Towards Hyperparameter-free Policy Selection for Offline Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

How to select between policies and value functions produced by different training algorithms in offline reinforcement learning (RL)—which is crucial for hyperparameter tuning—is an important open question. Existing approaches based on off-policy evaluation (OPE) often require additional function approximation and hence hyperparameters, creating a chicken-and-egg situation. In this paper, we design hyperparameter-free algorithms for policy selection based on `BVFT` [XJ20], a recent theoretical advance in value-function selection, and demonstrate their effectiveness in discrete-action benchmarks such as Atari. To address performance degradation due to poor critics in continuous-action domains, we further combine `BVFT` with OPE to get the best of both worlds, and obtain a hyperparameter-tuning method for Q-function based OPE with theoretical guarantees as a side product.

1 Introduction and Related Works

Learning a good policy from historical data without interactive access to the actual environment, or offline (batch) reinforcement learning (RL), is a promising approach to applying RL to real-world scenarios when high-fidelity simulators are not available [LKTF20]. Despite the fast development in the training algorithms, a burning question that remains wide open is how to tune their hyperparameters, sometimes known as the offline policy selection problem [Pai+20; YDNTS20; Fu+21].

Standard approaches reduce the problem to off-policy evaluation (OPE), which estimates the expected return of the candidate policies and choose accordingly. Unfortunately, OPE itself is a difficult problem, and standard estimators such as importance sampling suffer exponential (in horizon) variance [LMS15; JL16]. While polynomial-variance estimators exist, either using TD (e.g., Fitted-Q Evaluation, or FQE [LVY19]) or marginalized importance sampling [LLTZ18; NCDL19; UHJ20], they require additional function approximation, inducing yet another set of hyperparameters (e.g., the neural-net architecture) which need to be carefully chosen. [Pai+20] recently conclude that FQE can be effective for offline policy selection, but "an important remaining challenge is how to choose hyperparameters for FQE". (Incidentally, we are able to address this question as a side product of our approach in Section 5). In other words, to tune hyperparameters for training we need to tune the hyperparameters for OPE, creating a chicken-and-egg situation. To this end, we want to ask:

Can we design effective *hyperparameter-free* methods for offline policy selection?

The question has been investigated in the theoretical literature [FS11], mostly reformulated so that we select indirectly among value functions instead of policies to trade-off directness for tractability. More precisely, [FS11] imagines that training algorithms produce candidate Q-functions $Q_1, Q_2, \ldots, Q_m$, which is a reasonable assumption as most offline algorithms produce value functions as a side product. The goal is to select $Q_i \approx Q^\star$—assuming one exists—so that the induced greedy policy, π_{Q_i}, is near-optimal. While $\|Q - Q^\star\|$ is only a surrogate for the performance of π_Q, the hope is that whether $Q \approx Q^\star$ can be more easily verified from holdout data without additional function approximation,

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

possibly by estimating the Bellman error (or residual) $\|Q - \mathcal{T}Q\|$. Unfortunately, $\|Q - \mathcal{T}Q\|$ is not amendable to statistical estimation in stochastic environments [SB18]. The naïve estimator which squares the TD error (see "1-sample BR" in Proposition 3) suffers the infamous *double-sampling bias* [Bai95], and debiasing approaches demand additional function approximation (and hence hyperparameters) [ASM08; FS11]. In prototypical real-world applications, hyperparameter-free heuristics such as picking the highest Q [GGMVS20] are often used despite the lack of theoretical guarantees, which we will compare to in our experiments.

In this paper, we attack the problem based on a recent theoretical breakthrough in value-function selection: [XJ20] propose a theoretical algorithm, `BVFT`, which provides a workaround to the double sampling issue without requiring additional function approximation; they estimate a form of projected Bellman error as a surrogate for $\|Q - Q^\star\|$, where the function class for projection is created out of the candidate Q's themselves. See Section 3 for details. Our contributions are 2-fold:

