Behavior Estimation from Multi-Source Data for Offline Reinforcement Learning

Guoxi Zhang¹ and Hisashi Kashima¹,²
¹ Graduate School of Informatics, Kyoto University
² RIKEN Guardian Robot Project
guoxi@ml.ist.i.kyoto-u.ac.jp, kashima@i.kyoto-u.ac.jp

Abstract
Offline reinforcement learning (RL) have received rising interest due to its appealing data efficiency. The present study addresses behavior estimation, a task that aims at estimating the data-generating policy. In particular, this work considers a scenario where data are collected from multiple sources. Neglecting data heterogeneity, existing approaches cannot provide good estimates and impede policy learning. To overcome this drawback, the present study proposes a latent variable model and a model-learning algorithm to infer a set of policies from data, which allows an agent to use as behavior policy the policy that best describes a particular trajectory. To illustrate the benefit of such a fine-grained characterization for multi-source data, this work showcases how the proposed model can be incorporated into an existing offline RL algorithm. Lastly, with extensive empirical evaluation this work confirms the risks of neglecting data heterogeneity and the efficacy of the proposed model.

Introduction
In offline reinforcement learning (RL) (Lange, Gabel, and Riedmiller 2012), agents learn policies using existing data without costly online interaction. Due to such data efficiency, offline RL becomes an intriguing paradigm for applications such as recommender systems (Chen et al. 2019) and robot manipulation (Dasari et al. 2020).

Behavior estimation refers to the task of computing an estimate for data-generating policy. Such an estimate is an essential component of many offline RL algorithms. For example, the discounted COP-TD algorithm (Gelada and Bellemare 2019) and the OPPOSD algorithm (Liu et al. 2020) requires such estimate for importance sampling, and the BRAC-v algorithm (Wu, Tucker, and Nachum 2019) utilizes it in behavior regularization. Hence, behavior estimation is a premise for these algorithms to work in practice.

In particular, the present study addresses behavior estimation on data collected from multiple sources, which is common when collecting demonstrations via crowdsourcing (Mandlekar et al. 2018, 2021). Because different sources may prefer different actions, action distributions in such data can be multi-modal. Figure 1 shows an minimalist example, where the green robot learns from data generated by two demonstrators. Because the blue and the green demonstrator prefers different actions, the empirical action distribution has two modes for that state. Now consider the canonical approach for behavior estimation that assumes a single Gaussian policy for continuous problems. As illustrated by the histogram in orange in Figure 1, neglecting data heterogeneity, the canonical approach fails to model the data well.

Perhaps surprisingly, this canonical approach is widely adopted in literature (Gelada and Bellemare 2019; Liu et al. 2020; Pavse et al. 2020), and data heterogeneity has not receive much attention. While learning latent action spaces using variational autoencoder (VAE) (Kingma and Welling 2014) seems to be a remedy (Fujimoto, Meger, and Precup 2019; Zhou, Bajracharya, and Held 2020), the assumption of Gaussian action prior limits its expressiveness.

The present study claims that, though the empirical action distribution is multi-modal, each individual data source still induces uni-modal action distributions. Thus, data heterogeneity can be properly modeled by inferring a set of policies from data and the assignments of trajectories to these policies. In the example shown in Figure 1, this corresponds to provide the agent with two policies and inferred assignments of trajectories. To this end, this work proposes a latent...
variable model and a model-learning algorithm to infer such a set from data. Furthermore, to demonstrate how the proposed model benefits offline RL, this work proposes localized BRAC-v, which integrates the proposed model into the BRAC-v algorithm. Since BRAC-v demonstrates reasonable performance on single-source data but degenerates on multi-source data (Fu et al. 2020), LBRAC-v showcases the efficacy of the proposed approach.

For efficient model learning, the present study uses an embedding-based model parameterization and a vector-quantized variational inference algorithm. Policies and trajectories are embedded into a low-dimensional space, and their representation vectors are learned via action reconstruction. After model learning, agents can use these embeddings to retrieve the corresponding policies for trajectories.

