When is Offline Policy Selection Sample Efficient for Reinforcement Learning?

Vincent Liu*
RBC Borealis
Toronto, Canada
vincent.liu@rbc.com

Prabhat Nagarajan
University of Alberta
Edmonton, Canada
nagarajan@ualberta.ca

Andrew Patterson
Alberta Machine Intelligence Institute
Edmonton, Canada
ap3@ualberta.ca

Martha White
University of Alberta
Edmonton, Canada
whitem@ualberta.ca

ABSTRACT

Offline reinforcement learning algorithms often require careful hyperparameter tuning. Before deployment, we need to select amongst a set of candidate policies. However, there is limited understanding about the fundamental limits of this offline policy selection (OPS) problem. In this work we provide clarity on when sample efficient OPS is possible, primarily by connecting OPS to off-policy policy evaluation (OPE) and Bellman error (BE) estimation. We first show a hardness result, that in the worst case, OPS is just as hard as OPE, by proving a reduction of OPE to OPS. As a result, no OPS method can be more sample efficient than OPE in the worst case. We then connect BE estimation to the OPS problem, showing how BE can be used as a tool for OPS. While BE-based methods generally require stronger requirements than OPE, when those conditions are met they can be more sample efficient. Building on this insight, we propose a BE method for OPS, called Identifiable BE Selection (IBES), that has a straightforward method for selecting its own hyperparameters. We conclude with an empirical study comparing OPE and IBES, and by showing the difficulty of OPS on an offline Atari benchmark dataset.

KEYWORDS

Offline reinforcement learning; off-policy policy evaluation

ACM Reference Format:
Vincent Liu, Prabhat Nagarajan, Andrew Patterson, and Martha White. 2026. When is Offline Policy Selection Sample Efficient for Reinforcement Learning?. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 9 pages. https://doi.org/10.65109/VDEF3989

1 INTRODUCTION

Offline reinforcement learning (RL)—learning a policy from a dataset—is useful for many real-world applications, where learning from online interaction may be expensive or dangerous [18]. There have been significant advances in offline policy learning algorithms with demonstrations that performant policies can be learned purely from offline data [1]. Successful use of these algorithms, however, requires careful hyperparameter selection [11, 14, 28].

Despite the fact that hyperparameter selection is essential, it remains relatively unexplored for offline policy learning. One of the reasons is that it is inherently hard, as we show formally in this paper. Unlike supervised learning where validation performance can be used to select hyperparameters, in offline RL it is hard to validate a policy's performance. In fact, it is well known that *off-policy policy evaluation* (OPE)—estimating the performance of a policy from offline data—is hard [27]. To emphasize this fact, in Figure 1, we visualize the poor correlation between the true performance and performance estimates using a standard OPE approach called Fitted Q Evaluation (FQE) [16]. The datasets are not adversarially designed; rather they are two Atari datasets that were designed to provide reasonable data for offline learning algorithms [9].

Most offline RL papers sidestep the issue of hyperparameter selection and report results based on the best hyperparameter settings. In other words, they tune their hyperparameters by checking the performance of the policy in the environment, which is against the spirit and formulation of offline RL. Consider, for example, the Conservative Q-learning (CQL) algorithm [15], used to obtain the policies for Figure 1. For their experiments, CQL is built on top of already-tuned online algorithms for their environments, which implicitly relies on tuning using the environment rather than just the offline dataset. They then also directly tune the step size using the online performance of the offline learned policy in the environment. Several papers [12, 13] use the same methodology. Some works focus on designing algorithms that are robust to a specific hyperparameter [5], or require only a small number of hyperparameters to be tuned [10]. These approaches, however, still require some hyperparameters to be tuned; they were tuned using the performance in the environment.

There are a handful of works that use only the offline dataset to select hyperparameters. These can be categorized as papers focused on algorithms for hyperparameter selection in offline RL based on OPE [20, 24, 31], or papers that introduce offline RL algorithms and use sensible heuristics to select hyperparameters in their experiments [14, 22, 26, 33]. The heuristics are often tailored to the specific algorithm presented in these papers and do not come with performance guarantees.

*Work done at University of Alberta.

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/VDEF3989

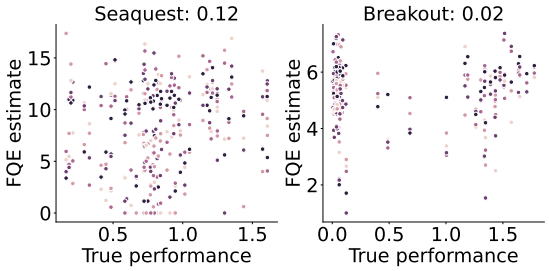


Figure 1: Correlation between true performance and estimated performance on Atari datasets. We generate 90 policies offline using the CQL algorithm with different choices for two hyperparameters: the number of training steps and the conservative parameter, and evaluate these policies using FQE on 5 different datasets. Each point in the scatter plot corresponds to a (policy, evaluation dataset) pair. The x-axis is the actual policy performance for that policy and y-axis is the estimated policy performance for that policy using that evaluation dataset. Colors represent different evaluation datasets. The Kendall rank correlation coefficient is shown in the title of the plot. If the FQE estimates accurately rank policies, we expect to see a strong linear relationship and a Kendall rank correlation coefficient close to 1. Neither of these behaviors are seen here, and it is clear FQE does not provide an effective mechanism to rank policies.

In general, there are as yet several open questions about the feasibility of hyperparameter selection in offline RL, both in theory and in practice. This is doubly true if we go beyond hyperparameter selection and consider selecting amongst different policy learning algorithms as well, each with their own hyperparameters. In this paper, we consider this more general problem, called *offline policy selection* (OPS), that can be used to select policy learning algorithms and their hyperparameters. Figure 2 depicts the full offline RL pipeline with OPS.

OPE is a general approach to OPS. We can simply run OPE to produce a value estimate for each candidate policy and select the policy with the highest value estimate. Moreover, we might intuit that OPS may, at least in some cases, be easier than OPE. OPS need only identify the best-performing policy, whereas OPE aims to produce an accurate value estimate of each policy, even if one candidate policy clearly dominates others. Therefore, the first question that needs to be answered is: *Is OPS easier than OPE?* Surprisingly, this basic question has not yet been explicitly answered.

Furthermore, we want to understand *when, or even if, alternative approaches can outperform OPE for OPS.* OPE might not be the ideal approach to OPS. OPE requires a large number of samples to evaluate any given policy in the worst case [27]. Moreover, OPE estimators do not resolve the OPS problem is because these OPE methods themselves have hyperparameters. For example, the MAGIC estimator [25] and the clipped importance sampling (IS) estimator [4] are sensitive to their hyperparameters. Even a variant of the IS estimators designed specifically for OPS [31] has hyperparameters. Other methods like FQE require learning a value function, and how to select the function class is still an open problem.