1. We design a practical implementation of `BVFT` based on novel theoretical observations, removing its last hyperparameter which determines a discretization resolution. We empirically demonstrate that `BVFT` enjoys promising performance in discrete-action benchmarks such as Atari games, sometimes using 20x less data than required by FQE-based policy selection.
2. The vanilla `BVFT` suffers performance degradation in continuous-action benchmarks, where the training algorithms often have an actor-critic structure and output a (π, Q) pair where Q is far away from $Q^\star$ for various reasons. To address this challenge, we propose `BVFT-PE`, a variant of `BVFT` that allows us to select among (π, Q) pairs and pick one where $Q \approx Q^\pi$ and π yields a high return. To further handle the issue that Q from the critic is often a poor fit of Q^π, we propose to use *multiple* OPE algorithms to re-fit Q^π, and run `BVFT-PE` among the produced (π, Q) pairs. While the OPE algorithms often have many hyperparameters that need to be set, `BVFT-PE` automatically chooses between them, leaving very few to no hyperparameters untunable. This allows us to combine the strengths of OPE and `BVFT` and get the best of both worlds. We also show additional results that `BVFT-PE` can be used for hyperparamter tuning in Q-function-based OPE and provide theoretical guarantees, which is of independent interest.

2 Preliminaries

Markov Decision Processes (MDPs) In RL, we often model the environment as an MDP, specified by its state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $R : \mathcal{S} \times \mathcal{A} \to [0, R_{\max}]$, transition function $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ ($\Delta(\cdot)$ is the probability simplex), discount factor $\gamma \in [0, 1)$, and a initial state distribution d_0. We assume $\mathcal{S} \times \mathcal{A}$ is finite but can be arbitrarily large. A deterministic policy $\pi : \mathcal{S} \to \mathcal{A}$ induces a random trajectory $s_0, a_0, r_0, s_1, a_1, r_1, \ldots$ where $s_0 \sim d_0, a_t = \pi(s_t)$, $r_t = R(s_t, a_t)$, and $s_{t+1} \sim P(\cdot|s_t, a_t)$, $\forall t$. We measure the performance of π using $J(\pi) := \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t | \pi]$. In discounted MDPs, there always exists an optimal policy $\pi^\star$ that maximizes $J(\cdot)$ for all starting states. It is the greedy policy of the optimal Q-function, $Q^\star$, i.e., $\pi^\star = \pi_{Q^\star} := (s \mapsto \arg\max_a Q^\star(s, a))$. $Q^\star$ is the fixed point of Bellman optimality equation, $Q^\star = \mathcal{T}Q^\star$, where $\forall f \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$, $(\mathcal{T}f)(s, a) := R(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[\max_{a'} f(s', a')]$. A related important concept is Q^π, which tells us the expected return of π when the trajectory starts from a specific state-action pair.

Offline Data In offline RL, we are given a dataset of (s, a, r, s') tuples and cannot directly interact with the MDP. For the theoretical part of the paper, we assume the standard *offline sampling protocol*, that the tuples are generated i.i.d. as $(s, a) \sim \mu$, $r = R(s, a)$, $s' \sim P(\cdot|s, a)$. With a slight abuse of notation, we also use $\mathbb{E}_\mu[\cdot]$ to denote the expectation over (s, a, r, s') sampled as above.

Policy Selection/Ranking We will use the following unified framework for policy selection throughout the paper: Suppose training algorithms (or the same algorithm with different hyperparameters) produce multiple (π, Q) pairs, $\{(\pi_i, Q_i)\}_{i=1}^m$, and our goal is to select a policy with good performance. The relationship between π and Q can differ in different contexts: for example, when training algorithms try to fit $Q^\star$ and induce a greedy policy, we have $\pi_i = \pi_{Q_i}$, and we only need to work with $\{Q_i\}_{i=1}^m$ as they contain all the relevant information. In the case where training algorithms have an actor-critic structure, π and Q are separate quantities and need to be reasoned about together. In real applications, the next step in the pipeline is to deploy the policy in the real system for online evaluation, and since we may have the resources to test more than 1 policies, we require all algorithms to produce a *ranking* over $\{\pi_i\}_{i=1}^m$, often by sorting the policies in ascending order w.r.t. a loss.

2.1 Experiment Setup

To avoid interrupting the flow of intertwined theoretical reasoning and empirical evaluation in the rest of the paper, we briefly describe our experiment setup here, with details deferred to Appendix C.