This study validates its claims empirically using the D4RL benchmark (Fu et al. 2020) and 15 new datasets. Experiment results show that algorithms that estimate single behavior policy worsened on multi-source data, which confirms the detriment of neglecting data heterogeneity. By contrast, LBRAC-v and existing VAE-based approaches achieved satisfactory performance. Moreover, LBRAC-v outperformed existing VAE-based approaches on datasets of moderate size and quality, while the existing approaches are better on large and high-quality datasets. These results illustrate the benefits and limitations of LBRAC-v. Lastly, with visualizations we show that the proposed model discovered patterns correlated with reward signals, which are further polished in policy learning. Increasing the size of the behavior set encourages clustering trajectories with behavior policies but inhibits such reward-related patterns. The contributions of this work are summarized as follows.

- This work considers behavior estimation on multi-source data and highlights the detrimental effect of behavior misspecification.
- It proposes a latent variable model to learn a behavior set to overcome behavior misspecification. In addition, it proposes LBRAC-v to showcase how the model benefits policy learning.
- It demonstrates the efficacy of LBRAC-v with experiments and provides visualization-based explanations.

**Related Work**

Prior art for offline RL discusses issues such as value over-estimation (Kumar et al. 2020) and poor convergence quality (Gelada and Bellemare 2019), and practical algorithms often require behavior estimation for performing importance sampling (Hallak and Mannor 2017; Gelada and Bellemare 2019; Nachum et al. 2019) or behavior regularization (Kumar et al. 2019; Wu, Tucker, and Nachum 2019; Kostrikov et al. 2021). However, none of them considers if data are from multiple sources.

Meanwhile, it is worth mentioning the difference between this study and imitation learning from noisy demonstrations (Wu et al. 2019; Tangkaratt et al. 2020; Wang et al. 2021). Although the latter opts for a well-performing policy, such a policy does not necessarily reproduce the data accurately. In consequence, quantities such as importance weights will be biased when computed using the policy.

Although several offline RL algorithms learn latent action spaces using VAE (Fujimoto, Meger, and Precup 2019; Zhou, Bajracharya, and Held 2020; Zhang et al. 2022), their assumption of Gaussian priors can be problematic. By contrast, the proposed model learns a discrete set of behavior policies that address multiple modality in data.

**Problem Statement**

**Preliminaries**

**Markov Decision Process:** An infinite-horizon discounted Markov decision process (MDP) \(\langle S, A, R, P, \gamma \rangle\) is a mathematical model for decision-making tasks. A state \(s \in S\) represents information provided to the agent for making a decision, and \(S\) is the set of all possible states. An action \(a \in A\) is an available option for a decision, and \(A\) is the set of possible actions. This study focuses on continuous control tasks, so actions are vectors in \(\mathbb{R}^{d_a}\) and \(d_a \in \mathbb{N}_+\).

The reward function \(R : S \times A \rightarrow \mathbb{R}\) provides the agent with scalar feedback for each decision. \(P\) is a distribution over states conditioned on states and actions, which governs state transitions. \(\gamma \in (0, 1)\) is the discount factor used to define Q-functions.

**Trajectories:** An MDP prescribes a sequential interaction procedure. Suppose the agent learns to navigate through a maze. Its current location is its states. At state \(s\), the agent selects an action \(a\) with a policy \(\pi\) and the corresponding reward for taking \(a\) at \(s\) is measured by its sum \(\sum_{t=1}^{\infty} \gamma^{t-1} r_{m, t}\). A useful transition from \(m\) to \(n\) can be written as \(\langle s_m, a_m, r_m, s_n, a_n \rangle\), which refers to two consecutive states, an action, and the corresponding reward for taking \(a\) at \(s\).