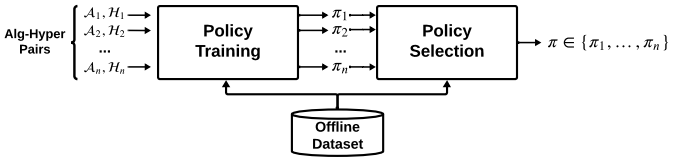


Figure 2: The offline RL pipeline with OPS. In the policy training phase, n algorithm-hyperparameter pairs are trained on an offline dataset to produce n candidate policies. An OPS algorithm then takes as input these n policies, and again utilizing offline data (potentially a validation dataset), select a final policy.

In this work, we make contributions towards answering these two questions. Our first contribution is to prove that OPS inherits the same hardness results as OPE, which has not been formally shown in the literature. This result implies no OPS approach can avoid exponential sample complexity in the worst case, and OPS requires additional assumptions to be sample efficient. Moreover, we connect the BE estimation problem to the OPS problem. We show that BE requires additional assumptions compared to OPE, but can be more efficient when the assumptions hold.

Our second contribution is to propose a simple OPS algorithm based on Bellman errors (BE), called Identifiable BE selection (IBES), with a simple mechanism to select hyperparameters. This is contrast to many OPE methods, even common approaches like FQE, for which hyperparameter selection is hard. In an empirical study, we compare different BE methods with varying sample sizes and show that IBES is more sample efficient across multiple environments.

2 BACKGROUND

In RL, the agent-environment interaction can be formalized as a finite horizon finite Markov decision process (MDP) $M = (\mathcal{S}, \mathcal{A}, H, s_0, Q)$. $\mathcal{S}$ is a set of states with size $S = |\mathcal{S}|$, and $\mathcal{A}$ is a set of actions with size $A = |\mathcal{A}|$, $H \in \mathbb{Z}^+$ is the horizon, and $s_0 \in \mathcal{S}$ is the initial state. We assume there is only a single initial state without loss of generality. The reward R and next state S' are sampled from Q, that is, $(R, S') \sim Q(\cdot|s, a)$. We assume the reward is bounded in $[0, r_{max}]$ almost surely so the total return of each episode is bounded in $[0, V_{max}]$ almost surely with $V_{max} = Hr_{max}$. The stochastic kernel Q induces a transition probability $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, and a mean reward function $r(s, a)$ which gives the mean reward when taking action a in state s.

A non-stationary policy is a sequence of memoryless policies $(\pi_0, \ldots, \pi_{H-1})$ where $\pi_h : \mathcal{S} \to \Delta(\mathcal{A})$. We assume that the set of states reachable at time step h, $\mathcal{S}_h \subset \mathcal{S}$, are disjoint, without loss of generality, because we can always define a new state space $\mathcal{S}' = \mathcal{S} \times [H]$, where $[H]$ denotes the set $[H] := \{0, 1, \ldots, H-1\}$. Then, it is sufficient to consider stationary policies $\pi : \mathcal{S} \to \Delta(\mathcal{A})$.

Given a policy π, for any $h \in [H]$ and $(s, a) \in \mathcal{S} \times \mathcal{A}$, we define the value function and the action-value function as $v_h^\pi(s) := \mathbb{E}^\pi[\sum_{t=h}^{H-1} r(S_t, A_t)|S_h = s]$ and $q_h^\pi(s, a) := \mathbb{E}^\pi[\sum_{t=h}^{H-1} r(S_t, A_t)|S_h = s, A_h = a]$, respectively. The expectation is with respect to $\mathbb{P}^\pi$, which is the probability measure on the random element $(S_0, A_0, R_0, \ldots, R_{H-1})$ induced by the policy π. Similar to the stationary policy, we also

use an action-value function q to denote the sequence of action-value function $(q_0, \ldots, q_{H-1})$ since the set of states reachable at each time step are disjoint. That is, for all $h \in [H]$ and $s \in \mathcal{S}_h$, $q(s, \cdot) = q_h(s, \cdot)$.

The Bellman evaluation operator $\mathcal{T}^\pi$ is defined as

$$(\mathcal{T}^\pi q)(s,a) = r(s,a) + \sum_{s' \in \mathcal{S}} P(s,a,s') \sum_{a' \in \mathcal{A}} \pi(a'|s') q(s',a'),$$

with $(\mathcal{T}^\pi q)(s, \cdot) = r(s, \cdot)$ for $s \in \mathcal{S}_{H-1}$. The Bellman optimality operator $(\mathcal{T} q)(s,a)$ is obtained if a greedy policy with respect to q is used. We use $J(\pi)$ to denote the value of the policy π, that is, the expected return from the initial state $J(\pi) = v_0^\pi(s_0)$. The optimal value function is defined by $v_h^*(s) := \sup_\pi v_h^\pi(s)$. A policy π is optimal if $J(\pi) = v_0^*(s_0)$.

In the offline setting, we are given a fixed set of transitions D with samples drawn from a data distribution μ. We consider the setting where the data is collected by a behavior policy π_b since the data collection scheme is more practical [29] and is used to collect data for benchmark datasets [9]. We use d_h^π to denote the data distribution at the horizon h by following the policy π, that is, $d_h^\pi(s,a) := \mathbb{P}^\pi(S_h = s, A_h = a)$, and $\mu_h(s,a) := \mathbb{P}^{\pi_b}(S_h = s, A_h = a)$.

3 ON THE SAMPLE COMPLEXITY OF OPS

We consider the offline policy selection (OPS) problem and off-policy policy evaluation (OPE) problem. We follow a similar notation used in Xiao et al. [29] to formally describe these problem settings. The OPS problem for a fixed number of episodes n is given by the tuple $(\mathcal{S}, \mathcal{A}, H, \nu, n, \mathcal{I})$. $\mathcal{I}$ is a set of instances of the form (M, d_b, Π) where $M \in \mathcal{M}(\mathcal{S}, \mathcal{A}, H, \nu)$ specifies an MDP with state space $\mathcal{S}$, action space $\mathcal{A}$, horizon H and the initial state distribution ν, d_b is a distribution over a trajectory $(S_0, A_0, R_0, \ldots, R_{H-1})$ by running the behavior policy π_b on M, and Π is a finite set of candidate policies. We consider the setting where Π is finite and does not depend on S, A or H.