Environments and datasets We perform empirical evaluation on OpenAI Gym [Bro+16], Atari games [BNVB13], and Mujoco [TET12]. Taxi [Die00] is used for sanity check. We use standard offline datasets when available (RLUnplugged [Gul+21] for Atari, and D4RL [FKNTL21] for MuJoCo), and generate our own otherwise by mixing a trained expert policy with 30% chance of acting suboptimally. [Fu+21] have proposed a new benchmark for offline policy selection, which is also based on RLUnplugged/D4RL which we use and has only become available very recently. Also this benchmark (and [VLJY19]) focuses on OPE and policy selection without value functions, which does not exactly fit our purposes. We leave the evaluation on their benchmarks to future work.

Training algorithms We use several different training algorithms to generate the candidate models, including offline algorithms such as BCQ [FCGP19] and CQL [KZTL20]. To test the robustness of the policy-selection methods w.r.t. how the candidate policies are generated, we also use algorithms that learn from online interactions such as DQN [Mni+15] in some domains, though policy selection is always performed using separate offline datasets. This scenario is also of interest in its own right, as one can imagine training on (possibly imperfect) simulators via online algorithms and using limited realworld offline data for policy selection. For each algorithm, we consider different neural architectures, learning rates, and learning steps as hyperparameters to produce multiple candidate policies (and value functions) for selection; see Table 1 in Appendix C for details.

Performance metrics In each experiment, we run different policy-selection methods and evaluate the produced policy rankings using the following two metrics adapted from [YDNTS20]:

top-k normalized regret We take the best policy within the top-k recommended by the ranking, and calculate its gap compared to the best policy among all candidates. To make the value more interpretable, we normalize the regret by the gap between the best and the worst candidate policies.

top-k precision We take the k best policies in terms of their groudtruth values, and return the proportion of them appearing in the top-k policies in the ranking.

This process is repeated for 200 or 300 runs, with randomness coming from re-sampling a subset of the dataset for policy selection (usually of size $50,000$; FQE needs much more data and we do not run it on random subsets) and sampling $m = 10$ or 15 policies from all candidates for comparison. All figures report the mean with error bars of twice the standard errors, i.e., 95% confidence intervals.

3 Background: Batch Value-Function Tournament (BVFT)

We briefly introduce the theoretical basis of our approach. As mentioned in Section 1, the Bellman error $\|Q - \mathcal{T}Q\|$ is an appealing quantity because $Q = Q^\star \Leftrightarrow \|Q - \mathcal{T}Q\|_\infty = 0$, but it is not amendable to statistical estimation in stochastic environments due to the double-sampling bias [Bai95; ASM08; FS11]. Given this caveat, a closely related quantity has been extensively studied in the literature: given function class $\mathcal{G} \subset [0, \frac{R_{\max}}{1-\gamma}]^{\mathcal{S}\times\mathcal{A}}$, the (mean-squared) *projected Bellman error* of Q w.r.t. $\mathcal{G}$ is defined as

$$\|Q - \mathcal{T}_{\mathcal{G}}Q\|_{2,\mu}^2, \text{ where } \mathcal{T}_{\mathcal{G}}Q := \arg\min_{g\in\mathcal{G}} \mathbb{E}_\mu[(g(s,a) - r - \gamma \max_{a'} Q(s',a'))^2]. \tag{1}$$

Here $\|\cdot\|_{2,\mu}^2 = \mathbb{E}_\mu[(\cdot)^2]$, and the dependence of $\mathcal{T}_{\mathcal{G}}$ on μ is suppressed for readability. This quantity can be straightforwardly estimated from data when $\mathcal{G}$ has bounded statistical complexity; we just need to replace all $\mathbb{E}_\mu[\cdot]$ with their finite-sample approximation. The property of the projected Bellman error largely depends on the choice of $\mathcal{G}$, which needs to satisfy certain conditions for $\|Q - \mathcal{T}_{\mathcal{G}}^\mu Q\|$ to be a good surrogate for $\|Q - Q^\star\|$. The seminal work of [Gor95] has provided the following sufficient condition: (see [XJ20, Proposition 4] for a formal proof)

Proposition 1. *If (1) $\mathcal{G}$ is a piecewise-constant class, and (2) $Q^\star \in \mathcal{G}$, then $\mathcal{T}_{\mathcal{G}}$ is γ-contraction under $\|\cdot\|_\infty$, and $Q = Q^\star \Leftrightarrow \|Q - \mathcal{T}_{\mathcal{G}}Q\|_{2,\mu} = 0$ if μ is fully supported on $\mathcal{S}\times\mathcal{A}$.*

Being piecewise constant means that there exists a partitioning of $\mathcal{S}\times\mathcal{A}$, and $\mathcal{G}$ consists of all members of $[0, \frac{R_{\max}}{1-\gamma}]^{\mathcal{S}\times\mathcal{A}}$ that remain constant within each partition. A partitioning whose induced $\mathcal{G}$ satisfies (1) and (2) is also closely related to $Q^\star$-preserving state abstractions [LWL06].