**Policies and Q-functions:** An agent selects actions with a policy \(\pi : S \rightarrow \Delta(A)\), where \(\Delta(A)\) means the set of distributions over \(A\). The Q-function of \(\pi\), \(Q^\pi(s, a) : S \times A \rightarrow \mathbb{R}\), gives the expected sum of discounted rewards obtained by taking \(a\) at \(s\) and following \(\pi\) subsequently. That is, \(Q^\pi(s) = \mathbb{E}_{\tau_m}[\sum_{t=1}^{\infty} \gamma^{t-1} r_{m, t}]\). A useful notion is transition. A transition from \(m\) to \(n\) can be written as \(\langle s_m, a_m, r_m, s_n \rangle\), which refers to two consecutive states, an action, and the corresponding reward for taking \(a\) at \(s\).

**Behavior Estimation from Multi-source Data**

The input for behavior estimation is a fixed set of trajectories \(D = \{\tau_1, \ldots, \tau_M\}\). Canonicly, \(D\) is assumed to be generated by a single policy \(b\) called the behavior policy. By contrast, this study considers \(D = \{\tau_1, \ldots, \tau_M\}\) to be generated by multiple policies used by multiple sources.
The output is a behavior set \( \mathcal{B} = \{b_k\}_{k=1}^{K} \), where \( b_k \in \mathcal{B} \) is a policy and \( K \in \mathbb{N} \) is the number of policies. Besides, the learner also outputs an assignment matrix \( G \in \{0, 1\}^{M \times K} \) of trajectories to \( \mathcal{B} \), which enables it to use \( \mathcal{B} \) for policy learning. \( G_{a,u} = 1 \) means that \( r_a \) is considered to be generated by \( b_u \), and \( G_{a,u} = 0 \) otherwise, so \( G \) allows for retrieving the corresponding \( b_k \in \mathcal{B} \) for a given trajectory. To this end, the learner is provided with access to the index of trajectories, though for the sake of simplicity this index is only referenced in subscripts.

The learning problem can be summarized as follows.

- Input: offline trajectories \( \mathcal{D} \).
- Output: \( K \) behavioral policies \( \mathcal{B} = \{b_k\}_{k=1}^{K} \) and the assignment matrix \( G \).

### Learning the Behavior Set

This section first describes the proposed latent variable model, its parameterization, and the model learning algorithm. Then, this section describes the proposed LBRAC-v algorithm for offline RL.

### Proposed Latent-Variable Model

The proposed model assumes trajectories are generated by \( K \) distinct policies. To generate trajectory \( \tau_m \), we sample a categorical variable \( z_m \) supported on \( \{1, 2, \ldots, K\} \) and then roll out behavior policy \( b_{z_m} \). This means that, at step \( t \), we sample \( a_{m,t} \) from \( b_{z_m}(a|s_{m,t}) \). Figure 2 shows a graphical model representation for this process.

A running assumption is that each trajectory is generated solely by one policy in \( \mathcal{B} \). Accordingly, the proposed model exploits the identity of trajectories when learning \( \mathcal{B} \). To understand the benefit, consider two transitions \( (s_m, a_m, r_m, s_m') \) and \( (s_m', a_m', r_m', s_m'') \), such that \( s_m = s_{m'} \), \( m \neq m' \), and \( a_m \neq a_{m'} \). When \( \tau_m \) and \( \tau_{m'} \) are generated by different policies, \( a_m \) and \( a_{m'} \) are probably sampled from different Gaussian distributions. Existing VAE-based approaches becomes deficient in this case, as they assume a Gaussian prior for actions. However, by assigning \( \tau_m \) and \( \tau_{m'} \) to different elements in \( \mathcal{B} \), the proposed model can leverage the simplicity of Gaussian policies without sacrificing expressiveness.

Figure 3 is a diagram for the proposed model. It consists of two major components: a policy network and a Q-network. The policy network represents \( \mathcal{B} \) using an embedding matrix \( E \in \mathbb{R}^{K \times d_e} \) and two functions \( d_e \) and \( f_p \), where \( d_e \in \mathbb{N} \), \( f_p: \mathbb{R}^{d_e} \rightarrow \mathbb{R}^{d_p} \rightarrow \mathbb{R}^{d_p \times d_A} \) converts representations of states and policies to action distributions. For a state \( s_m \), let \( \mu_{s_m} \) and \( \sigma_{s_m} \) be the mean and diagonal elements of the covariance matrix for the action distribution at \( s_m \), respectively. Then,

\[
[\mu_{s_m} \sigma_{s_m}^T] = f_p(e_{z_m}, f_e(s_m)). \tag{1}
\]

The Q-network of the proposed model uses an embedding matrix \( H \in \mathbb{R}^{K \times d_e} \) and two functions \( f_{s,a} \) and \( f_Q \) to represent the Q-functions of policies in \( \mathcal{B} \). Each row in \( H \) is an embedding vector for a policy in \( \mathcal{B} \). \( f_{s,a}: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}^{d_{s,a}} \) encodes state and actions in to vectors in \( \mathbb{R}^{d_p} \), where \( d_{s,a} \in \mathbb{N} \), \( f_Q: \mathbb{R}^{d_{s,a} + d_e} \rightarrow \mathbb{R} \) converts representations into Q-values:

\[
Q_{s,a} = f_Q(h_{z_s}, f_{s,a}(s,a)). \tag{2}
\]
Note that in both the policy network and the Q-network, all policies (or Q-functions) share \( f_s \) and \( f_p \) (or \( f_s,a \) and \( f_Q \)). This parameter-sharing mechanism reduces computational costs when compared to using \( K \) separate policy networks (or Q functions). Moreover, it forces these functions to encode information that is common for all policies in \( B \), which is beneficial when using them for \( \pi \).

### Parameter Learning

We need to learn \( E \), \( H \), and parameters of \( f_s \), \( f_p \), \( f_s,a \) and \( f_Q \). Figure 3 depicts key operations of the learning algorithm. The main idea is to introduce a matrix \( W \in \mathbb{R}^{M \times ds} \) as variational parameters. Each row of this matrix is an embedding vector for a trajectory in \( D \). The proposed algorithm uses the following posterior distribution \( q(z_m|s_m, a_m) \):

\[
q(z_m = k|s_m, a_m) = \begin{cases} 
1 & \text{for } k = \arg \max_j e_j^T w_m, \\
0 & \text{otherwise.
} \end{cases} 
\]

(3)

Let \( \tilde{z}_m = \arg \max_j e_j^T w_m \). This posterior distribution places all probability mass on \( \tilde{z}_m \), and \( e_{\tilde{z}_m} \) can be considered as the latent encoding for the behavior policy of \( \tau_m \). Overall, parameters are learned via action reconstruction. Assuming a uniform prior for \( z_m \), this in turn maximizes the evidence lowerbound for actions. However, care must be taken, as what relates \( s_m \) and \( a_m \) to \( e_{\tilde{z}_m} \) in Equation 3 is a non-differentiable searching operation.

Inspired by Fortuin et al. (2019), the proposed algorithm uses an objective consists of three terms: \( L_{\text{rec}}, L_{\text{com}}, \) and \( L_Q \). Specifically, \( L_{\text{rec}} \) encourages action reconstruction. Recall that \( f_p \) outputs action distributions, and let \( \ell \) be the corresponding log-likelihood function. Then, \( L_{\text{rec}} \) is defined as

\[
L_{\text{rec}} = -\ell(f_p(f_s(s_m), e_{\tilde{z}_m}); a_m) - \ell(f_p(f_s(s_m), w_m); a_m).
\]

(4)

The first term encourages \( e_{\tilde{z}_m} \) to reconstruct actions, and the second term does likewise for \( w_m \). Yet using Equation 4 alone, \( w_m \) is not related to \( e_{\tilde{z}_m} \), so \( w_m \) and \( e_{\tilde{z}_m} \) are not updated in a way that is consistent with \( \tilde{z}_m = \arg \max_j e_j^T w_m \). \( L_{\text{com}} \) mitigates this issue by penalizing the discrepancy between them:

\[
L_{\text{com}} = 1 - w_m^T e_{\tilde{z}_m}.
\]

(5)

\( L_Q \) is used for learning the Q-functions, which originates from fitted Q evaluation (Le, Voloshin, and Yue 2019).

\[
L_Q = \mathbb{E}_{a' \sim h_{\pi}} [r_m + \gamma \tilde{Q}^{h_{\pi}}(s_m', a') - \tilde{Q}^{h_{\pi}}(s_m, a_m)]^2
\]

(6)

\( \tilde{Q}^{h_{\pi}} \) is the target network for \( Q^{h_{\pi}} \).

In summary, for a transition \((s_m, a_m, r_m, s'_m)\), the proposed algorithm utilizes the following objective function:

\[
L = L_{\text{rec}} + \alpha L_{\text{com}} + L_Q,
\]

(7)

where \( \alpha \) is a hyper-parameter fixed to 0.1 in experiments. After learning, the assignment matrix \( G \) of can be computed using \( W \) and \( E \). The column index of the only non-zero element in the \( m^{th} \) row of \( G \) is \( \tilde{z}_m \). Additional details for the proposed model is provided in Appendix A.

### The Proposed LBRAC-v Algorithm

To demonstrate how the proposed model benefits offline RL, this study presents LBRAC-v, an extension of the BRAC-v algorithm (Wu, Tucker, and Nachum 2019). To begin with, we first review BRAC-v and explains why it degenerates on multi-source data.

BRAC-v leverages behavior regularization for offline RL. It penalizes the discrepancy between the policy being learned \( \pi \) and the behavior policy \( b \). On a transition, it minimizes the following objectives:

\[
L_{\text{critic}} = \mathbb{E}_{a' \sim \pi(\cdot|s_m')} [r_m + \gamma(\tilde{Q}(s_m', a') - \beta D(\pi, b, s_m')) - \tilde{Q}(s_m, a_m)]^2,
\]

\[
L_{\text{actor}} = \mathbb{E}_{a' \sim \pi(\cdot|s_m)} [\beta D(\pi, b, s_m) - \tilde{Q}(s_m, a'')]^2.
\]

(8)

\( \tilde{Q} \) is the target network for \( Q^\pi \). \( D(\pi, b, s) \) estimates the discrepancy between \( \pi(\cdot|s) \) and \( b(\cdot|s) \), and \( \beta \) is a hyperparameter. Essentially, BRAC-v subtracts the Q value at \( s'_m \) (or \( s_m \)) by \( \beta D(\pi, b, s_m') \) (or \( \beta D(\pi, b, s_m) \)) to penalize \( \pi \) for deviating from \( b \). Wu, Tucker, and Nachum (2019) suggested using KL-divergence for \( D(\pi, b, s) \):

\[
D(\pi, b, s) = \mathbb{E}_{a \sim \pi(\cdot|s)} [\log(\pi(a|s)) - \log(b(a|s))].
\]

(9)

Because BRAC-v assumes a single behavior policy, it faces issues when handling multi-source data. Consider again the example in Figure 1. As the mode of the estimated single behavior policy locates between modes of the two behavior policies, \( D(\pi, b, s) \) unfortunately takes the minimum value between data modes. In consequence, BRAC-v encourages \( \pi \) to take out-of-distribution actions.

To overcome this drawback, LBRAC-v integrates the proposed model into BRAC-v. When learning from \( \tau_m \), it first find \( b_{\tilde{z}_m} \) in \( B \) and then penalizes the deviation of \( \pi \) against \( b_{\tilde{z}_m} \). This fine-grained characterization of behavior policies prevents \( \pi \) to take out-of-distribution actions.

Another idea behind LBRAC-v is to reuse \( f_s \), \( f_p \), \( f_s,a \), and \( f_Q \) of a trained proposed model. Because these functions are adapted to the entire behavior set, they are also suitable for policies that are close to \( B \). Thus, LBRAC-v can benefit from the parameter-sharing mechanism of the proposed model. Specifically, LBRAC-v learns two vectors \( e_\pi \in \mathbb{R}^{d_e} \) and \( h_\pi \in \mathbb{R}^{d_h} \) as embedding vectors for \( \pi \) and \( Q^\pi \). \( \pi \) is parameterized by replacing \( e_{\tilde{z}_m} \) with \( e_\pi \) in Equation 1, and \( Q^\pi \) is parameterized by replacing \( h_{\tilde{z}_m} \) with \( h_\pi \) in Equation 2. In practice, \( e_\pi \) and \( h_\pi \) may be added to a trained proposed model to jointly optimize Equation 8 and Equation 7.

As a final remark, the proposed model can be straightforwardly integrated with other offline algorithms that utilizes parametric estimates of the behavior policy or latent action spaces. The present study presents LBRAC-v to illustrate how practitioners can extend existing algorithms for multi-source datasets.
Experiments

Datasets

Three continuous control tasks were included in experiments: halfcheetah, walker2d, and hopper. For each task, four versions of datasets were taken from the D4RL benchmark (Fu et al. 2020). The random and medium versions were generated by a random policy and an RL agent, respectively. The medium-replay version contains transitions collected for training the RL agent, and the medium-expert version is a combination of medium version with expert demonstrations. The former two are single-source datasets, whereas the latter two are multi-source datasets.

However, as shown in Table 3 in Appendix B, for the same task these datasets differ in two of the three aspects: the number of behavior policies, the number of transitions, and the quality of behavior policies. They cannot directly reveal how the number of behavior policies affects performance. Thus, this study generates five new datasets for each task, denoted as heterogeneous-$k$ ($k \in \{1, 2, 3, 4, 5\}$). For each task, we first trained five soft actor-critic (SAC) (Haarnoja et al. 2018) for an equal number of steps. Then, we generated the heterogeneous-$k$ version by rolling out the first $k$ agents for one million transitions. Each agent generated $1/k$ transitions. Consequently, heterogeneous-$k$ only differ in the number of behavior policies, which allows for investigating data heterogeneity.

Statistics of all the datasets and more details for heterogeneous-$k$ are provided in Table 3 in Appendix B.

Alternative Methods

Alternative methods are selected for three purposes. BC, BRAC-v, CQL, and MOPO are selected to demonstrate how data heterogeneity affects performance. MOPP and F-BRC are selected to show that latest algorithms with sophisticated parameterization for behavior policies still suffer from this issue. BCQ and PLAS are two existing VAE-based approaches, which are selected to investigate whether and when learning a behavior set is beneficial.

- BC: a method that estimates single behavior policy.
- BRAC-v (Wu, Tucker, and Nachum 2019): a model-free offline RL method that minimizes the KL divergence against an estimated single behavior policy.
- CQL (Kumar et al. 2020): a model-free method that does not rely on behavior policies.
- MOPO (Yu et al. 2020): a model-based algorithm that estimates state transition distributions.
- MOPP (Zhan, Zhu, and Xu 2022): a model-based algorithm that uses an ensemble of autoregressive dynamics models as the behavior policy.
- F-BRC (Kostrikov et al. 2021): a model-free method that minimizes the Fisher divergence against a mixture model for behavior policies.
- BCQ (Fujimoto, Meger, and Precup 2019): an algorithm that learns a VAE for actions.
- PLAS (Zhou, Bajracharya, and Held 2019): an algorithm that learns a VAE for actions and uses it to represent $\pi$.

Results on the D4RL datasets for BC, BRAC-v, CQL, and BCQ are taken from (Fu et al. 2020), and results for F-BRC, MOPO, MOPP, and PLAS are taken from the corresponding papers. For heterogeneous-$k$ datasets, the present study used the code provided by Fu et al. (2020) for BRAC-v and BCQ and the official code for CQL and PLAS.

Evaluation Metric

Algorithms are compared by trajectory returns (i.e. nondiscounted sum of rewards) obtained over 20 test runs. Returns are normalized into 0-100 scale as suggested by Fu.
Implementation Details

dc was set to eight. fs and fp were parameterized by two layers of feed-forward networks with 200 hidden units, while fs,a and fq were parameterized similarly but with 300 hidden units. The learning rates for the policy network and the Q-network were 5 × 10⁻⁵ and 1 × 10⁻⁴. Other details are available in Appendix A and the code is available here³.

The proposed model and LBRAC-v are trained for K ∈ {1, 5, 10, 15, 20}. Table 1 reports the best value for each dataset. Table 4 and Table 5 report results for all K. Experiments were repeated for five different random seeds, and this paper reports the average value of metrics and standard deviations.

Results

First, compare LBRAC-v with BRAC-v, CQL, and MOPO. As shown in Figure 4, performance of BRAC-v and CQL worsened on data generated by multiple policies. When k = 5, BRAC-v lost about 50% of the performance when k = 1. In contrast, LBRAC-v outperformed BRAC-v and achieved consistent performance across k. Table 1 shows results on D4RL datasets. Although BRAC-v had moderate performance on medium versions, it failed on the medium-replay version of walker2d and the medium-replay and medium-expert version of hopper. CQL had strong performance on medium-expert versions but not on medium-replay versions. In contrast, LBRAC-v surpassed BRAC-v on 10 datasets and CQL on nine datasets. It also outperformed MOPO on medium versions and both multi-source versions, except for the medium-replay version of halfcheetah. These results confirm the existence of behavior miss-specification and the efficacy of LBRAC-v on both multi-source and single-source data.

Then, let us compare LBRAC-v with F-BRC and MOPP. Even though they use sophisticated parameterization for the behavior policy, they still assume a single behavior policy. Figure 4 shows that LBRAC-v outperformed F-BRC on nine of the twelve datasets and MOPP on ten datasets. These results corroborate the importance of explicitly modeling multiple behavior policies from data.

Now compare LBRAC-v with PLAS and BCQ, the two VAE-based algorithms. As shown in Figure 4, LBRAC-v outperformed BCQ and PLAS on the heterogeneous-k versions for hopper, but it was outperformed by PLAS for halfcheetah. For walker2d, the three methods had similar performance. These results indicate that both the VAE-based approach and the proposed model can model multi-source data. Table 1 reveals the strength and weakness of them. LBRAC-v outperformed PLAS on all the medium and medium-replay versions and the random version for halfcheetah and walker2d. It outperformed BCQ on every dataset except for the random and medium-expert versions of hopper. In short, PLAS and BCQ have better performance on the medium-expert versions but not for the rest. As shown in Table 3 in Appendix B, this version has one time more samples than other versions, and its average returns is also one time better. Thus, the VAE-based approach is suitable for large and high-quality datasets, whereas the proposed model is suitable for small or low-quality datasets.

Finally, this study provides insights about the proposals using visualizations. They were created by projecting W to 2D space using principle component analysis. E and eπ were also projected to that space. Figure 5 shows visualizations of LBRAC-v for walker2d datasets. K is set to one to analyze effects of parts other than E. It shows that trajectories were clustered by returns, which indicates that sharing

et al. (2020). The higher, the better. See Appendix B for more details.

Because of the inherent randomness of training RL agents, there is performance fluctuation among the behavior policies for heterogeneous-k. To eliminate this fluctuation, for experiments on these datasets the present study uses the ratio between normalized returns of an algorithm and the average normalized returns of behavior policies as evaluation metric. This metric is termed relative return.

### Table 1: Results on D4RL datasets

| Task     | Version | BC     | F-BRC  | MOPP   | PLAS   | BRAC-v | CQL    | BCQ    | MOPO   | LBRAC-v |
|----------|---------|--------|--------|--------|--------|--------|--------|--------|--------|---------|
| halfcheetah | random  | 2.1    | 33.3±1.3 | 9.4±2.6 | 25.8   | 31.2   | 35.4   | 2.2    | 35.4±2.5 | 26.37±0.29 |
|          | medium  | 36.1   | 41.3±0.3 | 44.7±2.6 | 42.2   | 46.3   | 44.4   | 40.7   | 42.3±1.6 | 52.14±0.11 |
|          | medium-replay | 38.4 | 43.2±1.5 | 43.1±4.3 | 43.9   | 47.7   | 46.2   | 38.2   | 53.1±2.0 | 48.41±0.14 |
|          | medium-expert | 35.8 | 93.3±10.2 | 106.2±5.1 | 96.6   | 41.9   | 62.4   | 64.7   | 63.3±5.0 | 94.22±1.00 |
| walker2d | random  | 1.6    | 1.5±0.7 | 6.3±0.1 | 3.1    | 1.9    | 7.0    | 4.9    | 13.6±2.6 | 11.96±4.39 |
|          | medium  | 6.6    | 78.8±1.0 | 80.7±1.0 | 66.9   | 81.1   | 79.2   | 53.1   | 17.8±19.3 | 82.40±1.30 |
|          | medium-replay | 11.3 | 41.8±7.9 | 18.5±8.4 | 30.2   | 0.9    | 26.7   | 15.0   | 39.0±9.6 | 76.18±8.47 |
|          | medium-expert | 6.4  | 105.2±3.9 | 92.9±14.1 | 89.6   | 81.6   | 111.0  | 57.5   | 44.6±12.9 | 109.65±0.23 |
| hopper   | random  | 9.8    | 11.3±0.2 | 13.7±2.5 | 10.5   | 12.2   | 10.8   | 10.6   | 11.7±0.4 | 9.63±0.27 |
|          | medium  | 29.0   | 99.4±1.0 | 31.8±1.3 | 36.9   | 31.1   | 58.0   | 54.5   | 28.0±12.4 | 100.50±0.57 |
|          | medium-replay | 11.8 | 35.6±1.0 | 32.3±5.9 | 27.9   | 0.6    | 48.6   | 33.1   | 67.5±24.7 | 81.89±12.77 |
|          | medium-expert | 111.9| 112.4±0.3 | 95.4±28  | 111.0  | 0.8    | 98.7   | 110.9  | 23.7±6.0 | 98.08±6.39 |
Figure 5: Visualizations of LBRAC-v on the four versions of walker2d datasets provided by D4RL. $K = 1$. Trajectories were clustered by returns, especially for medium-replay and medium-expert versions.

Figure 6: Visualizations of LBRAC-v on the medium-expert version of walker2d with different values of $K$. Increasing $K$ resulted in fine-grained clustering of trajectories around behavior policies, but reward-related patterns become less apparent.

Figure 7: Visualizations of the proposed model on the medium-expert version of walker2d.

1. For small $K$, the proposed model discovers reward-related patterns for trajectories.
2. Such patterns get polished in policy learning.
3. Large value of $K$ encourages clustering trajectories by behavior policies but inhibits reward-related patterns.

Interested readers may find visualization obtained during training in Figure 9 in Appendix C for more insights.

Conclusion

Behavior estimation is a premise task of many offline RL algorithms. This work considered a scenario where training data are collected from multiple sources. We showed that it was detrimental to estimate a single behavior policy and overlook data heterogeneity in this case. To address this issue, this work proposed a latent variable model to estimate a set of behavior policies and integrated this model to the BRAC-v algorithm to showcase its usage. Empirical results confirmed the efficacy of the proposed model and the proposed extension. Moreover, visualizations showed that the proposed model discovered reward-related patterns for trajectories, which were further enhanced in policy learning. The present study is one of the few if any that addresses behavior estimation on multi-source data in literature and lays foundation for applying offline RL in real-world applications.

Acknowledgements

This work was supported by JST CREST Grant Number JP-MJCR21D1. The authors would like to thank Han Bao for valuable feedback on an early draft of this paper.
Zhang, H.; Shao, J.; Jiang, Y.; He, S.; Zhang, G.; and Ji, X. 2022. State Deviation Correction for Offline Reinforcement Learning. In Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence, 9022–9030. Virtual: AAAI Press.

Zhou, W.; Bajracharya, S.; and Held, D. 2020. PLAS: Latent Action Space for Offline Reinforcement Learning. In Proceedings of the Fourth Conference on Robot Learning, 1719–1735. Virtual: PMLR.