An OPS algorithm takes as input a batch of data D, which contains n trajectories, and a set of candidate policies Π, and outputs a policy $\pi \in \Pi$. The goal is to return the best policy with high probability.

Definition 1 ((ε, δ)-sound OPS algorithm)**.** Given $\varepsilon > 0$ and $\delta \in (0,1)$, an OPS algorithm $\mathcal{L}$ is (ε, δ)-sound on instance (M, d_b, Π) if

$$\Pr_{D \sim d_b}(J_M(\mathcal{L}(D, \Pi)) \geq J_M(\pi^\dagger) - \varepsilon) \geq 1 - \delta,$$

where $\pi^\dagger$ is the best policy in Π. We say an OPS algorithm $\mathcal{L}$ is (ε, δ)-sound on the problem $(\mathcal{S}, \mathcal{A}, H, \nu, n, \mathcal{I})$ if it is sound on any instance $(M, d_b, \Pi) \in \mathcal{I}$.

Given a pair (ε, δ), the sample complexity of OPS is the smallest integer n such that there exists a behavior policy π_b and an OPS algorithm $\mathcal{L}$ such that $\mathcal{L}$ is (ε, δ)-sound on the OPS problem $(\mathcal{S}, \mathcal{A}, H, \nu, n, \mathcal{I}(\pi_b))$ where $\mathcal{I}(\pi_b)$ denotes the set of instances with data distribution d_b. That is, if the sample complexity is lower-bounded by a number N_{OPS}, then, for any behavior policy π_b, there exists an MDP M and a set of candidate policies Π such that any (ε, δ)-sound OPS algorithm on (M, d_b, Π) requires at least N_{OPS} episodes.

Similarly, the OPE problem for a fixed number of episodes n is given by $(\mathcal{S}, \mathcal{A}, H, \nu, n, \mathcal{I})$. $\mathcal{I}$ is a set of instances of the form (M, d_b, π) where M and d_b are defined as above, and π is a target policy. An OPE algorithm takes as input a batch of data D and a target policy π, and outputs an estimate of the policy value. The goal is to estimate the true value accurately.

Definition 2 ((ε, δ)-sound OPE algorithm)**.** Given $\varepsilon \in (0, V_{max}/2)$ and $\delta \in (0,1)$, an OPE algorithm $\mathcal{L}$ is (ε, δ)-sound on instance (M, d_b, π) if

$$\Pr_{D \sim d_b}(|\mathcal{L}(D, \pi) - J_M(\pi)| \leq \varepsilon) \geq 1 - \delta.$$

We say an OPE algorithm $\mathcal{L}$ is (ε, δ)-sound on the problem $(\mathcal{S}, \mathcal{A}, H, \nu, n, \mathcal{I})$ if it is sound on any instance $(M, d_b, \pi) \in \mathcal{I}$. Note that ε should be less than $V_{max}/2$, otherwise the bound is trivial.

3.1 OPE as Subroutine for OPS

It is obvious that a sound OPE algorithm can be used for OPS, since we can run the sound OPE algorithm to evaluate each candidate policy and select the policy with the highest estimate. As a result, the sample complexity of OPS is upper-bounded by the sample complexity of OPE up to a logarithmic factor. For completeness, we state this formally in the theorem below.

Theorem 1 (Upper bound on sample complexity of OPS)**.** Given an MDP M, a data distribution d_b, and a set of policies Π, suppose that, for any pair (ε, δ), there exists an (ε, δ)-sound OPE algorithm $\mathcal{L}$ on any OPE instance $I \in \{(M, d_b, \pi) : \pi \in \Pi\}$ with a sample size at most $O(\mathrm{N}_{OPE}(S, A, H, 1/\varepsilon, 1/\delta))$. Then there exists an (ε, δ)-sound OPS algorithm for the OPS problem instance (M, d_b, Π) which requires at most $O(\mathrm{N}_{OPE}(S, A, H, 2/\varepsilon, |\Pi|/\delta))$ episodes.

In terms of the sample complexity, we have an extra $\sqrt{\log|\Pi|/n}$ term for OPS due to the union bound. For hyperparameter selection in practice, the size of the candidate set is often much smaller than n, so this extra term is negligible. However, if the set is too large, complexity regularization [3] may need to be considered. In this paper, we consider a finite candidate set.

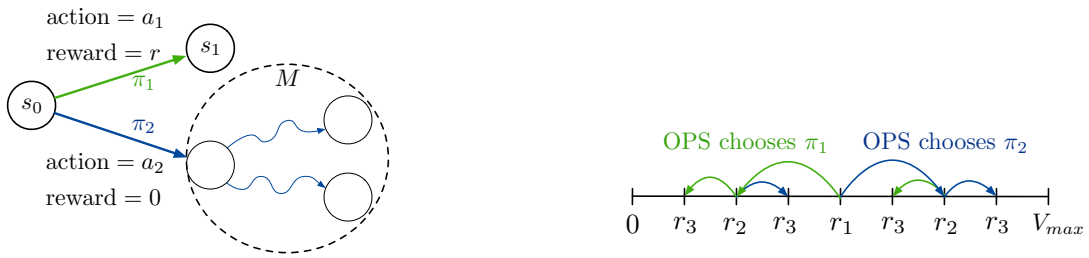
3.2 OPS is not Easier than OPE

We have shown that OPS is sample efficient when OPE is sample efficient. However, it remains unclear whether OPS can be sample efficient when OPE is not. In the following theorem, we lower bound the sample complexity of OPS by the sample complexity of OPE.

Theorem 2 (Lower bound on sample complexity of OPS)**.** Suppose for any data distribution d_b and any pair (ε, δ) with $\varepsilon \in (0, V_{max}/2)$ and $\delta \in (0,1)$, there exists an MDP M and a policy π such that any (ε, δ)-sound OPE algorithm requires at least $\Omega(\mathrm{N}_{OPE}(S, A, H, 1/\varepsilon, 1/\delta))$ episodes. Then there exists an MDP M' with $S' = S + 2$, $H' = H + 1$, and a set of candidate policies such that for any pair (ε, δ) with $\varepsilon \in (0, V_{max}/3)$ and $\delta \in (0, 1/m)$ where $m := \lceil \log(V_{max}/\varepsilon) \rceil \geq 1$, any (ε, δ)-sound OPS algorithm also requires at least $\Omega(\mathrm{N}_{OPE}(S, A, H, 3/2\varepsilon, 1/m\delta))$ episodes.

The proof sketch is to construct an OPE algorithm that queries OPS as a subroutine, as demonstrated in Figure 3. As a result, the sample complexity of OPS is lower bounded by the sample complexity of OPE. The proof can be found in the appendix.