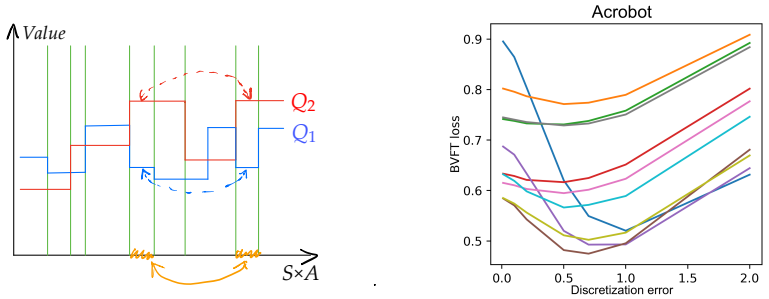

Figure 1: **Left:** Visualization of the partitioning that induces $\mathcal{G}_{1,2}$. The 2 subsets of $\mathcal{S} \times \mathcal{A}$ marked orange belong to the same partition despite being separated from each other, because Q_1 and Q_2 are constant across them. Since $\mathcal{S} \times \mathcal{A}$ is partitioned according to Q's *output*, the number of partitions is *independent* of $|\mathcal{S} \times \mathcal{A}|$, allowing `BVFT` to scale to arbitrarily complex state-action spaces. **Right:** `BVFT-loss` vs. discretization error ϵ_{dct} of 10 candidate Q's of a typical run in Acrobot, all having the U-shape predicted by our theoretical reasoning.

The problem is that it is very difficult to find $\mathcal{G}$ that satisfies the above 2 criteria,[1] and it will just be another set of hyperparameters to tune if we leave the design of $\mathcal{G}$ to the user. [XJ20] offers a resolution in the context of selecting $Q^\star$ from $\{Q_i\}_{i=1}^m$: consider the base case of $m = 2$ where we only need to select between Q_1 and Q_2. If one of them is $Q^\star$ (which can be relaxed) but we do not know which, we can still create $\mathcal{G}_{1,2}$ that satisfies both criteria of Proposition 1 as the minimal piecewise-constant class such that $\{Q_1, Q_2\} \subset \mathcal{G}_{1,2}$. If the output of each Q_i takes at most N possible values, $\mathcal{G}_{1,2}$ will be induced by partitioning $\mathcal{S} \times \mathcal{A}$ into at most N^2 regions; see Figure 1L. [XJ20] further shows that this idea extends to arbitrary m via pairwise comparison ("tournament"):

Proposition 2 (Simplification of [XJ20]; see Appendix B.1 for a proof sketch)**.** *If $Q^\star \in \{Q_i\}_{i=1}^m$ and μ is fully supported on $\mathcal{S} \times \mathcal{A}$, then $Q_i = Q^\star \Leftrightarrow Q_i$ having 0 `BVFT-loss`, where*

$$\texttt{BVFT-loss}(Q_i\,; \{Q_j\}_{j=1}^m) := \max_j \|Q_i - \mathcal{T}_{\mathcal{G}_{i,j}} Q_i\|_{2,\mu}. \tag{2}$$

For each Q_i, `BVFT` calculates its projected Bellman error w.r.t. $\mathcal{G}_{i,j}$—where $\mathcal{G}_{i,j}$ is created just like $\mathcal{G}_{1,2}$ but using Q_i itself and every other Q_j—and scores Q_i using the worst-case projected error. See [XJ20] for the complete theory that accounts for $Q^\star \notin \{Q_i\}_{i=1}^m$ and finite-sample effects. For readability we will stick to the above simplified reasoning.

Computation The empirical version of `BVFT-loss` can be computed exactly in closed form. The central step is the calculation of $\mathcal{T}_{\mathcal{G}} Q$. Recall that $\mathcal{G}$ is piecewise constant and induced by a $\mathcal{S} \times \mathcal{A}$ partitioning. For any $(\tilde{s}, \tilde{a})$, $(\mathcal{T}_{\mathcal{G}} Q)(\tilde{s}, \tilde{a})$ is simply the average of $r + \gamma \max_{a'} Q(s', a')$ over all data points (s, a, r, s') where (s, a) falls in the same partition as $(\tilde{s}, \tilde{a})$. Implemented with running averages, the computational complexity is $O(m^2 n)$ for m candidate functions and n data points in addition to $(|\mathcal{A}| + 1)mn$ candidate-function evaluations (i.e., caching $Q_i(s, a)$ and $\max_{a'} Q_i(s', a')$).

4 `BVFT` with Automatic Resolution Selection

Despite the appealing theoretical properties, it is unclear if `BVFT` can be converted into a practical algorithm. The sample complexity of estimating `BVFT-loss` depends on the complexity of $\mathcal{G}_{i,j}$, which is controlled by N^2 where N is the number of possible values in $[0, \frac{R_{\max}}{1-\gamma}]$ each Q_i can take. To handle the issue that N can be very large or even infinite, [XJ20] discretizes the output of $\{Q_i\}_{i=1}^m$ up to ϵ_{dct} error before producing $\{\mathcal{G}_{i,j}\}$, where ϵ_{dct} needs to be carefully chosen to trade-off between discretization errors and the complexity of $\mathcal{G}$ (which is now $O(1/\epsilon_{\text{dct}}^2)$). The theoretical value of ϵ_{dct} (as in [XJ20, Theorem 2]) not only relies on unknown properties of the MDP and the data, but is also very small which makes `BVFT-loss` expensive to estimate. Indeed, the previous theory predicts that `BVFT-loss` becomes an unstable statistic when $\epsilon_{\text{dct}} \to 0$ due to the unbounded complexity of $\mathcal{G}_{i,j}$.

[1] The ideal $\mathcal{G}$ can be created similarly to Figure 1 according to knowledge of $Q^\star$. In Appendix D we show that our approach closely tracks the skyline of using the ideal $\mathcal{G}$ in a tabular domain where we can compute $Q^\star$.

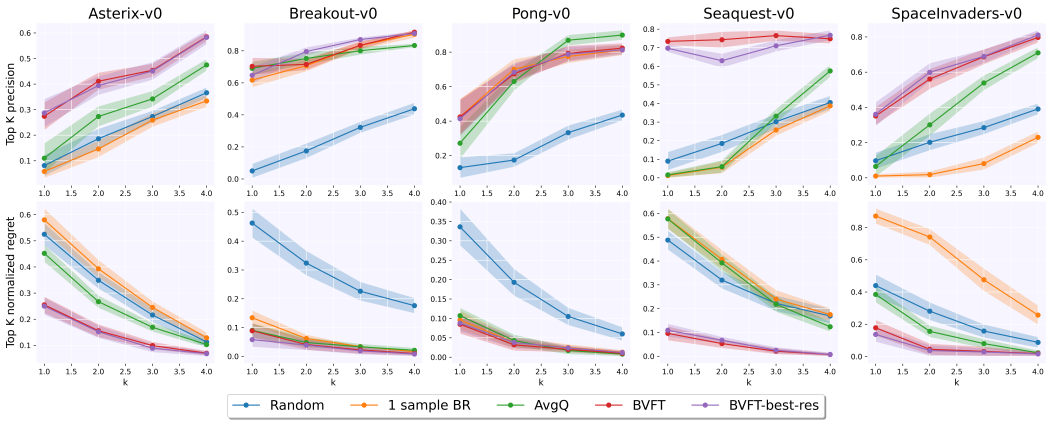

Figure 2: Top-k metrics of policy rankings vs. k in Atari. Row 1 shows top-k precision (the higher the better), and Row 2 shows top-k regret (the lower the better). Training algorithms are BCQ with different hyperparameters. The dataset for policy selection has 50,000 transition, which is an order of magnitude less than needed by FQE in Atari (see FQE in Enduro [VLJY19], as well as Figure 4).

To resolve this issue, we draw an interesting connection between `BVFT-loss` and the naïve "1-sample" estimator for $\|Q - \mathcal{T}Q\|$, which reveals the unexpected behavior of `BVFT-loss` in the regime of $\epsilon_{\text{dct}} \to 0$ that differs from the previous theoretical predictions:

Proposition 3. *Consider $\mathbb{E}_\mu[(Q(s,a) - r - \gamma \max_{a'} Q(s',a'))^2]$, the naïve and biased estimator for $\|Q - \mathcal{T}Q\|_{2,\mu}^2$ which we call "1-sample BR (Bellman residual)". If each candidate Q_i never predicts the same value for any two (s,a) pairs seen in the dataset, then with $\epsilon_{dct} = 0$, the empirial version of `BVFT-loss`$(Q_i; \{Q_j\}_{j=1}^m)$ and 1-sample BR of Q_i coincide (up to squaring).*

Proof. Given a dataset D consisting of (s,a,r,s') tuples, the empirical version of 1-sample BR is $\frac{1}{|D|}\sum_{(s,a,r,s')\in D}(Q(s,a) - r - \gamma \max_{a'} Q(s',a'))^2$, and that of `BVFT-loss` squared is $\max_j \frac{1}{|D|}\sum_{(s,a,r,s')\in D}(Q(s,a) - (\widehat{\mathcal{T}}_{\mathcal{G}_{i,j}} Q)(s,a))^2$, where $\widehat{\mathcal{T}}_{\mathcal{G}_{i,j}}$ is the empirical version of $\mathcal{T}_{\mathcal{G}_{i,j}}$ based on the same dataset. It suffices to show that for any data point (s,a,r,s'), $(\widehat{\mathcal{T}}_{\mathcal{G}_{i,j}} Q)(s,a) = r + \gamma \max_{a'} Q_i(s',a')$, which follows immediately from $\epsilon_{\text{dct}} = 0$ and Q_i never predicting the exact same value twice, since $\mathcal{G}_{i,j}$ does not provide any aggregation over data points in this case and $\widehat{\mathcal{T}}_{\mathcal{G}_{i,j}}$ coincides with the 1-sample Bellman update (c.f. the paragraph on computation in Section 3). □

This result implies that, 1-sample BR, which is a reasonable objective and coincides with $\|Q - \mathcal{T}Q\|$ in deterministic environments, provides a safeguard to `BVFT-loss` when ϵ_{dct} is too small. Since the double-sampling bias of 1-sample BR is positive [Bai95], we expect `BVFT-loss` to gradually decrease as ϵ_{dct} increases, hit a minimum, and increase again due to discretization errors when ϵ_{dct} becomes too large.[2] Indeed, this is precisely what we observe empirically; see Figure 1R.

Based on this novel observation, we propose to search for a grid of discretization errors in `BVFT` and pick the resolution that minimizes the loss (Eq.(2)); see pseudocode in Appendix A. In the experiments we will always use this rule to automatically select the discretization resolution for `BVFT` and its variants. The remaining questions can only be answered empirically: Does `BVFT` ever exhibit more interesting behavior than 1-sample BR (especially given their intimate relationship), is our resolution selection rule a good one, and how does `BVFT` compare to other baselines?

Empirical Evaluation To answer these questions, we empirically compare `BVFT`, 1-sample BR, and another simple hyperparameter-free heuristic, AvgQ [GGMVS20], which simply ranks $\{Q_i\}_{i=1}^m$ based on their average value on the data. "Random" ranks the policies in a completely random manner. `BVFT` and 1-sample BR rank $\{Q_i\}_{i=1}^m$ in ascending order of their loss functions, respectively. The experiments are done on 5 Atari games (Figure 2) and 4 Gym control problems (Figure 3; action space is made discrete in Pendulum).

[2] This can be violated in some extreme cases; see Appendix A for details.

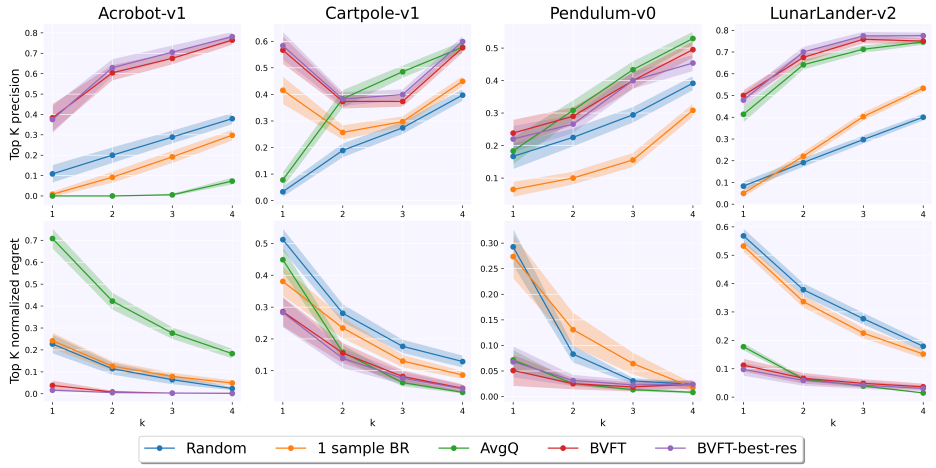

Figure 3: Results in Gym control problems. The dataset for policy selection has 50,000 transitions.

• *Does BVFT ever exhibit more interesting behavior than 1-sample BR?* Perhaps surprisingly, BVFT deviates significantly from the behavior of 1-sample BR, and almost always outperforms the latter by a wide margin. This is particularly interesting in CartPole and Pendulum, where the deterministic dynamics make 1-sample BR an **unbiased** estimate of $\|Q - \mathcal{T}Q\|$, and we expected it to perform well. Contrary to our expectation, 1-sample BR performs poorly even in these domains, where BVFT often performs much better. In fact, we observe similar phenomenon in a version of Taxi where we can compute $\|Q - \mathcal{T}Q\|$ as a skyline; see Appendix D.

• *Is our resolution selection rule effective?* To examine the effectiveness of our resolution selection rule, we compare to a skyline of BVFT itself ("BVFT-best-res"), where the best fixed resolution is chosen based on average statistics in the hindsight. As we can see, BVFT with automatic resolution selection closely tracks the performance of BVFT-best-res—to the extent that they are mostly indistinguishable—indicating that our resolution selection rule is near-optimal.

• *Comparison to AvgQ.* The simple heuristic of picking Q with the highest predicted value—which is sometimes used in prototypical applications [GGMVS20]—can be surprisingly effective sometimes, such as in Pendulum, LunarLander, and Pong. BVFT performs equally well in these domains. Moreover, as results from other domains reveal, AvgQ is very unstable and can fail catastrophically, such as in Acrobot and Seaquest, and is much less robust compared to BVFT.

One may wonder if AvgQ works better with *pessimistic* training algorithms: if every Q_i is an underestimation of the true return, then maximizing Q will be a well-justified heuristic. However, such pessimism is not always guaranteed especially when the function approximation is misspecified. In Appendix D we show additional experiments in Atari with a pessimistic algorithm CQL, and the results are qualitatively similar to Figure 2.

Comparison to OPE Despite the lack of no known method for tuning OPE's hyperparameters except for "cheating" in the simulator using online roll-outs (in Section 5 we will combine BVFT with OPE to address this issue), which makes it very difficult to have fair comparisons with OPE-based policy-selection methods, it is still instructive to have a rough sense of how our method compares to OPE. In this and the next sections, we choose Fitted Q-Evaluation (FQE) as a representative OPE algorithm.[3] Figure 4L shows the ranking metrics vs. sample size in 2 Atari games. We tune the neural architecture for FQE by choosing the training architecture that produces the *best* policy in Asterix.[4] While FQE has equally good performance compared to BVFT in the large-sample regime (10^6 transitions, which is typically required for FQE in Atari), the performance degradation is severe when sample size decreases. In comparison, BVFT is much more sample-efficient and provides almost the same level of performance with about 20x less samples, which is also what we used in Figure 2.

[3]Despite recent exciting developments in marginalized importance sampling (MIS), FQE often shows strong empirical performance thanks to its simplicity [VLJY19; Pai+20; Fu+21], whereas MIS is not as off-the-shelf due to its difficult optimization, so we defer the comparison as well as combining BVFT with MIS to future work.

[4]The recent OPE benchmarks [Pai+20; Fu+21] do not include Atari, so we choose our own hyperparameters.

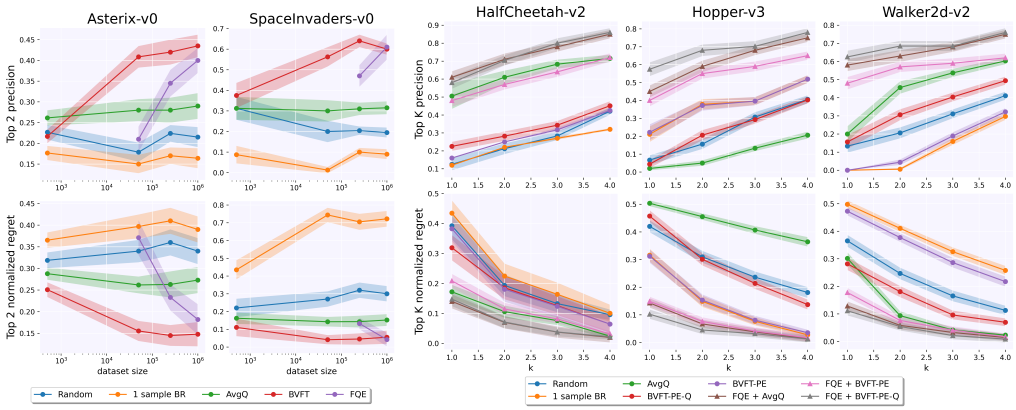

Figure 4: **Left:** Top-2 metrics vs. sample size in 2 Atari games, with FQE as an additional baseline. We did not experiment with FQE on small sample sizes because the trend of performance degradation is clear. **Right:** Top-k metrics vs. k across 3 Mujoco domains with continuous actions. All methods use a dataset of size $50,000$, except that the FQE component of any method uses 10^6.

Additional Results In Appendix D we examine the sensitivity of different methods to the dataset size and exploratoriness. `BVFT` remains advantageous compared to the baselines, is insensitive to data exploratoriness, and enjoys improved performance when more data becomes available unlike other hyperparmeter-free baselines.

5 Fighting Poor Value-Function Estimates with `BVFT` + OPE

The experiments so far are on discrete-action domains, and the candidate policies are always greedy w.r.t. the Q-function. However, in continuous-action domains, actor-critic-type algorithms are often used, where a critic Q is trained to evaluate a parameterized policy π, and the actor π is in turn improved based on Q. The trained (π, Q) pair in general does not satisfy $\pi = \pi_Q$, so it seems unwise to ignore π and only focus on Q in policy selection. On a related note, $\arg\max_{a'}$ is often expensive if not infeasible to calculate in continuous action spaces, so we need to make changes to `BVFT` anyway.

More importantly, the success of `BVFT` relies on the existence of a reasonable approximation of $Q^\star$ among the candidate functions, which does not always hold especially for actor-critic algorithms. Indeed, in the Mujoco experiments shown in Figure 4R, we observe poor performance of all hyperparameter-free methods, including variants of `BVFT` for continuous-action domains we develop later, and no method can consistently outperform the trivial baseline of random ranking. We address this issue of poor critics in the rest of this section. The issue is quite complicated and has many contributing factors—as we will explain subsequently—requiring us to take a multi-step approach to address one factor at a time.

Step 1: `BVFT-PE` and `BVFT-PE-Q` for Joint Selection of (π, Q) We first address the issue that $\arg\max_{a'}$ may be infeasible to calculate in continuous-action domains, and that both π and Q need to be taken into consideration for actor-critic algorithms. To address this issue, we propose a variant of `BVFT`, called `BVFT-PE`, with the following loss function:

$$\texttt{BVFT-PE-loss}((\pi_i, Q_i); \{(\pi_j, Q_j)\}_{j=1}^m) := \max_j \|Q_i - \mathcal{T}^{\pi_i}_{\mathcal{G}_{i,j}} Q_i\|_{2,\mu}, \tag{3}$$

where $\mathcal{G}_{i,j}$ is exactly the same as in `BVFT`, and $\mathcal{T}^{\pi_i}_{\mathcal{G}_{i,j}}$ is the same as Eq.(1), except that $\max_{a'} Q(s', a')$ should be replaced by $Q(s', \pi(s'))$. (In actor-critic algorithms, π is often a stochastic policy, in which case the term means $\mathbb{E}_{a' \sim \pi(\cdot|s')}[Q(s', a')]$.) This loss function naturally generalizes the `BVFT-loss`: when each π_i is the greedy policy of Q_i, we immediately have $\mathcal{T}^{\pi_i}_{\mathcal{G}_{i,j}} Q_i = \mathcal{T}_{\mathcal{G}_{i,j}} Q_i$, thus recovering `BVFT-loss` as a special case.

A closer inspection of Eq.(3) reveals an interesting fact: $\texttt{BVFT-PE-loss}((\pi_i, