum of negative covariance, will become negative, the alpha will decrease, and the policy will be less regularized to explore more state-action pairs. On the other hand, when the policy tries to explore uncertain state-action pairs where two unregularized value functions differ, the hypergradient descent will regularize policy more to finally converge to the appropriate level of regularization.

6. Actor-Critic BOPAH

In this section, we extend BOPAH to control tasks with continuous state and action spaces via practical approximations to KL-regularized policy iteration and hypergradient computation. We achieve this goal by adopting the Actor-Critic framework, where we train the parametric models of policies (i.e. actor), KL-regularized value functions (i.e. critic), and state-dependent hyperparameters. The resulting algorithm, Actor-Critic BOPAH (AC-BOPAH), thus performs alternating updates of the three parametric models.

6.1. Parametric Model of State-dependent Hyperparameters

Although in principle, we could use any parametric model to represent state-dependent hyperparameters, we focus on analyzing the linear model for $\alpha_\xi(s)$, which particularly allows for stable training of KL-regularized value functions. To see this, suppose

$$\alpha_\xi(s) \triangleq \sum_{i=1}^{d_\xi} \xi_i \phi_i(s) = \xi^\top \phi(s) \tag{13}$$

where $\xi = [\xi_1, \ldots, \xi_{d_\xi}]^\top$, and $\phi : \mathcal{S} \to \mathbb{R}^{d_\xi}$ is a feature function of states such that $\xi^\top \phi(s) \geq 0$ for all s (e.g. $\xi \geq 0$ and $\phi_i(s)$'s are RBFs). The feature function $\phi(s)$ is predefined and fixed, and only ξ is optimized during training.

This linear parameterization leads to the following decom-

ξ_s such that $\nabla_{\xi_{s_i}} \alpha_\xi(s_j) = \mathbb{1}(s_i = s_j)$, this corresponds to the fully state-dependent hyperparameters $\alpha(s)$.

position of the KL-regularized value function:

$$\begin{aligned}\widetilde{Q}^{\pi}_{\widehat{M}_{\text{train}}}(s,a) &= \mathbb{E}\Big[r_0 + \sum_{t=1}^{\infty}\gamma^t\Big(r_t - \alpha_\xi(s_t)\mathbb{KL}^{\pi,\mu}_{s_t}\Big)\Big] \\ &= \underbrace{\mathbb{E}\Big[\sum_{t=0}^{\infty}\gamma^t r_t\Big]}_{\triangleq\, Q^{\pi}_{\theta_{\text{train}}}(s,a)} + \xi^\top \underbrace{\mathbb{E}\Big[\sum_{t=1}^{\infty}\gamma^t\big(-\phi(s)\mathbb{KL}^{\pi,\mu}_{s_t}\big)\Big]}_{\triangleq\, \mathbf{Q}^{\pi}_{\theta^{\mathbb{KL}}_{\text{train}}}(s,a)} \quad (14)\end{aligned}$$

Note in Eq. (14) that both $Q^{\pi}_{\theta_{\text{train}}}(s,a)$ and $\mathbf{Q}^{\pi}_{\theta^{\mathbb{KL}}_{\text{train}}}(s,a)$ are not dependent on ξ but only on π. Thus, we could train the separate models for $Q^{\pi}_{\theta_{\text{train}}}(s,a)$ and $\mathbf{Q}^{\pi}_{\theta^{\mathbb{KL}}_{\text{train}}}(s,a)$ instead of the single model for $\widetilde{Q}^{\pi}_{\widehat{M}_{\text{train}}}(s,a)$, making the training more stable since they are not affected by the change in the hyperparameters ξ.

Furthermore, the auxiliary value function $\nabla_\xi \widetilde{Q}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s,a)$ in Theorem 5.1 is also directly derived using $\mathbf{Q}^{\pi}_{\theta^{\mathbb{KL}}_{\text{train}}}(s,a)$ without any further introduction of parameterized functions since $\nabla_\xi \alpha_\xi(s) = \phi(s)$:

$$\nabla_\xi \widetilde{Q}^{\pi}_{\widehat{M}_{\text{train}}}(s,a) = \mathbf{Q}^{\pi}_{\theta^{\mathbb{KL}}_{\text{train}}}(s,a) \quad (15)$$

We also train the action-value critic for the validation-set MDP, defined in the standard way:

$$Q^{\pi}_{\theta_{\text{valid}}}(s,a) \triangleq \mathbb{E}_{\pi,\widehat{M}_{\text{valid}}}\left[\sum_{t=0}^{\infty}\gamma^t r_t\right] \quad (16)$$

Finally, we define state-value critics $V^{\pi}_{\psi_{\text{train}}}(s)$, $\mathbf{V}_{\psi^{\mathbb{KL}}_{\text{train}}}(s)$, and $V^{\pi}_{\psi_{\text{valid}}}(s)$ similarly to Eq. (14) and Eq. (16), which completes our parameterization of the value functions and the hyperparameters for AC-BOPAH. As for the actor $\pi_\omega(a|s)$, we use the Gaussian policy with its mean and covariance parameterized by neural networks.

6.2. Objective Functions

We now present the objective functions to train each parametric model for the actor and the critic with fixed hyperparameters. First, the parameters for the reward value functions, $\{\psi_{\text{train}}, \theta_{\text{train}}, \psi_{\text{valid}}, \theta_{\text{valid}}\}$, are trained by minimizing the squared residual errors: for each $\text{data} \in \{\text{train}, \text{valid}\}$,

$$\begin{aligned}J(\psi_{\text{data}}) &= \mathbb{E}_{\substack{s\sim\mathcal{D}_{\text{data}}\\ a\sim\pi_\omega}}\Big[\big(V_{\psi_{\text{data}}}(s) - Q_{\theta_{\text{data}}}(s,a)\big)^2\Big] \\ J(\theta_{\text{data}}) &= \mathbb{E}_{\substack{(s,a,r,s')\\ \sim\mathcal{D}_{\text{data}}}}\Big[\big(Q_{\theta_{\text{data}}}(s,a) - r - \gamma V_{\bar{\psi}_{\text{data}}}(s')\big)^2\Big]\end{aligned}$$

where $\bar{\psi}_{\text{data}}$ is an exponential moving average of ψ_{data} (i.e. soft target update). Similarly, the parameters for the KL value functions $\{\psi^{\mathbb{KL}}_{\text{train}}, \theta^{\mathbb{KL}}_{\text{train}}\}$ are trained by minimizing:

$$\begin{aligned}J(\psi^{\mathbb{KL}}_{\text{train}}) &= \mathbb{E}_{\substack{s\sim\mathcal{D}_{\text{train}}\\ a\sim\pi_\omega}}\big\|\mathbf{V}_{\psi^{\mathbb{KL}}_{\text{train}}}(s) - \mathbf{Q}_{\theta^{\mathbb{KL}}_{\text{train}}}(s,a) + \phi(s)\mathbb{KL}^{\pi,\mu}_s\big\|_2^2 \\ J(\theta^{\mathbb{KL}}_{\text{train}}) &= \mathbb{E}_{\substack{(s,a,s')\\ \sim\mathcal{D}_{\text{train}}}}\big\|\mathbf{Q}_{\theta^{\mathbb{KL}}_{\text{train}}}(s,a) - \gamma\mathbf{V}_{\bar{\psi}^{\mathbb{KL}}_{\text{train}}}(s')\big\|_2^2\end{aligned}$$

where $\bar{\psi}^{\mathbb{KL}}_{\text{train}}$ is an exponential moving average of $\psi^{\mathbb{KL}}_{\text{train}}$. Then, the policy parameters are optimized by minimizing the expected KL-divergence of Eq. (5), which yields:

$$J(\omega) = \mathbb{E}_{\substack{s\sim\mathcal{D}_{\text{train}}\\ a\sim\pi_\omega}}\Big[\alpha_\xi(s)\mathbb{KL}^{\pi,\mu}_s - Q^{\pi}_{\theta_{\text{train}}}(s,a) - \xi^\top\mathbf{Q}^{\pi}_{\theta^{\mathbb{KL}}_{\text{train}}}(s,a)\Big]$$

Here, we use the analytic formula for computing the KL-divergence $\mathbb{KL}^{\pi,\mu}_s$ and adopt the reparameterization trick to the Gaussian policy when computing the gradient in order to reduce the variance of stochastic gradients. We also perform conservative training with bootstrapped Q and apply gradient penalty of critic networks. Without them, the algorithm is prone to divergence during training due to the overestimation of the uncertain state-action. Note that other batch RL algorithms (Kumar et al., 2019) use similar conservative estimation techniques for stabilization of training. The gradient penalty constrains the Lipschitz constant of the critic, which prevents outputting extremely high value for the uncertain region that can be encountered by weak behavior-regularization during hyperparameter optimization. More technical details for the experiments can be found in the Appendix E. Iterative optimization of $\{J(\psi_{\text{data}}), J(\theta_{\text{data}}), J(\psi^{\mathbb{KL}}_{\text{train}}), J(\theta^{\mathbb{KL}}_{\text{train}})\}$ and $J(\omega)$ results in an actor-critic algorithm that performs approximate KL-regularized policy iteration. We will denote this actor-critic algorithm as KLAC in the experiments when we use fixed hyperparameters.

6.3. Clipped Importance Sampling for Hypergradients

Finally, we compute the approximate hypergradients by exploiting the current actor and critics as the approximate solutions of the KL-regularized MDP with respect to the current hyperparameter ξ. From Eq. (10) and Eq. (14-15),

$$\nabla_\xi \rho_{\widehat{M}_{\text{valid}}}(\pi) = \mathbb{E}_{\pi,\widehat{M}_{\text{valid}}}\left[\sum_{t=0}^{\infty}\gamma^t\boldsymbol{\chi}_\xi(s_t)\right] \quad (17)$$

$$\begin{aligned}\text{s.t. } \boldsymbol{\chi}_\xi(s) = \frac{1}{\big(\xi^\top\phi(s)\big)^2}\operatorname{cov}_{a\sim\pi}\Big[&(\xi^\top\phi(s))\mathbf{Q}^{\pi}_{\theta^{\mathbb{KL}}_{\text{train}}}(s,a) \\ &-\phi(s)\big(Q^{\pi}_{\theta_{\text{train}}}(s,a) + \xi^\top\mathbf{Q}^{\pi}_{\theta^{\mathbb{KL}}_{\text{train}}}(s,a)\big), Q^{\pi}_{\theta_{\text{valid}}}(s,a)\Big]\end{aligned}$$

Here, for immediate (off-policy) policy evaluation of π with respect to the auxiliary reward $\boldsymbol{\chi}_\xi(s_t)$ using the validation trajectory collected by μ, we adopt a clipped importance sampling. For the validation dataset $\mathcal{D}_{\text{valid}} = \{\tau_1, \ldots, \tau_{N_\tau}\}$ such that $\tau_n = \{(s^n_t, a^n_t, r^n_t, s'^n_t)\}_{t=0}^{T_n}$, we

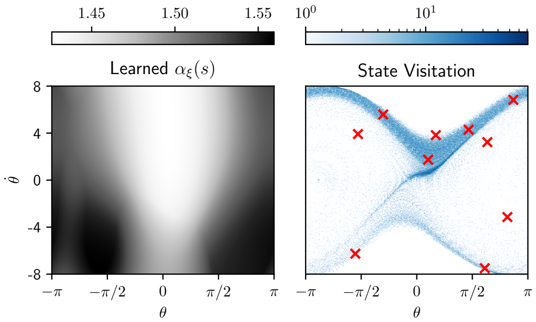


Figure 2. Example of the learned $\alpha_\xi(s)$ in Pendulum-v0. The symbol × denotes the representative states for center of RBFs, obtained from the cover tree algorithm.

estimate the approximate hypergradient as follows:

$$\nabla_\xi \rho_{\widehat{M}_{\text{valid}}}(\pi) \approx \frac{1}{N_\tau} \sum_{n=1}^{N_\tau} \sum_{t=0}^{T_n} \gamma^t w_{0:t}^n \boldsymbol{\chi}_\xi(s_t) \tag{18}$$

$$\text{s.t. } w_{0:t}^n \triangleq \text{clip}\left(\left(\prod_{k=0}^{t} \frac{\pi(a_k^n|s_k^n)}{\mu(a_k^n|s_k^n)} \right), w_{\min}, w_{\max} \right),$$

and update the hyperparameter via hypergradient ascent. In practice, we alternate between H_f steps of optimizing the actor-critic (inner-problem) and one step of optimizing the hyperparameter (outer-problem). We refer this algorithm as AC-BOPAH.