There exist several hardness results for OPE in tabular settings and with linear function approximation [27, 32]. Theorem 2 implies



(a) Constructing M' and a candidate policy set

(b) Searching for the true policy value

Figure 3: Visual depiction of the reduction of OPE to OPS. Given a MDP M and a target policy π, we can construct a new MDP M' and two candidate policies $\{\pi_1, \pi_2\}$ for OPS, as shown in (a). The MDP construction was first mentioned in Wang et al. [27]. π_1 chooses a_1 in s_0, which leads to a terminal state s_1, and can arbitrarily select actions in other states. π_2 chooses a_2 and is otherwise identical to the target policy π. Figure (b) describes the search procedure to find the policy value by calling the OPS subroutine. When the OPS query returns π_1, we follow the green arrow. When the OPS query returns π_2, we follow the blue arrow. We can keep searching for the true policy value by setting r for the OPS query, until the desired precision is reached.

that the same hardness results hold for OPS since lower bounds for OPE are also lower bounds for OPS. Theorem 2, however, does not imply that OPS and OPE are always equally hard. There are instances where OPS is easy but OPE is not. For example, when all policies in the candidate set all have the same value, any random policy selection is sound. However, OPE can still be difficult in such a scenario.

3.3 Implication: We Need Assumptions for Sample Efficient OPS

In the previous section, we show that OPS and OPE problem are equally difficult in the worst case scenario. Based on the insights, we can show a lower bound on sample complexity of OPS for finite horizon finite MDPs. Below, we present an exponential lower bound, using the same lower bound construction from Xiao et al. [29] and Theorem 2.

Corollary 1 (Lower bound on the sample complexity of OPS)**.** For any positive integers S, A, H with $S > 2H$ and a pair (ε, δ) with $0 < \varepsilon \leq \sqrt{1/8}$, $\delta \in (0, 1)$, any (ε, δ)-sound OPS algorithm needs at least $\tilde{\Omega}(A^{H-1}/\varepsilon^2)$ episodes.

These result suggests that we need to consider additional assumptions on the environment, the data distribution, or the candidate set to achieve sample efficient OPS. Without such assumptions, any sound OPS algorithm would suffer exponential sample complexity in the worst case. One direction is to consider when OPE can be sample efficient, since we can leverage Theorem 1 to inherit the sample complexity result from OPE to OPS. There is a wealth of literature on OPE, in particular, FQE and MIS (or DICE) methods have been shown to be effective for OPS empirically [20, 31]. FQE has also been shown to be sample efficient under a standard data coverage assumption, that is, data coverage for all the candidate policies, and a function approximation assumption.

4 BELLMAN ERROR SELECTION FOR OPS

The natural alternative to OPE is to use Bellman errors (BE) or value error to select the best value functions [8], however, there is mixed evidence on whether such approaches are effective. Tang and Wiens [24] empirically show that selecting the candidate value function with the smallest TD errors perform poorly. Similarly, Paine et al. [20] present positive experimental results using FQE for OPS but conclude that the use of TD errors was ineffective. Other works, however, have developed new OPS algorithms that rely on the use of BE. Zhang and Jiang [34] propose a value-function selection algorithm called BVFT, which computes the (empirical) projected BE for each pair of candidate value functions. Lee et al. [17] provide a method for selecting the best function class from a nested set of function classes for Fitted Q-Iteration. These theoretically-sound methods, however, either rely on strong assumptions or are applicable only in specialized settings. It is unclear whether these methods are better than OPE.

In this section, we first show conditions when BE can be used OPS, which are generally stricter than the conditions needed for OPE methods. This suggests that, in theory, there is no benefit of using BE for OPS. On the other hand, BE offers a significant advantage in practice—we have a simple way to do model selection. As a demonstration, we propose a BE selection method that uses cross-validation for model selection.

4.1 BE Selection Problem

The selection problem using BE is slightly different from the OPS problem we defined in the previous section. Suppose we are given a set of candidate value functions $\mathcal{Q} := \{q_1, \ldots, q_K\}$ and let $\Pi = \{\pi_1, \ldots, \pi_K\}$ be the set of corresponding (greedy) policies, a common strategy is to select the action value function with the smallest BE [8]. In the finite horizon setting, the Bellman error with respect to q_i is defined as

$$\mathcal{E}(q_i) := \frac{1}{H} \sum_{h=0}^{H-1} \|q_i - \mathcal{T} q_i\|_{2,\mu_h}^2$$

where $\|q\|_{p,\mu_h} := (\sum_{(s,a) \in \mathcal{S}_h \times \mathcal{A}} \mu_h(s,a) |q(s,a)|^p)^{1/p}$. We define (ε, δ)-sound *BE selection* in the following.

Definition 3 ((ε, δ)-sound BE selection)**.** Given a set of candidate value functions $\mathcal{Q}$, $\varepsilon > 0$ and $\delta \in (0, 1)$, an BE selection algorithm $\mathcal{L}$, which takes $D, \mathcal{Q}$ as input and outputs $q \in \mathcal{Q}$, is (ε, δ)-sound on

$\mathcal{Q}$ if $\mathcal{E}(\mathcal{L}(D,\mathcal{Q})) \le \min_{i=1,\dots,|\mathcal{Q}|} \mathcal{E}(q_i) + \varepsilon$ with probability at least $1-\delta$.

In order to connect the BE selection problem to the OPS problem in Definition 1, we need the following assumptions:

(1) (date coverage)
$\exists C$ such that $\forall \pi \in \Pi \cup \{\pi^*\}$, $\max_{h\in[H]} \max_{s\in\mathcal{S}_h, a\in\mathcal{A}_h} \frac{d_h^\pi(s,a)}{\mu_h(s,a)} \le C$,

(2) (suboptimality of the candidate set) $\min_{q\in\mathcal{Q}} \mathcal{E}(q) \le \varepsilon_{sub}$.

With these assumptions, an $(\varepsilon_{est}, \delta)$-sound BE selection algorithm $\mathcal{L}$ on $\mathcal{Q}$ is $(2H\sqrt{C(\varepsilon_{sub}+\varepsilon_{est})}, \delta)$-sound for the OPS problem on Π. The bound can be easily derived based on existing results [7, 30]. We present the result here for clarity and include the complete theorem and the proof in the appendix.

Compared to OPE methods such as FQE, we need data coverage for both the candidate policies Π and an optimal policy. We also have an additional error ε_{sub} that does not go to zero even when we can collect more samples for OPS. Only ε_{est} goes to zero as n goes to infinity.

To see how poor the guarantee can be due to ε_{sub}, suppose we have two action values $q_1 = 100q^*$ and $q_2 = q^*$+some random noise, then q_1 has a large Bellman error but π_1 is actually optimal. We would choose q_2 since it has a lower Bellman error, even when we can collect an infinite number of samples. To obtain a meaningful guarantee, we need to include another value function, for example, $q_3 = q^*$ to make $\varepsilon_{sub} = 0$. As we collect more samples, we can estimate the Bellman error more accurately and choose q_3 eventually.

4.2 A Sound BE Selection Algorithm with Cross-Validation

If the data coverage and suboptimality of the candidate set assumptions hold, the remaining part is to have a sound BE selection algorithm. In this subsection, we show that there is a simple BE selection method, which we call Identifiable BE Selection (IBES).

In deterministic environments, the BE can be easily estimated using TD errors. Given a state-action value q with $v_q(s) := \max_a q(s,a)$ the corresponding state value, the BE estimate is

$$\mathrm{TDE}(q) := \frac{1}{|D|} \sum_{(s,a,s',r)\in D} (q(s,a) - r - v_q(s'))^2.$$

In stochastic environments, estimating BE typically involves an additional regression problem [2]. Antos et al. [2] propose an estimator for the BE by introducing an auxiliary function $g \in \mathcal{G}$ where $\mathcal{G}$ is a function class:

$$\hat{\mathcal{E}}(q) = \max_{g\in\mathcal{G}} [\frac{1}{|D|} \sum_{(s,a,s',r)\in D} (q(s,a) - r - v_q(s'))^2 - \frac{1}{|D|} \sum_{(s,a,s',r)\in D} (g(s,a) - r - v_q(s'))^2]. \quad (1)$$

We can rewrite the inner term in the equation by the change of variable [6, 21]. Let $t(r,s') := r + v_q(s')$ be the target and $\delta(s,a,r,s') := t(r,s') - q(s,a)$ be the TD error to simplify the equations. Consider a new auxiliary function $h(s,a) = g(s,a) - q(s,a)$,

Algorithm 1 Identifiable BE Selection (IBES) with holdout validation

Inputs: Candidate set $\mathcal{Q}$, training data D, validation data D_{val}
Hyperparameters: A set of function classes $\mathcal{G}_1, \dots, \mathcal{G}_M$ for model selection
Let $\delta(s,a,r,s') := r + v_q(s') - q(s,a)$
for $q \in \mathcal{Q}$ **do**
  for $m = 1, \dots, M$ **do**
    Perform regression:
    $\hat{g}_m \leftarrow \min_{g\in\mathcal{G}_m} \frac{1}{|D|} \sum_D (g(s,a) - \delta(s,a,r,s'))^2$
    Compute validation error:
    $l(\hat{g}_m) \leftarrow \frac{1}{|D_{val}|} \sum_{D_{val}} (\hat{g}_m(s,a) - \delta(s,a,r,s'))^2$
  Find the best function class: $k \leftarrow \arg\min_{m=1,\dots,M} l(\hat{g}_m)$
  Estimate the Bellman error for q:
  $\mathrm{BE}(q) \leftarrow \frac{1}{|D|} \sum_D 2\hat{g}_k(s,a)\delta(s,a,r,s') - \hat{g}_k(s,a)^2$
Output: $q^\dagger \leftarrow \arg\min_{q\in\mathcal{Q}} \mathrm{BE}(q)$

then the inner term in Equation (1) is

$$\begin{aligned} &\frac{1}{|D|} \sum_{(s,a,s',r)\in D} \left[(q(s,a) - r - v_q(s'))^2 - (g(s,a) - r - v_q(s'))^2 \right] \\ &= \frac{1}{|D|} \sum_{(s,a,s',r)\in D} \left[(t(r,s') - q(s,a))^2 - (t(r,s') - g(s,a))^2 \right] \\ &= \frac{1}{|D|} \sum_{(s,a,s',r)\in D} \left[2h(s,a)\delta(s,a,r,s') - h(s,a)^2 \right]. \end{aligned} \quad (2)$$

That is, we can also use an auxiliary function h to predict $\mathcal{T}q - q$, instead of $\mathcal{T}q$, and use the auxiliary function to estimate the Bellman error. The benefit is that the Bellman errors are more likely to be predictable. If there is an action-value function $q \in \mathcal{Q}$ with a small BE, the Bellman errors are nearly zero everywhere and any reasonable function class are able to represent the solution.

The remaining part is to find a function class that has low approximation error and a low statistical complexity. Fortunately, we can perform model selection to choose the function approximation $\mathcal{G}$. This is because we are running regression with fixed targets in Eq (1), and model selection for regression is well-studied. For example, consider a finite set of potential function classes $\mathcal{G}_1, \dots, \mathcal{G}_M$, we can use a holdout validation set or cross-validation to select the best function class and other hyperparameters. Therefore, we can choose a function class such that it has a low approximation error and a small complexity measure, which can potentially result in improved sample efficiency. We describe the full procedure of IBES in Algorithm 1.

4.3 Comparison to Existing Methods

There are other works consider selecting a value function that has the smallest BE or is the closest to the optimal value function. Farahmand and Szepesvári [8] consider selecting a value function such that, with high probability, the output value functions has the smallest BE. They propose to fit a regression model $\tilde{q}_i$ to predict $\mathcal{T}q_i$ and bound the BE by $\|q_i - \tilde{q}_i\|_{2,\mu}^2 + b_i$ where the first term can be viewed as the (empirical) projected Bellman error, and the second term b_i is a high-probability upper bound on the excess risk of the regression model, which is assumed to be given. There is a related

work on using BE selection method for OPS [35]. They propose a method called Supervised Bellman Validation (SBV), which is effectively an empirical version of the method from Farahmand and Szepesvári [8], without an additional upper bound on the excess risk. In our experiments, we find that our method outper forms SBV in terms of sample efficiency, likely because the auxiliary function h is used to predict the Bellman error, instead of $\mathcal{T}q$.

Zhang and Jiang [34] propose to use a (empirical) projected Bellman error, called BVFT loss, with piecewise constant function classes. Their selection algorithm chooses the value function with the smallest BVFT loss, assuming q^* is in the candidate set (approximately) and a stronger data assumption is satisfied. Interestingly, this condition on having q^* is essentially equivalent to our condition requiring small ε_{sub}, since q^* has exactly zero BE. The algorithms, though, are quite different from ours. Their method is computationally expensive since it scales with $O(|\Pi|^2)$ instead of $O(|\Pi|)$, making the method impractical when the candidate set is not small. Our method requires a weaker data coverage assumption and the computation cost scales linearly with $O(|\Pi|)$. Note that the sample complexity of our methods depends on the function class that is used to perform the regression. BVFT can be viewed as using the piecewise constant function classes to fit the Bellman target, the sample complexity depends on the piecewise constant function classes, which is measured by the number of discretization bins $(V_{max}/\varepsilon_{dct})^2$ and ε_{dct} is the discretization resolution.

4.4 Comparison between BE and OPE for OPS

Table 1 summarizes the comparison of BE and OPE methods. In theory, OPE methods such as FQE require weaker assumptions compared to IBES and BVFT. One of the strict assumptions for IBES and BVFT is that the candidate policy set contains a nearly optimal action value function, which we often cannot verify in the offline setting. This suggests that OPE is a more robust and reliable method for OPS.

On the other hand, if we satisfy these stronger assumptions, then IBES has several advantages. IBES can be much more sample efficient in deterministic environments, or even in stochastic environments by choosing appropriate function approximators. For IBES, we are not plagued by the issue of having hard-to-specify hyperparameters. This is critical for the offline setting, where we cannot test different hyperparameter choices in the environment.

Table 1: A summary of OPS methods. The first two methods are OPE methods and the last two methods are BE-based methods.

Method	Data Coverage	Suboptimality
IS	Action coverage	No assumption
FQE	Data coverage for Π	No assumption
IBES,SBV	Data coverage for $\Pi \cup \{\pi^*\}$	$\min_{q \in \mathcal{Q}} \mathcal{E}(q) \leq \varepsilon_{sub}$
BVFT	Stronger data coverage [34]	$q^* \in \mathcal{Q}$

5 EXPERIMENTS

In this section, we empirically investigate the different BE methods as well as FQE for OPS. The goal in the experiments is to gain a better understanding of how IBES and FQE perform when we vary important factors such as data coverage, sample size, and candidate policies. We first compare different BE-based methods to show the advantage of our proposed method IBES, and investigate the differences between IBES and FQE in classic control environments where we vary the data coverage. Finally, we perform an experiment on the Atari offline benchmark dataset.

To evaluate the performance of OPS, we consider the normalized top-k regret used in Zhang and Jiang [34]. Top-k regret is the gap between the best policy within the top-k policies, and the best policy among all candidate policies. We then normalize the regret by the difference between the best and the worst policy, so the normalized regret is between 0 and 1. The normalized regret can be interpreted as the percentage of degradation if we use offline selection instead of online selection, since online selection with Monte Carlo evaluation can always achieve a regret of zero. OPS corresponds to $k = 1$; for most results we use $k = 1$, but include some results for $k > 1$. All hyperparameters for OPS methods are selected using only the datasets, not by peaking at performance on the real environment; for more details, see the appendix.

5.1 Comparison between BE-based Methods

In our first set of experiments, we compare different BE-based methods for OPS. We conduct experiments on two standard RL environments: Acrobot and Cartpole. We also include the stochastic versions of these environments with sticky actions [19], which we call Stochastic Acrobot and Stochastic Cartpole. We generate a set of candidate policy-value pairs by running CQL with different hyperparameters on a batch of data collected by a trained policy with random actions taken 40% of the time, and generate datasets for OPS that provide good data coverage. We then use either FQE or IBES for OPS. We also included SBV [35] and BVFT [34],[1] and a random selection baseline which selects a policy uniformly at random from the candidate set.

We first compare different BE methods, and investigate the effect of using the Bellman error $\mathcal{T}q - q$ as the target. We include a baseline that uses $\mathcal{T}q$ as the target, called IBES-TQ. Note that we also perform model selection for SBV and IBES-TQ, similar to IBES. In Figure 4, we can see all BE methods converge to the same performance as the sample size grows larger, but IBES using the Bellman error as target is much more sample efficient than IBES-TQ and SBV across all environment. This is likely due to the fact that Bellman error is easier to predict so using a smaller neural network is sufficient and has a better sample efficiency. We also found BVFT performs similar to TDE (which are overlapping in Cartpole, Acrobot and Stochastic Cartpole), which is likely due to the fact that BVFT with small discretization is equivalent to TDE.

Now we show that the use of validation for model selection in IBES is important. To do so, we compare IBES (Algorithm 1) which takes as input a set of model classes that varies the number of hidden units to IBES with only a single model class where the network has a fixed number of hidden units. In our experiment, we use a two layer neural network model as the function approximation, and perform model selection to find the number of hidden units from the

[1]We modified the BVFT implementation from the author of Zhang and Jiang [34] (https://github.com/jasonzhang929/BVFT_empirical_experiments/).

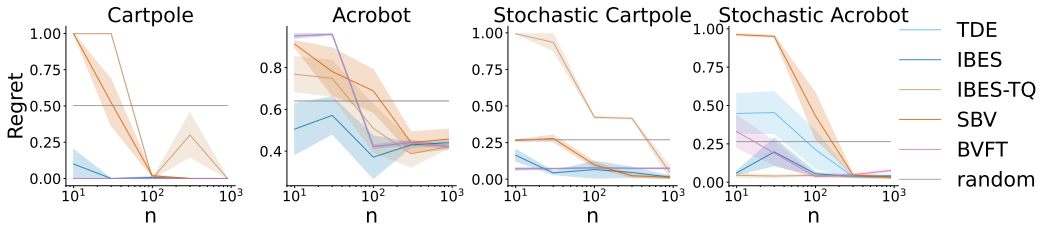


Figure 4: Comparison between BE methods. The figure shows the normalized top-1 regret with varying sample size, averaged over 10 runs with one standard error. IBES consistently achieves the lowest regret across environments.

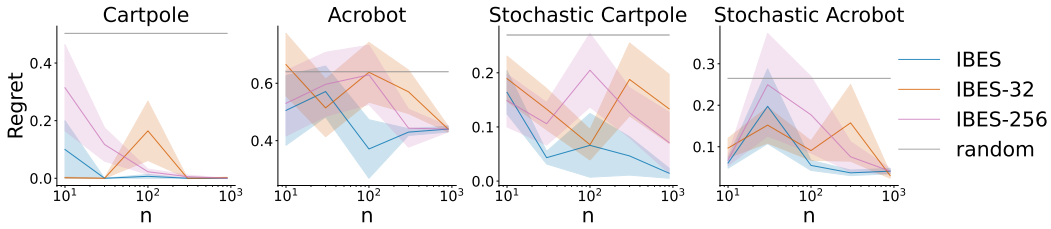


Figure 5: Comparison to BE with a fixed number of hidden units. The figure shows the normalized top-1 regret averaged over 10 runs with one standard error. IBES with model selection consistently achieves the lowest regret across environments.

set {32, 64, 128, 256} for IBES. In Figure 5, we include two baselines called IBES-32 and IBES-256, which has the hidden units of size 32 and 256 respectively. In general, IBES with model selection achieves a regret less than or equal to that of IBES with a fixed number of hidden units across all environments. The performance of IBES with model selection and IBES-256 converges as the sample size gets larger. This result shows an improvement in terms of sample efficiency due to model selection. In stochastic environments, IBES with a small number of hidden units does not work well even with a large sample size. The results suggest that the ability to perform model selection to balance approximation and estimation error is important to improve sample efficiency while achieving low regret.

In summary, the experiment suggests the proposed BE method is more sample efficient, due to the fact that the method predicts the Bellman error, and the model selection procedure.

5.2 Comparison between FQE and IBES under Different Data Coverage

In the second set of experiments, we aim to compare BE methods to OPE methods for model selection. Specifically, we compare IBES to FQE under different degrees of data coverage and different sample sizes. We design two different datasets for the experiments: (a) well-covered data is generated such that all candidate policies are well-covered, and (b) well-covered data with optimal trajectories includes more diverse trajectories collected by an ε-greedy optimal policy (which was used in the previous experiment).

Figure 6 shows the results for top-1 regret with varying numbers of episodes where the top row uses dataset (a) and the bottom row uses dataset (b). We first focus on the asymptotic performance ($n \approx 10^3$) across different datasets. FQE performs very well with a small regret on well-covered and diverse data. IBES performs better with optimal trajectories, especially in Cartpole. This result matches our theoretical result that IBES requires a stronger data coverage for an optimal policy. Moreover, IBES often performs better than FQE with a small sample size, suggesting that it could offer a slightly better sample efficiency than FQE.

Investigating the results deeper, we observed that when IBES does not perform well (Acrobot with optimal trajectories), it is often the case that one of the value functions has the smallest Bellman error but it is not far from optimal. When FQE does not perform well (Stochastic Cartpole with optimal trajectories), it is often the case that one of the candidate policies is highly overestimated. Therefore, we include a simple two-stage method that first uses FQE to select k_1 policies, then use IBES to find the top-k_2 policies amongst the k_1 selected policies. We set $k_1 = 10$ and $k_2 = 1$ in our experiment. The idea is similar to the two-stage method proposed in Tang and Wiens [24]. We label the two-stage method as FQE+IBES in the figure. We can see that it performs consistently well, better than either method alone. We hypothesize the explanation is that FQE usually performs very well for top-k selection with a large k, even though top-1 regret might be bad. We then use IBES to select from a subset of reasonably good candidate policies, where the candidate with the smallest error is more likely to be optimal. Further investigation about combining multiple OPS methods might be a promising research direction.

5.3 Comparison between FQE and IBES on Atari Datasets

Finally, we conduct experiments on benchmark Atari datasets to show the hardness of OPS for offline RL, and compare IBES to FQE in a more practical setting. We use the offline data on Breakout and

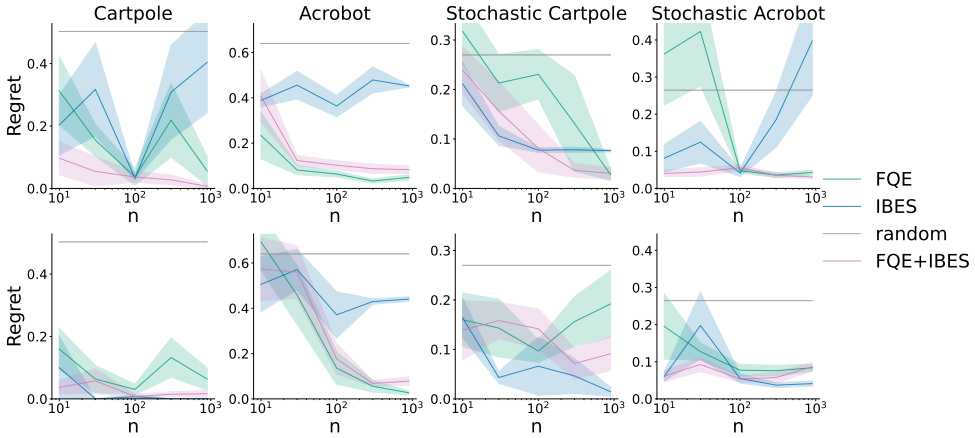


Figure 6: Comparison between IBES and FQE under different sample size and data distributions. The figure shows the normalized top-1 regret with varying sample size, averaged over 10 runs with one standard error.

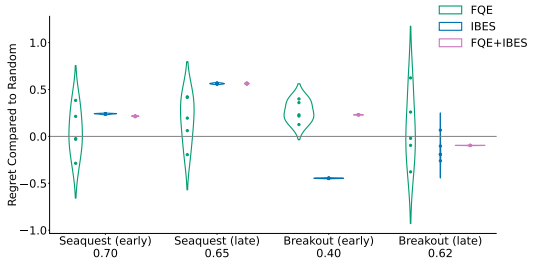


Figure 7: Regret improvement over the random baseline on the Atari dataset. We show the regret compared to the random baseline. A positive value means the method outperforms random selection, and a negative value means the method performs worse than random selection. We show the distribution of the regret improvement across 5 random seeds (each seed is a single point in the violin plots) and the regret of the random baseline under the x-axis labels. None of these methods can consistently outperform the random baseline, showing the hardness of OPS.

Seaquest from the DQN replay dataset[2], a commonly used benchmark in offline RL. We sample 1 million transitions from different learning stages to promote data coverage. Unlike our previous experiment, the data coverage might be poor due to the absent of explicit exploration to cover all candidate policies. We use 50% of the data to generate a set of candidate policy-value pairs by running CQL with different number of gradient steps and different regularization coefficients, as specified in Kumar et al. [15]. We use the other 50% data to perform OPS using FQE or IBES. We generate two candidate sets for each environment. The early-learning candidate contains policies with gradient steps in $\{50k, \cdots, 500k\}$ and final-learning candidate contains policies with gradient steps in $\{550k, \ldots, 1000k\}$. In this experiment, we use the CQL and FQE implementation from the d3rlpy library [23].

[2]https://research.google/resources/datasets/dqn-replay/

Figure 7 shows the top-1 regret. We can see that these none of the OPS methods can consistently outperform the random baseline, showing the hardness of OPS in a practical situation where we can not control the data coverage. For Breakout with early-learning candidate set, we found that IBES tends to pick the candidate value function with a small number of gradient steps. This is likely due to the fact that none of the candidate value functions are close to optimal (that is, large ε_{sub}) and the value function with a small number of training steps has a small magnitude and hence a small estimated Bellman error. IBES performs better with the candidate set containing policies in the final learning stage, where it is more likely to contain a value function that is close to optimal. This shows the limitation of IBES, as discussed in the previous section. We can also see that FQE is more robust to the choice candidate set since it does not require one of the value function being optimal, as long as all candidate set are covered by the data. However, it is more sensitive to the data used to run FQE, and hence high variance across different random seeds.

6 CONCLUSION

In this paper, we made contributions towards understanding when OPS is feasible for RL. One of our main results that the sample complexity of OPS is lower-bounded by the sample complexity of OPE implies that without satisfying conditions to make OPE feasible, we cannot do policy selection efficiently. Our second contribution is the proposed IBES algorithm, and we empirically show that it is more sample efficient than existing BE-based methods for OPS.

We expect active research topics will be (1) to identify suitable conditions on the policies, environments and data, to make OPS sample efficient, (2) to design offline RL algorithms that have sound methods to select their own hyperparameters, and (3) to investigate how to combine multiple methods for a better OPS algorithm. In offline RL, we cannot select hyperparameters by testing in the real-world, and instead are limited to using the offline data. OPS is arguably one of the most critical steps towards bringing RL into the real-world, and there is much more to understand.

REFERENCES

[1] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. 2020. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*.

[2] András Antos, Csaba Szepesvári, and Rémi Munos. 2008. Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning* (2008).

[3] Peter L Bartlett, Stéphane Boucheron, and Gábor Lugosi. 2002. Model selection and error estimation. *Machine Learning* (2002).

[4] Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X Charles, D Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. 2013. Counterfactual Reasoning and Learning Systems: The Example of Computational Advertising. *Journal of Machine Learning Research* (2013).

[5] Ching-An Cheng, Tengyang Xie, Nan Jiang, and Alekh Agarwal. 2022. Adversarially trained actor critic for offline reinforcement learning. In *International Conference on Machine Learning*.

[6] Bo Dai, Albert Shaw, Lihong Li, Lin Xiao, Niao He, Zhen Liu, Jianshu Chen, and Le Song. 2018. SBEED: Convergent reinforcement learning with nonlinear function approximation. In *International Conference on Machine Learning*.

[7] Yaqi Duan, Chi Jin, and Zhiyuan Li. 2021. Risk bounds and Rademacher complexity in batch reinforcement learning. In *International Conference on Machine Learning*.

[8] Amir-massoud Farahmand and Csaba Szepesvári. 2011. Model selection in reinforcement learning. *Machine Learning* (2011).

[9] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. 2020. D4RL: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219* (2020).

[10] Scott Fujimoto and Shixiang Shane Gu. 2021. A minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems* (2021).

[11] Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Thomas Paine, Sergio Gómez, Konrad Zolna, Rishabh Agarwal, Josh S Merel, Daniel J Mankowitz, Cosmin Paduraru, et al. 2020. RL unplugged: A suite of benchmarks for offline reinforcement learning. *Advances in Neural Information Processing Systems* (2020).

[12] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. 2021. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*.

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

[14] Aviral Kumar, Anikait Singh, Stephen Tian, Chelsea Finn, and Sergey Levine. 2021. A workflow for offline model-free robotic reinforcement learning. In *Conference on Robot Learning*.

[15] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative Q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems* (2020).

[16] Hoang Le, Cameron Voloshin, and Yisong Yue. 2019. Batch policy learning under constraints. In *International Conference on Machine Learning*.

[17] Jonathan N Lee, George Tucker, Ofir Nachum, Bo Dai, and Emma Brunskill. 2022. Oracle Inequalities for Model Selection in Offline Reinforcement Learning. *Advances in Neural Information Processing Systems* (2022).

[18] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643* (2020).

[19] Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. 2018. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research* (2018).

[20] Tom Le Paine, Cosmin Paduraru, Andrea Michi, Caglar Gulcehre, Konrad Zolna, Alexander Novikov, Ziyu Wang, and Nando de Freitas. 2020. Hyperparameter selection for offline reinforcement learning. *arXiv preprint arXiv:2007.09055* (2020).

[21] Andrew Patterson, Adam White, and Martha White. 2022. A generalized projected Bellman error for off-policy value estimation in reinforcement learning. *Journal of Machine Learning Research* (2022).

[22] Han Qi, Yi Su, Aviral Kumar, and Sergey Levine. 2022. Data-driven offline decision-making via invariant representation learning. *Advances in Neural Information Processing Systems* (2022).

[23] Takuma Seno and Michita Imai. 2022. d3rlpy: An Offline Deep Reinforcement Learning Library. *Journal of Machine Learning Research* (2022).

[24] Shengpu Tang and Jenna Wiens. 2021. Model selection for offline reinforcement learning: Practical considerations for healthcare settings. In *Machine Learning for Healthcare Conference*.

[25] Philip Thomas and Emma Brunskill. 2016. Data-efficient off-policy policy evaluation for reinforcement learning. In *International Conference on Machine Learning*.

[26] Brandon Trabucco, Aviral Kumar, Xinyang Geng, and Sergey Levine. 2021. Conservative objective models for effective offline model-based optimization. In *International Conference on Machine Learning*.

[27] Ruosong Wang, Dean P Foster, and Sham M Kakade. 2021. What are the statistical limits of offline RL with linear function approximation?. In *International Conference on Learning Representations*.

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

[29] Chenjun Xiao, Ilbin Lee, Bo Dai, Dale Schuurmans, and Csaba Szepesvari. 2022. The Curse of Passive Data Collection in Batch Reinforcement Learning. In *International Conference on Artificial Intelligence and Statistics*.

[30] Tengyang Xie and Nan Jiang. 2020. Q* approximation schemes for batch reinforcement learning: A theoretical comparison. In *Conference on Uncertainty in Artificial Intelligence*.

[31] Mengjiao Yang, Bo Dai, Ofir Nachum, George Tucker, and Dale Schuurmans. 2022. Offline policy selection under uncertainty. In *International Conference on Artificial Intelligence and Statistics*.

[32] Ming Yin and Yu-Xiang Wang. 2021. Optimal uniform OPE and model-based offline reinforcement learning in time-homogeneous, reward-free and task-agnostic settings. *Advances in Neural Information Processing Systems* (2021).

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

[34] Siyuan Zhang and Nan Jiang. 2021. Towards hyperparameter-free policy selection for offline reinforcement learning. *Advances in Neural Information Processing Systems* (2021).

[35] Joshua P Zitovsky, Daniel De Marchi, Rishabh Agarwal, and Michael Rene Kosorok. 2023. Revisiting Bellman Errors for Offline Model Selection. In *International Conference on Machine Learning*.