Illustrative Example Figure 2 visualizes $\alpha_\xi(\cdot)$ optimized by AC-BOPAH in Pendulum-v0. We sampled trajectories of 10^3 episodes using a suboptimal behavior policy which was partially trained for only 10^4 steps by soft actor-critic (SAC) (Haarnoja et al., 2018). We selected 10 representative states within the batch data via the cover tree algorithm (Beygelzimer et al., 2006) and used them for the basis of each RBF $\phi_i(s)$ in $\alpha(s)$. Finally, we run AC-BOPAH for 10^6 steps.

As we can inspect from the left heatmap in Figure 2, the optimized $\alpha_\xi(\cdot)$ has lower values in the densely collected state region while it has higher values in the sparsely collected state region. This is a desirable result: we can be safely deviate from the behavior policy for optimization in experience-rich areas but should be conservative and fall back to the behavior policy in experience-sparse areas of the state space.

7. Experiments

7.1. Model-based BOPHA on Random MDPs

In order to probe how safely and efficiently BOPAH can improve performance over the behavior policy with respect to the varying number of trajectories and optimality of the behavior policy, we conducted repeated experiments using randomly generated MDPs. The experimental protocol follows that of (Laroche et al., 2019), and details can be found in the Appendix F. In essence, we repeated 10k runs, where each random MDP M was created with $|S| = 50$, $|A| = 4$, $\gamma = 0.95$, where the maximum episode length was set to 50. The ζ-optimal behavior policy μ denotes $\rho_M(\mu) = \zeta\rho_M(\pi^*) + (1-\zeta)\rho_M(\pi_{\text{unif}})$.

We compare the model-based BOPAH with four algorithms: (1) BasicRL (simple baseline appeared in 5.1), (2) Reward-adjusted MDP (RaMDP) (Petrik et al., 2016), (3) Robust MDP (Nilim & El Ghaoui, 2005; Iyengar, 2005), (4) SPIBB (Laroche et al., 2019), using various sizes of trajectories and two optimalities of behavior policy (near-optimal and near-random policies). For each run, we measure the normalized performance of π: $\bar{\rho}_M(\pi) = \frac{\rho_M(\pi)-\rho_M(\mu)}{\rho_M(\pi^*)-\rho_M(\mu)} \in (-\infty, 1]$, which measures how much the algorithm improves its performance over the behavior policy. Finally, we report the mean (normalized) performance and the conditional value at risk performance (CVaR). The $x\%$-CVaR denotes the mean (normalized) performance of the worst $x\%$ runs.

Figure 3a-3b presents the result when the behavior policy is near-optimal ($\zeta = 0.9$), where the behavior policy is nearly deterministic, thus the collected trajectories could cover only part of the entire state-action space. On the other hand, Figure 3c-3d represents the results when the behavior policy is more randomized ($\zeta = 0.5$), thus the collected trajectories could cover the entire state-action space fairly well. In both settings, BOPAH with state-dependent hyperparameters consistently matched or exceeded other algorithms, which highlights effectiveness and robustness of our algorithm.

7.2. AC-BOPAH on Continuous Control Tasks

In this experiment, we evaluate the effectiveness of AC-BOPAH on continuous control tasks, using the MuJoCo environments in the OpenAI gym (Todorov et al., 2012; Brockman et al., 2016). We first obtained the behavior policy by running SAC (Haarnoja et al., 2018) for half-million steps and then prepared the dataset consisting of 10^3 trajectories shared by all algorithms. For AC-BOPAH, we held out 20% of trajectories as the validation set. We compare our AC-BOPAH with two behavior cloning baselines, one with the Gaussian policy (BC) and the other one with the variational autoencoder (VAE-BC). We also compare with two state-of-the-art batch deep RL algorithms for continuous control problems, BCQ (Fujimoto et al., 2019) and BEAR-QL (Kumar et al., 2019). We used their published code and hyperparameters ($\Phi = 0.05$ for BCQ and $\epsilon = 0.05$ for

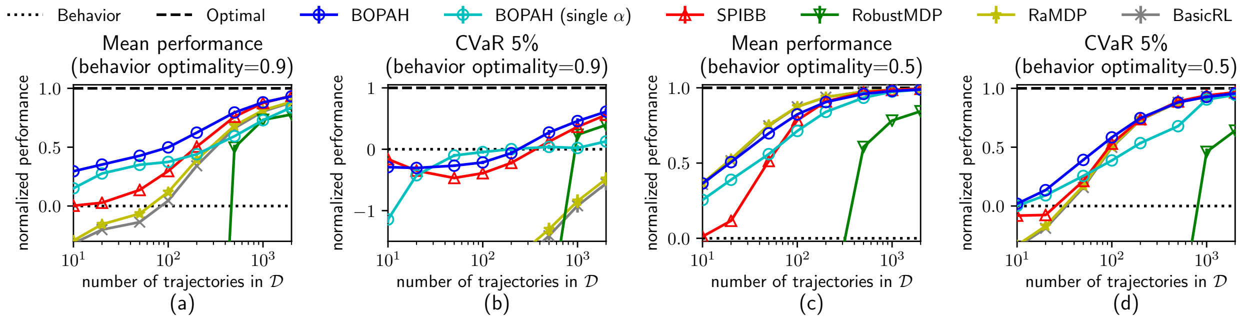


Figure 3. The result from random MDP experiments.

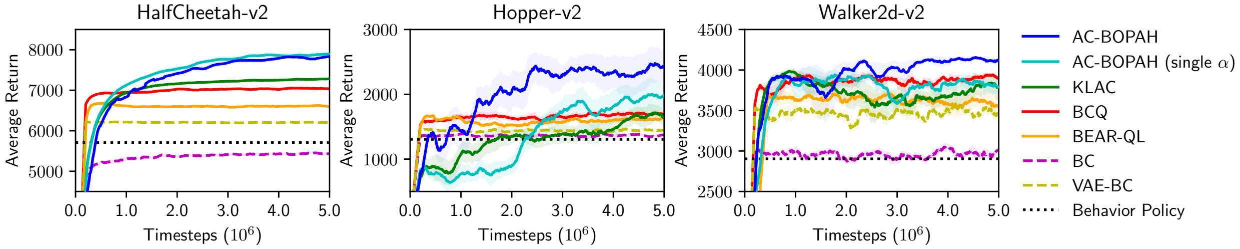


Figure 4. The results of continuous control experiments averaged over 5 trials, which are moving-averaged with a window size of 20. The shaded area represents the standard error.

BEAR-QL) therein for obtaining experimental results.[2] We report two versions of our algorithm, AC-BOPAH (single α) that uses a global hyperparameter α and AC-BOPAH that uses state-dependent hyperparameter $\alpha_\xi(s)$ with $|\xi| = 21$. Both versions are initialized to start from $\alpha_\xi(s) = 10^2 \; \forall s$. KLAC is the KL-regularized actor-critic with $\alpha = 10^2$ held constant for all tasks.

As presented in Figure 4, AC-BOPAH consistently outperformed the state-of-the-art algorithms by large margins. While the KLAC was on a par with other algorithms, AC-BOPAH made further improvement by optimizing the hyperparameters using the held-out validation set. AC-BOPAH (with state-dependent KL-regularization) shows clear improvement over the constant α version, except for the HalfCheetah-v2 domain where the latter already achieved near-optimal performance.

8. Conclusion

In this work, we presented the generalized KL-regularization and the BOPAH framework for batch RL, which propose the optimization of state-dependent regularization via the hypergradient ascent. We provided a formal analysis that motivates the objective used in BOPAH, and presented two concrete versions, (1) model-based BOPAH that assumes tabular environment and computes exact hypergradients, and (2) AC-BOPAH that uses actor-critic architecture to compute approximate hypergradients in more challenging continuous tasks. We empirically demonstrated that both model-based BOPAH and AC-BOPAH outperform the state-of-the-art algorithms, supporting the hypothesis that batch RL can significantly benefit from hyperparameter optimization. While we introduced BOPAH with the KL-regularization for batch RL in this work, we believe that our BOPAH framework can be extended to other related problems in RL with limited experience data, which shall be interesting direction for future work.

Acknowledgments

This work was supported by the National Research Foundation (NRF) of Korea (NRF-2019R1A2C1087634 and NRF-2019M3F2A1072238), the Ministry of Science and Information communication Technology (MSIT) of Korea (IITP No. 2020-0-00940, IITP 2019-0-00075 and IITP No. 2017-0-01779 XAI), and POSCO.

[2] For more thorough comparison, we also conducted additional experiments with hyperparameter grid-search for BCQ and BEAR as in (Wu et al., 2019), i.e. $\Phi \in \{0.005, 0.015, 0.05, 0.15, 0.5\}$ for BCQ and $\epsilon \in \{0.015, 0.05, 0.15, 0.5, 1.5\}$ for BEAR-QL. We observed improvement only for BCQ in Halfcheetah (with $\Phi = 0.5$), which managed to perform better than AC-BOPAH. However, such tuning requires access to the true environment, which is not feasible in the batch RL setting.

References

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In *International Conference on Machine Learning*, pp. 22–31, 2017.

Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. *International Conference on Machine Learning*, 2020.

Bengio, Y. Gradient-based optimization of hyperparameters. *Neural Computation*, 12(8):1889–1900, August 2000.

Bergstra, J. and Bengio, Y. Random search for hyperparameter optimization. *Journal of Machine Learning Research*, 13:281–305, February 2012. ISSN 1532-4435.

Beygelzimer, A., Kakade, S., and Langford, J. Cover trees for nearest neighbor. In *International Conference on Machine Learning*, pp. 97–104, 2006.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

Fedus*, W., Rosca*, M., Lakshminarayanan, B., Dai, A. M., Mohamed, S., and Goodfellow, I. Many paths to equilibrium: GANs do not need to decrease a divergence at every step. In *International Conference on Learning Representations*, 2018.

Fox, R., Pakman, A., and Tishby, N. Taming the noise in reinforcement learning via soft updates. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, pp. 202–211, 2016.

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

Galashov, A., Jayakumar, S., Hasenclever, L., Tirumala, D., Schwarz, J., Desjardins, G., Czarnecki, W. M., Teh, Y. W., Pascanu, R., and Heess, N. Information asymmetry in KL-regularized RL. In *International Conference on Learning Representations*, 2019.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pp. 5769–5779, 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1856–1865, 2018.

Hasselt, H. V. Double q-learning. In *Advances in Neural Information Processing Systems*, pp. 2613–2621, 2010.

Iyengar, G. N. Robust dynamic programming. *Mathematics of Operations Research*, 30(2):257–280, May 2005.

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*, pp. 12498–12509, 2019.

Jaques, N., Ghandeharioun, A., Shen, J. H., Ferguson, C., Lapedriza, À., Jones, N., Gu, S., and Picard, R. W. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *CoRR*, abs/1907.00456, 2019.

Kappen, H. J., Gómez, V., and Opper, M. Optimal control as a graphical model inference problem. *Machine Learning*, 87(2):159–182, 2012.

Kodali, N., Abernethy, J., Hays, J., and Kira, Z. On convergence and stability of gans, 2017.

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

Laroche, R., Trichelair, P., and Des Combes, R. T. Safe policy improvement with baseline bootstrapping. In *International Conference on Machine Learning*, pp. 3652–3661, 2019.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

Maclaurin, D., Duvenaud, D., and Adams, R. P. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*, pp. 2113–2122, 2015.

Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 1054–1062, 2016.

Nilim, A. and El Ghaoui, L. Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, September 2005.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep exploration via bootstrapped dqn. In *Advances in Neural Information Processing Systems*, pp. 4026–4034, 2016.

Pedregosa, F. Hyperparameter optimization with approximate gradient. In *International Conference on Machine Learning*, pp. 737–746, 2016.

Petrik, M., Ghavamzadeh, M., and Chow, Y. Safe policy improvement by minimizing robust baseline regret. In *Advances in Neural Information Processing Systems*, pp. 2298–2306, 2016.

Pfau, D. and Vinyals, O. Connecting generative adversarial networks and actor-critic methods, 2016.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.

Schulman, J., Chen, X., and Abbeel, P. Equivalence between policy gradients and soft q-learning, 2017.

Siegel, N., Springenberg, J. T., Berkenkamp, F., Abdolmaleki, A., Neunert, M., Lampe, T., Hafner, R., Heess, N., and Riedmiller, M. Keep doing what worked: Behavior modelling priors for offline reinforcement learning. In *International Conference on Learning Representations*, 2020.

Sun, W., Gordon, G. J., Boots, B., and Bagnell, J. Dual policy iteration. In *Advances in Neural Information Processing Systems*, pp. 7059–7069, 2018.

Todorov, E. Linearly-solvable markov decision problems. In *Advances in Neural Information Processing Systems*, pp. 1369–1376, 2007.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. *ArXiv*, abs/1911.11361, 2019.

Batch Reinforcement Learning with Hyperparameter Gradients -Supplementary Material-

A. Proof for Theorem 4.1

For the proof of Theorem 4.1, we additionally define the following terms:

$[\mathbf{d}_0]_s$	$\triangleq$	$d_0(s)$, the initial state probability vector
$[\mathbf{d}^\pi_M]_s$	$\triangleq$	$d^\pi(s)$, the marginal discounted state probability vector induced by π and P
$[\mathbf{d}^\pi_{\widehat{M}}]_s$	$\triangleq$	$d^\pi_{\widehat{M}}(s)$, the marginal discounted state probability vector induced by π and $\widehat{P}$
$[\mathbf{P}^\pi]_{s,s'}$	$\triangleq$	$\sum_a P(s'\|s,a)\pi(a\|s)$, the transition matrix following policy π
$[\widehat{\mathbf{P}}^\pi]_{s,s'}$	$\triangleq$	$\sum_a \widehat{P}(s'\|s,a)\pi(a\|s)$, the approximate transition matrix following policy π
$[\mathbf{r}^\pi]_s$	$\triangleq$	$\sum_a R(s,a)\pi(a\|s)$, the reward vector following policy π

Lemma A.1. *Given that all rewards are bounded $|R(s,a)| \leq R_{\max}$, the error of the estimate based on an approximate model, i.e. $\rho_{\widehat{M}}(\mu)$, is bounded by:*

$$\left|\rho_M(\mu) - \rho_{\widehat{M}}(\mu)\right| \leq \frac{2\gamma R_{\max}}{(1-\gamma)^2} \mathbb{E}_{\substack{s\sim d^\mu_M \\ a\sim\mu}} \left[\mathbb{TV}\left(P(s'|s,a)||\widehat{P}(s'|s,a)\right)\right]. \tag{19}$$

Proof. Here, for simplicity, we omit the subscript μ in $\mathbf{r}, \mathbf{P}$, and $\widehat{\mathbf{P}}$.

$$\begin{aligned}
\left|\rho_M(\mu) - \rho_{\widehat{M}}(\mu)\right| &= \left|\mathbf{r}^\top \left[\left(I-\gamma\mathbf{P}\right)^{-1} - \left(I-\gamma\widehat{\mathbf{P}}\right)^{-1}\right]\mathbf{d}_0\right| = \left|\mathbf{r}^\top \left(I-\gamma\widehat{\mathbf{P}}\right)^{-1}\left[\gamma\mathbf{P}-\gamma\widehat{\mathbf{P}}\right](I-\gamma\mathbf{P})^{-1}\mathbf{d}_0\right| \\
&= \gamma\left|\mathbf{r}^\top\left(I-\gamma\widehat{\mathbf{P}}\right)^{-1}\left[\mathbf{P}-\widehat{\mathbf{P}}\right]\frac{\mathbf{d}^\mu_M}{1-\gamma}\right| \\
&\leq \frac{\gamma}{1-\gamma}\left\|\mathbf{r}^\top\left(I-\gamma\widehat{\mathbf{P}}\right)^{-1}\right\|_\infty \left\|\left[\mathbf{P}-\widehat{\mathbf{P}}\right]\mathbf{d}^\mu_M\right\|_1 \\
&\leq \frac{\gamma R_{\max}}{(1-\gamma)^2}\left\|\left[\mathbf{P}-\widehat{\mathbf{P}}\right]\mathbf{d}^\mu_M\right\|_1 = \frac{\gamma R_{\max}}{(1-\gamma)^2}\sum_{s'}\left|\sum_{s,a}\left[P(s'|s,a)-\widehat{P}(s'|s,a)\right]\mu(a|s)d^\mu_M(s)\right| \\
&\leq \frac{\gamma R_{\max}}{(1-\gamma)^2}\sum_{s',a,s}\left|P(s'|s,a)-\widehat{P}(s'|s,a)\right|\mu(a|s)d^\mu_M(s) \\
&= \frac{2\gamma R_{\max}}{(1-\gamma)^2}\mathbb{E}_{\substack{s\sim d^\mu_M \\ a\sim\mu}}\left[\mathbb{TV}\left(P(s'|s,a)||\widehat{P}(s'|s,a)\right)\right]
\end{aligned}$$

□

Lemma A.2. *Given that all rewards are bounded $|R(s,a)| \leq R_{\max}$, we can bound the difference of the evaluations of two policies π, μ as:*

$$|\rho_M(\pi) - \rho_M(\mu)| \leq \frac{2R_{\max}}{(1-\gamma)^2}\mathbb{E}_{s\sim d^\pi_M}\left[\mathbb{TV}(\pi(a|s)||\mu(a|s))\right]. \tag{20}$$

Proof.

$$\begin{aligned}|\rho_M(\pi) - \rho_M(\mu)| &= \left|\left[(\mathbf{r}^\pi)^\top (I - \gamma \mathbf{P}^\pi)^{-1} - (\mathbf{r}^\mu)^\top (I - \gamma \mathbf{P}^\mu)^{-1}\right] \mathbf{d}_0\right| \\ &\leq \left|(\mathbf{r}^\pi - \mathbf{r}^\mu)^\top \frac{\mathbf{d}_M^\pi}{1-\gamma}\right| + \left|(\mathbf{r}^\mu)^\top \left[(I - \gamma \mathbf{P}^\pi)^{-1} - (I - \gamma \mathbf{P}^\mu)^{-1}\right] \mathbf{d}_0\right|\end{aligned}$$

$$\left|(\mathbf{r}^\pi - \mathbf{r}^\mu)^\top \frac{\mathbf{d}_M^\pi}{1-\gamma}\right| \leq \frac{2R_{\max}}{1-\gamma} \mathbb{E}_{s\sim d_M^\pi}[\mathbb{TV}(\pi(a|s)||\mu(a|s))]$$

$$\begin{aligned}\left|(\mathbf{r}^\mu)^\top \left[(I - \gamma \mathbf{P}^\pi)^{-1} - (I - \gamma \mathbf{P}^\mu)^{-1}\right] \mathbf{d}_0\right| &\leq \frac{\gamma R_{\max}}{(1-\gamma)^2} \left\|\left[\mathbf{P}^\pi - \mathbf{P}^\mu\right] \mathbf{d}^\pi\right\|_1 \\ &= \frac{\gamma R_{\max}}{(1-\gamma)^2} \sum_{s'} \left|\sum_{s,a} P(s'|s,a) \left[\pi(a|s) - \mu(a|s)\right] d_M^\pi(s)\right| \\ &\leq \frac{\gamma R_{\max}}{(1-\gamma)^2} \sum_{a,s} |\pi(a|s) - \mu(a|s)|\, d_M^\pi(s) \\ &= \frac{2\gamma R_{\max}}{(1-\gamma)^2} \mathbb{E}_{s\sim d_M^\pi} \left[\mathbb{TV}(\pi(a|s)||\mu(a|s))\right]\end{aligned}$$

□

Theorem 4.1. *Define the errors* $\epsilon_M^\pi = \mathbb{E}_{s\sim d_M^\pi}\left[\mathbb{TV}(\pi(a|s)||\mu(a|s))\right]$, $\epsilon_{\widehat{M}}^\pi = \mathbb{E}_{s\sim d_{\widehat{M}}^\pi}\left[\mathbb{TV}(\pi(a|s)||\mu(a|s))\right]$ *and* $\epsilon_M^P = \mathbb{E}_{\substack{s\sim d_M^\mu \\ a\sim\mu}}\left[\mathbb{TV}\left(P(s'|s,a)\big|\big|\widehat{P}(s'|s,a)\right)\right]$. *Given that all rewards are bounded* $|r| \leq R_{\max}$*, the error of an approximate model's policy evaluation estimate* $\rho_{\widehat{M}}(\pi)$ *is bounded by:*

$$|\rho_M(\pi) - \rho_{\widehat{M}}(\pi)| \leq c_1\left(\epsilon_M^\pi + \epsilon_{\widehat{M}}^\pi\right) + c_2\epsilon_M^P \tag{21}$$

where $c_1 = \frac{2R_{\max}}{(1-\gamma)^2}$ *and* $c_2 = \frac{2\gamma R_{\max}}{(1-\gamma)^2}$.

Proof. By the triangular inequality,

$$\left|\rho_M(\pi) - \rho_{\widehat{M}}(\pi)\right| \leq |\rho_M(\pi) - \rho_M(\mu)| + \left|\rho_M(\mu) - \rho_{\widehat{M}}(\mu)\right| + \left|\rho_{\widehat{M}}(\mu) - \rho_{\widehat{M}}(\pi)\right|.$$

By applying Lemma A.1 for second term, and applying Lemma A.2 for first and third term, we recover the result. □

Corollary 4.1. *The negative baseline regret (Petrik et al., 2016) of* π_δ^**, which is the performance improvement by adopting* π_δ^* *instead of the baseline policy* μ *on the true environment* M*, is lower bounded by:*

$$\rho_M(\pi_\delta^*) - \rho_M(\mu) \geq \rho_{\widehat{M}}(\pi_\delta^*) - \rho_{\widehat{M}}(\mu) - c_1\sqrt{2\delta} - 2c_2\epsilon_M^P$$

with c_1 *and* c_2 *defined in Theorem 4.1.*

Proof. According to the constraints of Eq. (2), Pinsker's inequality and Jensen's inequality, we have

$$\begin{aligned}\mathop{\mathbb{E}}_{s\sim d_M^{\pi_\delta^*}}\left[\mathbb{TV}_s^{\pi_\delta^*,\mu}\right] &\leq \mathop{\mathbb{E}}_{s\sim d_M^{\pi_\delta^*}}\left[\sqrt{\frac{1}{2}\mathbb{KL}_s^{\pi_\delta^*,\mu}}\right] \leq \sqrt{\frac{1}{2}\mathop{\mathbb{E}}_{s\sim d_M^{\pi_\delta^*}}\left[\mathbb{KL}_s^{\pi_\delta^*,\mu}\right]} \leq \sqrt{\frac{\delta}{2}}, \\ \mathop{\mathbb{E}}_{s\sim d_{\widehat{M}}^{\pi_\delta^*}}\left[\mathbb{TV}_s^{\pi_\delta^*,\mu}\right] &\leq \mathop{\mathbb{E}}_{s\sim d_{\widehat{M}}^{\pi_\delta^*}}\left[\sqrt{\frac{1}{2}\mathbb{KL}_s^{\pi_\delta^*,\mu}}\right] \leq \sqrt{\frac{1}{2}\mathop{\mathbb{E}}_{s\sim d_{\widehat{M}}^{\pi_\delta^*}}\left[\mathbb{KL}_s^{\pi_\delta^*,\mu}\right]} \leq \sqrt{\frac{\delta}{2}}.\end{aligned}$$

We substitute them into Theorem 4.1 to get:

$$\rho_M(\pi_\delta^*) - \rho_{\widehat{M}}(\pi_\delta^*) \geq -c_1\sqrt{2\delta} - c_2\epsilon_M^P.$$

Similarly, $\rho_M(\mu) - \rho_{\widehat{M}}(\mu) \geq -c_2\epsilon_M^P$ holds as ϵ_M^π and $\epsilon_{\widehat{M}}^\pi$ are zero in this case. Adding two equations completes the proof. □

B. Proofs for KL-regularized Policy Iteration

B.1. Proof of Lemma 4.1

Lemma 4.1. *For a given π with $\mathbb{KL}_s^{\pi,\mu} < \infty \; \forall s$, the backup operator $\mathcal{T}_{\mathbb{KL}}^{\pi}$ is a contraction mapping and has an unique fixed point solution $\mathcal{T}_{\mathbb{KL}}^{\pi}\widetilde{Q}^{\pi} = \widetilde{Q}^{\pi}$. In other words, for any $\widetilde{Q}^0 : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, define $\widetilde{Q}^{k+1} = \mathcal{T}_{\mathbb{KL}}^{\pi}\widetilde{Q}^{k}$. Then the sequence $\widetilde{Q}^k$ converges to KL-regularized Q-value function of π as $k \to \infty$.*

Proof. We first show the *monotonicity* of $\mathcal{T}_{\mathbb{KL}}^{\pi}$. Note that if $\widetilde{Q}(s,a) \le \widetilde{Q}'(s,a)$ for $s \in \mathcal{S}, a \in \mathcal{A}$,

$$\begin{aligned}\left(\mathcal{T}_{\mathbb{KL}}^{\pi}\widetilde{Q}^{\pi}\right)(s,a) &= R(s,a) - \mathbb{E}_{s'\sim p(s'|s,a)}\left[\alpha(s')\mathbb{KL}_s^{\pi,\mu}\right] + \mathbb{E}_{\substack{s'\sim p(s'|s,a)\\ a'\sim\pi(a'|s')}}\left[\widetilde{Q}(s',a')\right]\\ &\le R(s,a) - \mathbb{E}_{s'\sim p(s'|s,a)}\left[\alpha(s')\mathbb{KL}_s^{\pi,\mu}\right] + \mathbb{E}_{\substack{s'\sim p(s'|s,a)\\ a'\sim\pi(a'|s')}}\left[\widetilde{Q}'(s',a')\right] = \left(\mathcal{T}_{\mathbb{KL}}^{\pi}\widetilde{Q}^{\pi}\right)(s,a)\end{aligned}$$

holds for $s \in \mathcal{S}, a \in \mathcal{A}$.

The *additivity* of $\mathcal{T}_{\mathbb{KL}}^{\pi}$ can also be proved. For $d \in \mathbb{R}$ and $\mathbb{1} \in \mathbb{R}^{\mathcal{S}}$ (the vector of all ones),

$$\begin{aligned}\left(\mathcal{T}_{\mathbb{KL}}^{\pi}(\widetilde{Q}^{\pi} + d\mathbb{1})\right)(s,a) &= R(s,a) - \mathbb{E}_{s'\sim p(s'|s,a)}\left[\alpha(s')\mathbb{KL}_s^{\pi,\mu}\right] + \mathbb{E}_{\substack{s'\sim p(s'|s,a)\\ a'\sim\pi(a'|s')}}\left[\widetilde{Q}(s',a') + d\right]\\ &= R(s,a) - \mathbb{E}_{s'\sim p(s'|s,a)}\left[\alpha(s')\mathbb{KL}_s^{\pi,\mu}\right] + \mathbb{E}_{\substack{s'\sim p(s'|s,a)\\ a'\sim\pi(a'|s')}}\left[\widetilde{Q}(s',a')\right] + \gamma d\\ &= \left(\mathcal{T}_{\mathbb{KL}}^{\pi}\widetilde{Q}^{\pi}\right)(s,a) + \gamma d\end{aligned}$$

holds for $s \in \mathcal{S}, a \in \mathcal{A}$.

Finally, let $d = \|\widetilde{Q}(s,a) - \widetilde{Q}'(s,a)\|_\infty$. Then, for $s \in \mathcal{S}, a \in \mathcal{A}$,

$$\begin{aligned}&\widetilde{Q}(s,a) - d \le \widetilde{Q}'(s,a) \le \widetilde{Q}(s,a) + d\\ \Longrightarrow\; &\left(\mathcal{T}_{\mathbb{KL}}^{\pi}(\widetilde{Q} - d\mathbb{1})\right)(s,a) \le \left(\mathcal{T}_{\mathbb{KL}}^{\pi}\widetilde{Q}'\right)(s,a) \le \left(\mathcal{T}_{\mathbb{KL}}^{\pi}(\widetilde{Q} + d\mathbb{1})\right)(s,a) \quad \text{(monotonicity)}\\ \Longrightarrow\; &\left(\mathcal{T}_{\mathbb{KL}}^{\pi}\widetilde{Q}\right)(s,a) - \gamma d \le \left(\mathcal{T}_{\mathbb{KL}}^{\pi}\widetilde{Q}'\right)(s,a) \le \left(\mathcal{T}_{\mathbb{KL}}^{\pi}\widetilde{Q}\right)(s,a) + \gamma d \quad \text{(additivity)}\\ \Longrightarrow\; &\left\|\left(\mathcal{T}_{\mathbb{KL}}^{\pi}\widetilde{Q}\right)(s,a) - \left(\mathcal{T}_{\mathbb{KL}}^{\pi}\widetilde{Q}'\right)(s,a)\right\|_\infty \le \gamma\|\widetilde{Q}(s,a) - \widetilde{Q}'(s,a)\|_\infty.\end{aligned}$$

Therefore, $\mathcal{T}_{\mathbb{KL}}^{\pi}$ is a γ-contraction. By Banach fixed point theorem, it admits a unique fixed point $\mathcal{T}_{\mathbb{KL}}^{\pi}\widetilde{Q}^{\pi} = \widetilde{Q}^{\pi}$. □

Note that, if we deal with the two non-degenerate, normally distributed policies π and μ, then $\mathbb{KL}_s^{\pi,\mu} < \infty$ always holds.

B.2. Proof of Lemma 4.2

Lemma 4.2. *Given a policy $\pi \in \Pi$ and its value function $\widetilde{Q}^{\pi}$, if we update the new policy π^{new} by $\pi^{\text{new}}(\cdot|s) = \arg\min_{\pi'\in\Pi}\mathbb{KL}\left(\pi'(\cdot|s)\middle\|\frac{1}{Z^{\pi}(s)}\exp\left(\frac{\widetilde{Q}^{\pi}(s,\cdot)}{\alpha(s)}\right)\mu(\cdot|s)\right)$ where $Z^{\pi}(s)$ is the normalization constant, then $\widetilde{Q}^{\pi^{\text{new}}}(s,a) \ge \widetilde{Q}^{\pi}(s,a)\;\forall s,a$.*

Proof. Note that $\mathbb{KL}\left(\pi^{\text{new}}(\cdot|s)\middle\|\frac{1}{Z^{\pi}(s)}\exp\left(\frac{\widetilde{Q}^{\pi}(s,\cdot)}{\alpha(s)}\right)\mu(\cdot|s)\right) \le \mathbb{KL}\left(\pi(\cdot|s)\middle\|\frac{1}{Z^{\pi}(s)}\exp\left(\frac{\widetilde{Q}^{\pi}(s,\cdot)}{\alpha(s)}\right)\mu(\cdot|s)\right)$ must hold as we can always choose $\pi^{\text{new}} = \pi \in \Pi$. Then,

$$\begin{aligned}
&\mathbb{E}_{a\sim\pi^{\text{new}}(\cdot|s)}\left[\log \pi^{\text{new}}(a|s) - \frac{\widetilde{Q}^{\pi}(s,a)}{\alpha(s)} - \log \mu(a|s) + \log Z^{\pi}(s)\right] \\
&\qquad\qquad \le \mathbb{E}_{a\sim\pi(\cdot|s)}\left[\log \pi(a|s) - \frac{\widetilde{Q}^{\pi}(s,a)}{\alpha(s)} - \log \mu(a|s) + \log Z^{\pi}(s)\right] \\
\implies\ & \alpha(s)\mathbb{KL}_s^{\pi^{\text{new}},\mu} - \mathbb{E}_{a\sim\pi^{\text{new}}(\cdot|s)}\left[\widetilde{Q}^{\pi}(s,a)\right] \le \alpha(s)\mathbb{KL}_s^{\pi,\mu} - \mathbb{E}_{a\sim\pi(\cdot|s)}\left[\widetilde{Q}^{\pi}(s,a)\right]
\end{aligned}$$

Then, we can show that

$$\begin{aligned}
\widetilde{Q}^{\pi}(s_0,a_0) &= R(s_0,a_0) + \gamma\mathbb{E}_{s_1}\left[\mathbb{E}_{a_1\sim\pi}\left[Q^{\pi}(s_1,a_1)\right] - \alpha(s_1)\mathbb{KL}_{s_1}^{\pi,\mu}\right] \\
&\le R(s_0,a_0) + \gamma\mathbb{E}_{s_1}\left[\mathbb{E}_{a_1\sim\pi^{\text{new}}}\left[Q^{\pi}(s_1,a_1)\right] - \alpha(s_1)\mathbb{KL}_{s_1}^{\pi^{\text{new}},\mu}\right] \\
&\qquad\vdots \\
&\le R(s_0,a_0) + \mathbb{E}_{\pi^{\text{new}}}\left[\sum_{t=1}^{\infty}\gamma^t\Big(R(s_t,a_t) - \alpha(s_t)\mathbb{KL}_{s_t}^{\pi^{\text{new}},\mu}\right] \\
&= \widetilde{Q}^{\pi^{\text{new}}}(s_0,a_0)
\end{aligned}$$

□

Theorem 4.2. (KL-Regularized Policy Iteration) *Suppose that* $|R(s,a)| \le R_{\max}$ *and* $\mathbb{KL}_s^{\pi,\mu} < \infty$. *Starting from any* $\pi_0 \in \Pi$, *the sequence of the value functions* $\widetilde{Q}^{\pi_k}$ *and the improved policies* π_{k+1} *converge to the optimal value function and the optimal policy* $\pi^* \in \Pi$, *i.e.* $\lim_{k\to\infty}\widetilde{Q}^{\pi_k}(s,a) \ge \widetilde{Q}^{\pi}(s,a)$ *for any* $\pi \in \Pi$, $s \in \mathcal{S}$, *and* $a \in \mathcal{A}$.

Proof. By Lemma 4.2, the sequence of the value functions $\widetilde{Q}^{\pi_k}$ is monotonically increasing. Since the reward and KL are bounded, $\widetilde{Q}^{\pi}$ is bounded, and the sequence converges to some π^*. At convergence,

$$\mathbb{KL}\left(\pi^*(\cdot|s)\middle\|\frac{1}{Z^{\pi}(s)}\exp\left(\frac{\widetilde{Q}^{\pi}(s,\cdot)}{\alpha(s)}\right)\mu(\cdot|s)\right) < \mathbb{KL}\left(\pi(\cdot|s)\middle\|\frac{1}{Z^{\pi}(s)}\exp\left(\frac{\widetilde{Q}^{\pi}(s,\cdot)}{\alpha(s)}\right)\mu(\cdot|s)\right)$$

should hold for all $\pi \in \Pi, \pi \neq \pi^*$. Using the argument as in Lemma 4.2, we get $\widetilde{Q}^{\pi^*}(s,a) > \widetilde{Q}^{\pi}(s,a)$ for all $s \in \mathcal{S}, a \in \mathcal{A}$. Therefore, π^* is optimal in Π. □

C. Proof of Theorem 5.1

Throughout this section, suppose $\alpha_\xi(s)$ is parameterized by ξ. Let $\pi_\xi^* = \arg\max_{\pi, \widehat{M}_{\text{train}}} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t - \alpha_\xi(s_t)\mathbb{KL}_{s_t}^{\pi,\mu}\right]$ be the optimal policy of the generalized KL-regularized MDP $\widehat{M}_{\text{train}}$ with a fixed yet arbitrary ξ.

Lemma C.1. (KL-Regularized Bellman Optimality Equation (Fox et al., 2016)) *KL-regularized optimal value function of* $\widehat{M}_{\text{train}}$ *is defined recursively as:*

$$\widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s,a) = R(s,a) + \gamma \mathbb{E}_{s'}\left[\widetilde{V}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s')\right]$$
$$\widetilde{V}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s) = \alpha_\xi(s) \log \mathbb{E}_{a\sim\mu(\cdot|s)}\left[\exp\left(\frac{\widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s,a)}{\alpha_\xi(s)}\right)\right]$$

Proof. Note that in the tabular case, the optimal policy and its value function $\widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}$ should satisfy the following relationship by Lemma 4.1 and Lemma 4.2:

$$\pi_\xi^*(a|s) = \frac{\exp\left(\frac{\widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s,a)}{\alpha_\xi(s)}\right)\mu(a|s)}{\sum_{a'} \exp\left(\frac{\widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s,a')}{\alpha_\xi(s)}\right)\mu(a'|s)}$$

By rearranging this, we obtain:

$$\log \frac{\pi_\xi^*(a|s)}{\mu(a|s)} = \frac{Q^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s,a)}{\alpha_\xi(s)} - \log\left(\sum_{a'} \exp\left(\frac{\widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s,a')}{\alpha_\xi(s)}\right)\mu(a'|s)\right)$$

By taking expectation on both sides,

$$\mathbb{E}_{a\sim\pi_\xi^*(\cdot|s)}\left[\log \frac{\pi_\xi^*(a|s)}{\mu(a|s)}\right] = \frac{\mathbb{E}_{a\sim\pi_\xi^*(\cdot|s)}\left[\widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s,a)\right]}{\alpha_\xi(s)} - \log\left(\sum_{a'} \exp\left(\frac{\widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s,a')}{\alpha_\xi(s)}\right)\mu(a'|s)\right)$$
$$\therefore \widetilde{V}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s) = -\alpha_\xi(s)\mathbb{KL}\Big(\pi_\xi^*(\cdot|s)||\mu(\cdot|s)\Big) + \mathbb{E}_{a\sim\pi_\xi^*(\cdot|s)}\left[\widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s,a)\right] = \log\left(\sum_{a'} \exp\left(\frac{\widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s,a')}{\alpha_\xi(s)}\right)\mu(a'|s)\right)$$

□

We remark that Lemma C.1 implies:

$$\pi_\xi^*(a|s) = \exp\left(\frac{\widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s,a) - \widetilde{V}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s)}{\alpha_\xi(s)}\right)\mu(a|s) \tag{22}$$

Lemma C.2. *Let* $\widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s,a)$ *be the optimal KL-regularized value function of* $\widehat{M}_{\text{train}}$. *Then, the following holds:*

$$\nabla_\xi \widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s,a) = \mathbb{E}_{\pi_\xi^*, \widehat{M}_{\text{train}}}\left[\sum_{t=1}^{\infty} \gamma^t \Big(-\mathbb{KL}\Big(\pi_\xi^*(\cdot|s_t)||\mu(\cdot|s_t)\Big)\nabla_\xi \alpha_\xi(s_t)\Big) \mid s_0 = s, a_0 = a\right]$$
$$\nabla_\xi \widetilde{V}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s) = \mathbb{E}_{\pi_\xi^*, \widehat{M}_{\text{train}}}\left[\sum_{t=0}^{\infty} \gamma^t \Big(-\mathbb{KL}\Big(\pi_\xi^*(\cdot|s_t)||\mu(\cdot|s_t)\Big)\nabla_\xi \alpha_\xi(s_t)\Big) \mid s_0 = s\right]$$

Proof. We use the recursive form of the KL-regularized Bellman optimality equation by Lemma C.1:

$$
\begin{aligned}
&\nabla_\xi \widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s,a)\\
&= \nabla_\xi \left(R(s,a) + \gamma \mathbb{E}_{s'} \left[\alpha_\xi(s') \log \mathbb{E}_{a' \sim \mu(\cdot|s')} \left[\exp\left(\frac{\widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s',a')}{\alpha_\xi(s')} \right) \right] \right] \right)\\
&= \gamma \mathbb{E}_{s'} \left[\nabla_\xi \alpha_\xi(s') \log \mathbb{E}_{a' \sim \mu(\cdot|s')} \left[\exp\left(\frac{\widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s',a')}{\alpha_\xi(s')} \right) \right] + \alpha_\xi(s') \frac{\mathbb{E}_{a' \sim \mu(\cdot|s')} \left[\exp\left(\frac{\widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s',a')}{\alpha_\xi(s')} \right) \nabla_\xi \left(\frac{\widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s',a')}{\alpha_\xi(s')} \right) \right]}{\mathbb{E}_{a' \sim \mu(\cdot|s')} \left[\exp\left(\frac{\widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s',a')}{\alpha_\xi(s')} \right) \right]} \right]\\
&= \gamma \mathbb{E}_{s'} \left[\frac{\nabla_\xi \alpha_\xi(s')}{\alpha_\xi(s')} \widetilde{V}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s') + \alpha_\xi(s') \mathbb{E}_{a' \sim \pi_\xi^*(\cdot|s')} \left[\nabla_\xi \left(\frac{\widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s',a')}{\alpha_\xi(s')} \right) \right] \right]\\
&= \gamma \mathbb{E}_{s'} \left[\frac{\nabla_\xi \alpha_\xi(s') \widetilde{V}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s')}{\alpha_\xi(s')} + \alpha_\xi(s') \mathbb{E}_{a' \sim \pi_\xi^*(\cdot|s')} \left[\frac{\nabla_\xi \widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s',a') \alpha_\xi(s') - \widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s',a') \nabla_\xi \alpha_\xi(s')}{\alpha_\xi(s')^2} \right] \right]\\
&= \gamma \mathbb{E}_{s', a' \sim \pi_\xi^*(\cdot|s')} \left[\frac{\nabla_\xi \alpha_\xi(s') \widetilde{V}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s') + \nabla_\xi \widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s',a') \alpha_\xi(s') - \widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s',a') \nabla_\xi \alpha_\xi(s')}{\alpha_\xi(s')} \right]\\
&= \gamma \mathbb{E}_{s', a' \sim \pi_\xi^*(\cdot|s')} \left[\frac{\widetilde{V}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s') - \widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s',a')}{\alpha_\xi(s')} \nabla_\xi \alpha_\xi(s') + \nabla_\xi \widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s',a') \right]\\
&= \gamma \mathbb{E}_{s', a' \sim \pi_\xi^*(\cdot|s')} \left[-\mathbb{KL}\Big(\pi_\xi^*(\cdot|s_t) || \mu(\cdot|s_t)\Big) \nabla_\xi \alpha_\xi(s') + \nabla_\xi \widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s',a') \right]
\end{aligned}
$$

Finally, we obtain the result by repeatedly expanding the recursion. We can obtain the result for $\nabla_\xi \widetilde{V}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s)$ in a similar manner. □

The result of Lemma C.2 is natural. As α is increased, the KL-regularized return in train MDP $\widehat{M}_{\text{train}}$ is penalized more.

Lemma C.3. *The following inequality holds:*

$$
\frac{\nabla_\xi \pi_\xi^*(a|s)}{\pi_\xi^*(a|s)} = \frac{\Big(\nabla_\xi \widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s,a) - \nabla_\xi \widetilde{V}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s)\Big) \alpha_\xi(s) - \Big(\widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s,a) - \widetilde{V}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s)\Big) \nabla_\xi \alpha_\xi(s)}{\alpha_\xi(s)^2}
$$

Proof. We start from Eq. (22).

$$
\begin{aligned}
\nabla_\xi \pi_\xi^*(a|s) &= \nabla_\xi \left[\exp\left(\frac{\widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s,a) - \widetilde{V}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s)}{\alpha_\xi(s)} \right) \mu(a|s) \right]\\
&= \pi_\xi^*(a|s) \left[\frac{\nabla_\xi \Big(\widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s,a) - \widetilde{V}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s)\Big) \alpha_\xi(s) - \Big(\widetilde{Q}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s,a) - \widetilde{V}^{\pi_\xi^*}_{\widehat{M}_{\text{train}}}(s)\Big) \nabla_\xi \alpha_\xi(s)}{\alpha_\xi(s)^2} \right]
\end{aligned}
$$

□

Now we are ready to provide the proof of the main theorem.

Theorem 5.1. *Suppose that the state-dependent function $\alpha_\xi(s)$ for Eq. (8) is parameterized by ξ. Then, the hypergradient $\nabla_\xi \rho_{\widehat{M}_{\text{valid}}}(\pi_\xi^*) = \nabla_\xi \mathbb{E}_{\pi_\xi^*, \widehat{M}_{\text{valid}}} \left[\sum_{t=0}^\infty \gamma^t r_t \right]$ is given by:*

$$
\nabla_\xi \rho_{\widehat{M}_{\text{valid}}}(\pi_\xi^*) = \mathbb{E}_{\pi_\xi^*, \widehat{M}_{\text{valid}}} \left[\sum_{t=0}^{\infty} \gamma^t \boldsymbol{\chi}_\xi(s_t) \right]
$$

$$
\begin{aligned}
\textit{s.t.} \qquad \pi^*_\xi &\triangleq \arg\max_\pi \mathbb{E}_{\pi, \widehat{M}_{\text{train}}} \left[\sum_{t=0}^{\infty} \gamma^t \Big(r_t - \alpha_\xi(s_t) \mathbb{KL}^{\pi,\mu}_{s_t} \Big) \right] \\
\nabla_\xi \widetilde{Q}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s,a) &\triangleq \mathbb{E}_{\pi^*_\xi, \widehat{M}_{\text{train}}} \left[\sum_{t=1}^{\infty} \gamma^t \Big(-\nabla_\xi \alpha_\xi(s_t) \mathbb{KL}^{\pi^*_\xi,\mu}_{s_t} \Big) \right] \\
\widetilde{Q}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s,a) &\triangleq \mathbb{E}_{\pi^*_\xi, \widehat{M}_{\text{train}}} \left[\sum_{t=0}^{\infty} \gamma^t \Big(r_t - \alpha_\xi(s_t) \mathbb{KL}^{\pi,\mu}_{s_t} \Big) \right] \\
Q^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s,a) &\triangleq \mathbb{E}_{\pi^*_\xi, \widehat{M}_{\text{valid}}} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] \\
\boldsymbol{\chi}_\xi(s) &\triangleq \frac{1}{\alpha_\xi(s)^2} \text{cov}_{a \sim \pi^*_\xi} \left[\alpha_\xi(s) \nabla_\xi \widetilde{Q}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s,a) - \nabla_\xi \alpha_\xi(s) \widetilde{Q}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s,a), Q^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s,a) \right]
\end{aligned}
$$

where $\text{cov}_a[\mathbf{f}(a), g(a)]_i = \mathbb{E}[f_i(a)g(a)] - \mathbb{E}[f_i(a)]\mathbb{E}[g(a)]$ *denotes element-wise covariance between the vector* $\mathbf{f}(\cdot)$ *and the scalar* $g(\cdot)$.

Proof. By Lemma C.3,

$$
\begin{aligned}
&\mathbb{E}_{a \sim \pi^*_\xi} \left[\frac{\nabla_\xi \pi^*_\xi(a|s)}{\pi^*_\xi(a|s)} Q^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s,a) \right] \\
&= \mathbb{E}_{a \sim \pi^*_\xi} \left[\frac{\Big(\nabla_\xi \widetilde{Q}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s,a) - \nabla_\xi \widetilde{V}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s) \Big) \alpha_\xi(s) - \Big(\widetilde{Q}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s,a) - \widetilde{V}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s) \Big) \nabla_\xi \alpha_\xi(s)}{\alpha_\xi(s)^2} Q^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s,a) \right] \\
&= \frac{\mathbb{E}_{a \sim \pi^*_\xi} \left[\nabla_\xi \widetilde{Q}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s,a) Q^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s,a) \right] - \nabla_\xi \widetilde{V}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s) \mathbb{E}_{a \sim \pi^*_\xi} \left[Q^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s,a) \right]}{\alpha_\xi(s)} \\
&\quad - \frac{\mathbb{E}_{a \sim \pi^*_\xi} \left[\widetilde{Q}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s,a) Q^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s,a) \right] - \widetilde{V}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s) \mathbb{E}_{a \sim \pi^*_\xi} \left[Q^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s,a) \right]}{\alpha_\xi(s)^2} \nabla_\xi \alpha_\xi(s) \\
&= \frac{\mathbb{E}_{a \sim \pi^*_\xi} \left[\nabla_\xi \widetilde{Q}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s,a) Q^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s,a) \right] - \mathbb{E}_{a \sim \pi^*_\xi} \left[-\nabla_\xi \alpha_\xi(s) \mathbb{KL}\Big(\pi^*_\xi(\cdot|s) || \mu(\cdot|s)\Big) + \nabla_\xi \widetilde{Q}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s,a) \right] \mathbb{E}_{a \sim \pi^*_\xi} \left[Q^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s,a) \right]}{\alpha_\xi(s)} \\
&\quad - \frac{\mathbb{E}_{a \sim \pi^*_\xi} \left[\widetilde{Q}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s,a) Q^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s,a) \right] - \mathbb{E}_{a \sim \pi^*_\xi} \left[-\alpha_\xi(s) \mathbb{KL}\Big(\pi^*_\xi(\cdot|s) || \mu(\cdot|s)\Big) + \widetilde{Q}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s,a) \right] \mathbb{E}_{a \sim \pi^*_\xi} \left[Q^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s,a) \right]}{\alpha_\xi(s)^2} \nabla_\xi \alpha_\xi(s) \\
&= \frac{\mathbb{E}_{a \sim \pi^*_\xi} \left[\nabla_\xi \widetilde{Q}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s,a) Q^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s,a) \right] - \mathbb{E}_{a \sim \pi^*_\xi} \left[\nabla_\xi \widetilde{Q}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s,a) \right] \mathbb{E}_{a \sim \pi^*_\xi} \left[Q^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s,a) \right] + \nabla_\xi \alpha_\xi(s) \mathbb{KL}\Big(\pi^*_\xi(\cdot|s) || \mu(\cdot|s)\Big)}{\alpha_\xi(s)} \\
&\quad - \frac{\mathbb{E}_{a \sim \pi^*_\xi} \left[\widetilde{Q}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s,a) Q^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s,a) \right] - \mathbb{E}_{a \sim \pi^*_\xi} \left[\widetilde{Q}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s,a) \right] \mathbb{E}_{a \sim \pi^*_\xi} \left[Q^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s,a) \right] + \alpha_\xi(s) \mathbb{KL}\Big(\pi^*_\xi(\cdot|s) || \mu(\cdot|s)\Big)}{\alpha_\xi(s)^2} \nabla_\xi \alpha_\xi(s) \\
&= \frac{\text{cov}_{a \sim \pi^*_\xi} \left[\nabla_\xi \widetilde{Q}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s,a), Q^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s,a) \right]}{\alpha_\xi(s)} - \frac{\text{cov}_{a \sim \pi^*_\xi} \left[\widetilde{Q}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s,a), Q^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s,a) \right]}{\alpha_\xi(s)^2} \nabla_\xi \alpha_\xi(s) \\
&= \frac{\text{cov}_{a \sim \pi^*_\xi} \left[\alpha_\xi(s) \nabla_\xi \widetilde{Q}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s,a) - \nabla_\xi \alpha_\xi(s) \widetilde{Q}^{\pi^*_\xi}_{\widehat{M}_{\text{train}}}(s,a), Q^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s,a) \right]}{\alpha_\xi(s)^2} \triangleq \boldsymbol{\chi}_\xi(s) \qquad (23)
\end{aligned}
$$

Then, we use the recursive form of the standard Bellman equation for valid MDP $\widehat{M}_{\text{valid}}$.

$$
\begin{aligned}
\nabla_\xi V^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s) &= \nabla_\xi \sum_a \pi^*_\xi(a|s)\left[R(s,a)+\gamma\mathbb{E}_{s'}\left[V^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s')\right]\right] \\
&= \sum_a \nabla_\xi \pi^*_\xi(a|s)\left[R(s,a)+\gamma\mathbb{E}_{s'}\left[V^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s')\right]\right] + \sum_a \pi^*_\xi(a|s)\left[\gamma\mathbb{E}_{s'}\left[\nabla_\xi V^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s')\right]\right] \\
&= \mathbb{E}_{a\sim\pi^*_\xi}\left[\frac{\nabla_\xi \pi^*_\xi(a|s)}{\pi^*_\xi(a|s)} Q^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s,a) + \gamma\mathbb{E}_{s'}\left[\nabla_\xi V^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s')\right]\right] \\
&= \boldsymbol{\chi}_\xi(s) + \gamma\mathbb{E}_{a\sim\pi^*_\xi, s'}\left[\nabla_\xi V^{\pi^*_\xi}_{\widehat{M}_{\text{valid}}}(s')\right]
\end{aligned}
$$

Finally, by repeatedly expanding the recursion, we obtain the result in the theorem. □

D. Illustrative Example: Diverse Mixture of Policies

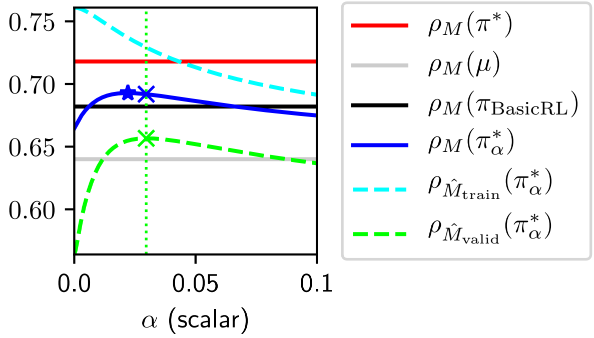


Figure 5. Experimental result on a random MDP using diverse mixture of policies, where $\mathcal{S} = \{s^1, \dots s^{20}\}$, $|\mathcal{A}| = 4$, and α controls the degree of regularization. The result is obtained by averaging over 300 trials.

We also conducted experiments on a diverse mixture of policies in the illustrative example in Section 5.1, where the trajectory is collected by diverse policies of $\mu = \zeta\pi^* + (1-\zeta)\pi_{\text{unif}}$ such that $\zeta \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$, instead of a single fixed policy $(\zeta = 0.7)$. We collected batch data $\mathcal{D}$ consisting of 100 episodes with a maximum time step 50, i.e. 20 episodes for each ζ. The rest of the experimental settings are identical to those described in Section 5.1. Although the diversity of trajectories has increased, the overall trend remains the same, as presented in Figure 5.

E. AC-BOPAH Implementation Details

In this section, we describe the technical details of AC-BOPAH we omitted in the main text due to the space limit.

E.1. Conservative Training with Bootstrapped Q

Using ensembles of Q functions to mitigate the overestimation bias when obtaining maximum Q value is a well-known strategy in deep reinforcement learning (Hasselt, 2010; Haarnoja et al., 2018). When training with a fixed batch of experiences, this bias becomes an important issue since the overestimation of Q values for unseen actions accumulates over updates and leads to a potential divergence (Kumar et al., 2019).

To penalize Q values on unseen actions, we choose to estimate the uncertainty by bootstrapping (Osband et al., 2016), i.e. learning k independent Q values $\{Q_{\theta_1}, ..., Q_{\theta_K}\}$ with their own initialization and training set that each element is sampled with replacement from original training set. The conservative Q estimate then can be computed based on bootstrap lower confidence bound (LCB),

$$\underset{k=1..K}{\text{LCB}}[Q_{\theta_k}] = \frac{1}{K}\sum_{k=1}^{K} Q_{\theta_k} - \lambda_c \sqrt{\text{var}_k[Q_{\theta_k}]}, \tag{24}$$

and be used as the target of value function learning. While this plays the similar role as the ensembled Q trained with same dataset in (Fujimoto et al., 2019), we found that, in practice, using the bootstrap resampled datasets for each Q values was crucial in estimating accurate uncertainty. Using the conservative estimate also helped the algorithm to compute hypergradients more reliably by making the train critic and the valid critic more different value predictions. In our experiments, we used $K = 5$ and $\lambda_c = 1$ for all experiments.

E.2. Gradient Penalty of Critic Network

The connections between actor-critic methods and generative adversarial networks (GANs) have been discussed (Pfau & Vinyals, 2016) in that the actor (generator) network is updated using the gradient signal from the critic (discriminator) network, and both of the models suffer from instability of training. For training GANs, a gradient penalty to the discriminator has been widely adopted in order to stabilize the overall training dynamics (Gulrajani et al., 2017; Kodali et al., 2017; Fedus* et al., 2018). Similarly, we introduced the gradient penalty to the critic network, which was helpful in stabilizing the off-policy learning of actor-critic. We finally use the following penalty (Kodali et al., 2017):

$$\begin{aligned}\mathcal{R}(\theta) &= \mathbb{E}_{s\sim D, a\sim\pi}\left[\max\left(0, \|\nabla_{s,a} Q_\theta(s,a)\| - \kappa\right)^2\right] \\ \mathcal{R}(\psi) &= \mathbb{E}_{s\sim D, a\sim\pi}\left[\max\left(0, \|\nabla_{s} V_\psi(s)\| - \kappa\right)^2\right]\end{aligned} \tag{25}$$

which prevents the magnitude of the critic's gradient from growing too large, thus making the actor's learning signal be more stable. Finally, we optimize the parameters of the critics with Eq. (25), using the following regularized objectives: for each $\theta \in \{\theta_{\text{train}}, \theta^{\mathbb{KL}}_{\text{train}}, \theta_{\text{valid}}\}$ and $\psi \in \{\psi_{\text{train}}, \psi^{\mathbb{KL}}_{\text{train}}, \psi_{\text{valid}}\}$,

$$\begin{aligned}J^\dagger(\theta) &\triangleq J(\theta) + \lambda_g \mathcal{R}(\theta) \\ J^\dagger(\psi) &\triangleq J(\psi) + \lambda_g \mathcal{R}(\psi)\end{aligned} \tag{26}$$

E.3. Design of the Feature Function $\phi(s)$

We subsampled 10^4 states from the training data $\mathcal{D}_{\text{train}}$, and recovered $d_\xi - 1$ representative points by building a cover tree (Beygelzimer et al., 2006)[3]. The remaining one dimension is reserved for the rest of state space, which is not represented by any of the representative points. After fixing the representative points $\{s_i^r\}_{i=1}^{d_\xi-1}$, we defined $\phi(s) = [\phi_1(s), ..., \phi_{d_\xi}(s)]^\top$ as:

$$u_i(s) = \begin{cases} \exp\left(-\frac{\|s-s_i^r\|_2^2}{2\sigma^2}\right) & \text{for } i = 1, ..., d_\xi - 1 \\ \max\left(1 - \sum_{j=1}^{d_\xi-1} u_j(s), 0\right) & \text{for } i = d_\xi \end{cases}$$

$$\phi_i = \frac{u_i}{\sum_{j=1}^{d_\xi} u_j(s)}$$

[3] We used the implementation in https://github.com/patvarilly/CoverTree

such that $\phi_i(s) > 0\ \ \forall i$ and $\sum_{i=1}^{d_\xi} \phi_i(s) = 1$ are guaranteed. We used the median heuristic to find appropriate $\sigma = \sqrt{H_n/8}$, where $H_n = \text{Med}\{\|s_i^r - s_j^r\|^2 \ :\ 1 \le i < j \le d_\xi - 1\}$, where Med is the empirical median.

E.4. Pseudo-code of AC-BOPAH

Algorithm 1 Actor-Critic BOPAH

```
**Input:** Training data $\mathcal{D}_{\text{train}}$, validation data $\mathcal{D}_{\text{valid}}$
Build a cover tree on $\mathcal{D}_{\text{train}}$ to construct the RBF feature functions $\phi(s)$
Initialize Q-function parameters $\{\theta_{\text{train},k}, \theta_{\text{valid},k}, \theta^{\mathbb{KL}}_{\text{train},k}\}_{k=1}^{K}$
Initialize V-function parameters $\psi_{\text{train}}, \psi_{\text{valid}}, \psi^{\mathbb{KL}}_{\text{train}}$
Initialize target network parameters $\bar{\psi}_{\text{train}} \leftarrow \psi_{\text{train}}, \bar{\psi}_{\text{valid}} \leftarrow \psi_{\text{valid}}, \bar{\psi}^{\mathbb{KL}}_{\text{train}} \leftarrow \psi^{\mathbb{KL}}_{\text{train}}$
Initialize policy parameters $\omega_b$, $\omega$
Initialize hyperparameters $\xi$
for $k = 1, \ldots, K$ do
    Sample bootstrapped update weights $w_{\text{train},k} \sim \text{Mult}\big(|\mathcal{D}_{\text{train}}|, \frac{1}{|\mathcal{D}_{\text{train}}|}\mathbf{1}\big)$
    Sample bootstrapped update weights $w_{\text{valid},k} \sim \text{Mult}\big(|\mathcal{D}_{\text{valid}}|, \frac{1}{|\mathcal{D}_{\text{valid}}|}\mathbf{1}\big)$
end for
for each iteration do
    Sample mini-batches $\{\mathcal{B}_{\text{train}}, \mathcal{B}_{\text{valid}}\}$ from $\{\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{valid}}\}$
    # Update Q-function parameters
    for $k = 1, \ldots, K$ do
        $\theta_{\text{train},k} \leftarrow \theta_{\text{train},k} - \eta \nabla_{\theta_{\text{train},k}} J^{\dagger}(\theta_{\text{train},k})$ with $\mathcal{B}_{\text{train}}$ weighted by $w_{\text{train},k}$
        $\theta^{\mathbb{KL}}_{\text{train},k} \leftarrow \theta^{\mathbb{KL}}_{\text{train},k} - \eta \nabla_{\theta^{\mathbb{KL}}_{\text{train},k}} J^{\dagger}(\theta^{\mathbb{KL}}_{\text{train},k})$ with $\mathcal{B}_{\text{train}}$ weighted by $w_{\text{train},k}$
        $\theta_{\text{valid},k} \leftarrow \theta_{\text{valid},k} - \eta \nabla_{\theta_{\text{valid},k}} J^{\dagger}(\theta_{\text{valid},k})$ with $\mathcal{B}_{\text{valid}}$ weighted by $w_{\text{valid},k}$
    end for
    # Update V-function parameters
    $\psi_{\text{train}} \leftarrow \psi_{\text{train}} - \eta \nabla_{\psi_{\text{train}}} J^{\dagger}(\psi_{\text{train}})$ with $\mathcal{B}_{\text{train}}$
    $\psi^{\mathbb{KL}}_{\text{train}} \leftarrow \psi^{\mathbb{KL}}_{\text{train}} - \eta \nabla_{\psi^{\mathbb{KL}}_{\text{train}}} J^{\dagger}(\psi^{\mathbb{KL}}_{\text{train}})$ with $\mathcal{B}_{\text{train}}$
    $\psi_{\text{valid}} \leftarrow \psi_{\text{valid}} - \eta \nabla_{\psi_{\text{valid}}} J^{\dagger}(\psi_{\text{valid}})$ with $\mathcal{B}_{\text{valid}}$
    # Update behavior policy parameters
    $\omega_b \leftarrow \omega_b + \eta \nabla_{\omega_b} \log p(\mathcal{B}_{\text{train}}|\omega_b)$ with $\mathcal{B}_{\text{train}}$
    # Update policy parameters
    $\omega \leftarrow \omega - \eta \nabla_{\omega} J(\omega)$ with $\mathcal{B}_{\text{train}}$
    # Update target network parameters (soft target update)
    $\bar{\psi}_{\text{train}} \leftarrow (1-\tau)\bar{\psi}_{\text{train}} + \tau \psi_{\text{train}}$
    $\bar{\psi}^{\mathbb{KL}}_{\text{train}} \leftarrow (1-\tau)\bar{\psi}^{\mathbb{KL}}_{\text{train}} + \tau \psi^{\mathbb{KL}}_{\text{train}}$
    $\bar{\psi}_{\text{valid}} \leftarrow (1-\tau)\bar{\psi}_{\text{valid}} + \tau \psi_{\text{valid}}$
    every $H_f$ iterations after initial warm-up
        Compute the hypergradient $\nabla_{\xi} \rho_{\widehat{M}_{\text{valid}}}(\pi_\omega)$ with $\mathcal{D}_{\text{valid}}$ using Eq. (18)
        $\xi \leftarrow \text{clip}(\xi + \eta_\xi \nabla_\xi \rho_{\widehat{M}_{\text{valid}}}(\pi_\omega), \xi^{\text{min}}, \xi^{\text{max}})$
end for
```

F. Random MDP: Experimental Protocol

We conducted experiments on randomly generated MDPs to investigate how safely and efficiently BOPAH can improve performance over the behavior policy with respect to the varying number of trajectories and the degrees of optimality. Basically, we follow the experimental protocol of (Laroche et al., 2019).

F.1. Random MDP Generation

For each run, we constructed a random MDP with $|S| = 50$, $|A| = 4$, $\gamma = 0.95$, and a fixed initial state s_0. The transition probability is constructed randomly in that it has connectivity of 4. More specifically, for each (s, a), we uniformly sampled 4 states $\{s'_1, s'_2, s'_3, s'_4\}$ without duplicates. Then, the transition probabilities to those states are set to $[p(s_1|s, a), p(s_2|s, a), p(s_3|s, a), p(s_4|s, a)] \sim \text{Dir}(1, 1, 1, 1)$, where Dir denotes a Dirichlet distribution. The reward of 1 is only given to the state that minimizes the optimal value at the initial state, whereas other states have zero rewards. The design of the reward function can roughly be understood as choosing the farthest state from the initial state as the goal state.

F.2. Behavior Policy Generation

Careful readers may notice that there are infinitely many ways to construct a ζ-optimal behavior policy. Among the infinite possibilities, we construct the behavior policy μ as the following process, where we referred to the implementation code of SPIBB (Laroche et al., 2019)[4]. The process consists of two steps. As the first step, we compute the optimal policy π^* and its optimal value function Q^*. Then, we *soften* the optimal policy to π_{soft} by $\text{softmax}\big(Q^*(s, a)/\tau\big)$ and increase the temperature τ until π_{soft} reaches $\frac{\zeta+1}{2}$-optimal performance. As the second step, we repeatedly perturb the π_{soft} by discounting the probability of selecting an optimal action until the performance reaches ζ-optimality. The pseudo-code of this process is provided in Algorithm 2.

Algorithm 2 Behavior Policy Generation

```
Input: MDP $M$, Optimality of the behavior policy $\zeta$
Compute the optimal policy $\pi^*$ and its value function $Q^*(s, a)$ on the given MDP $M$.
Initialize $\pi_{\text{soft}} \leftarrow \pi^*$
Initialize $\tau \leftarrow 10^{-7}$
while $\rho_M(\pi_{\text{soft}}) > \frac{1}{2}\rho_M(\pi^*) + \frac{1}{2}\Big(\zeta\rho_M(\pi^*) + (1-\zeta)\rho_M(\pi_{\text{unif}})\Big)$ do
    Update $\pi_{\text{soft}}$ to $\pi_{\text{soft}}(a|s) \propto \exp\left(\frac{Q^*(s,a)}{\tau}\right)$
    $\tau \leftarrow \tau/0.9$
end while
Initialize $\mu \leftarrow \pi_{\text{soft}}$
while $\rho_M(\mu) > \zeta\rho_M(\pi^*) + (1-\zeta)\rho_M(\pi_{\text{unif}})$ do
    Sample $s \in \mathcal{S}$ uniformly at random.
    $\mu(a^*|s) \leftarrow 0.9\mu(a^*|s)$ where $a^* = \arg\max_a Q^*(s, a)$.
    Normalize $\mu(\cdot|s)$ so as to ensure $\sum_a \mu(a|s) = 1$.
end while
Output: The behavior policy $\mu$
```

After the behavior policy μ is generated, we sample trajectories using μ to form the batch data. Each episode is terminated either when 50 time steps have passed or the agent reached the rewarding state. We conducted experiments on the settings (the number of trajectories) $\in \{10, 20, 50, 100, 200, 500, 1000, 2000\}$. For each setting, we repeated 10k runs and reported the mean performance and the 5%-CVaR performance.

F.3. Hyperparameter Optimization via k-fold Cross-Validation

For the random MDP experiments, we divide the entire batch data $\mathcal{D}$ into 2 folds with the same number of trajectories, where each fold yields the corresponding MLE MDP pairs $\{(\widehat{M}^1_{\text{train}}, \widehat{M}^1_{\text{valid}}), (\widehat{M}^2_{\text{train}}, \widehat{M}^2_{\text{valid}})\}$. Then, for BOPAH, we

[4] https://github.com/RomainLaroche/SPIBB

optimize the hyperparameters to maximize the *average* performance over 2 valid MDPs:

$$\begin{aligned}
&\arg\max_{\xi} \frac{1}{2}\sum_{k=1}^{2} \rho_{\widehat{M}^k_{\text{valid}}}(\pi^k_\xi) \\
&\text{s.t. } \pi^k_\xi = \arg\max_{\pi} \mathbb{E}_{\pi, \widehat{M}^k_{\text{train}}}\left[\sum_{t=0}^{\infty} \gamma^t (r_t - \alpha_\xi(s_t)\mathbb{KL}^{\pi,\mu}_{s_t})\right]
\end{aligned}$$

This hyperparameter optimization is performed by the hypergradient ascent:

$$\xi \leftarrow \xi + \eta \left(\frac{1}{2}\sum_{k=1}^{2} \left[\nabla_\xi \rho_{\widehat{M}^k_{\text{valid}}}(\pi^k_\xi) \right] \right)$$

where each $\nabla_\xi \rho_{\widehat{M}^k_{\text{valid}}}(\pi^k_\xi)$ is obtained by Eq. (10), and η is a learning rate.

F.4. Hyperparameters of the Comparison Algorithms

We compare BOPAH with BasicRL, RaMDP (Petrik et al., 2016), RobustMDP (Nilim & El Ghaoui, 2005; Iyengar, 2005), and SPIBB (Laroche et al., 2019). For their hyperparameters, we follow the setting in the public code of SPIBB. For RaMDP, $\kappa = 0.003$ is used for the reward-adjusting hyperparameter. For RobustMDP, $\delta = 0.001$ is used for the confidence interval parameter to construct an uncertainty set. Finally, $N_\wedge = 5$ is used for SPIBB.

G. Continuous Control Experiment: Hyperparameter Settings

All the actor networks π_ω and critic networks Q_θ, V_ψ except for the KL-divergence critics $\mathbf{Q}_{\theta^{\mathbb{KL}}_{\text{train}}}, \mathbf{V}_{\psi^{\mathbb{KL}}_{\text{train}}}$ used two hidden layers of 100 units each. The KL-divergence critics $\mathbf{Q}_{\theta^{\mathbb{KL}}_{\text{train}}}, \mathbf{V}_{\psi^{\mathbb{KL}}_{\text{train}}}$ used two hidden layers of 450 units each. Other hyperparameters are described in Table 1. For the boundary values for importance sampling weight, we simply tested $w^{\text{min}} = \exp(-7) \approx 10^{-3}$ and $w^{\text{max}} = \exp(7) \approx 10^3$ and not tuned them further. The gradient penalty and boundary values for KL-coefficient are chosen such that the critic networks do not diverge.

Table 1. List of Hyperparameters

$N_\tau^T = 800$	Number of trajectories in training data
$N_\tau^V = 200$	Number of trajectories in validation data
$\gamma = 0.99$	Discount factor
$B = 64$	Minibatch size
$\tau = 5 \times 10^{-3}$	Soft target update rate
$H_f = 500$	Hypergradient update frequency
$H_w = 10^5$	Iterations for warm-up before the hypergradient updates
$K = 5$	Number of networks for bootstrapping
$\lambda_c = 1$	Conservativeness parameter
$N_{\text{cov}} = 20$	Number of samples used to evaluate covariances between Q functions
$d_\xi = 21$	Number of state-dependent functions $\{c_i(\cdot)\}_{i=1}^{d_\xi}$ for coefficient learning
$\kappa = 30$	Gradient penalty threshold (Hopper, Halfcheetah)
$\kappa = 100$	Gradient penalty threshold (Walker)
$\lambda_g = 2 \times 10^{-2}$	Gradient penalty coefficient
$\eta = 3 \times 10^{-4}$	Actor-critic learning rate
$w^{\text{min}} = \exp(-7)$	Clipped importance weight minimum
$w^{\text{max}} = \exp(7)$	Clipped importance weight maximum
$\xi^{\text{min}} = 0.75$	Minimum of ξ
$\xi^{\text{max}} = 500$	Maximum of ξ

H. Continuous Control Experiments using Entire Training Experiences

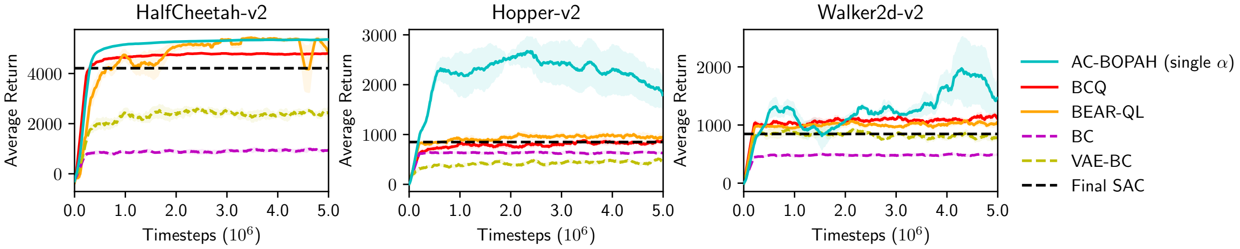


Figure 6. The results of continuous control experiments averaged over 3 trials, which are moving-averaged with a window size of 20. The shaded area represents the standard error.

As we only demonstrated the performance of our algorithm using single near-optimal data collection policy, we conducted additional experiments using different data distributions to complement the experiments shown in the main text. In this experiment, we evaluate the effectiveness of AC-BOPAH on continuous control tasks similar to the subsection 7.2, but with differently collected data. We first obtained the behavior policy by running SAC (Haarnoja et al., 2018) for half-million steps and stored all experienced transitions. This collection procedure, which corresponds to the *Final buffer* experiment in (Fujimoto et al., 2019), creates a dataset with a diverse set of states and actions that are sufficient to train an agent without regularization. We found that the α in AC-BOPAH converged to minimum values due to the diverse and sufficiently large dataset, and we only report one version of AC-BOPAH since using state-dependent α did not improve from using single α. As presented in Figure 6, even with differently collected dataset, AC-BOPAH outperforms other baseline algorithms as well.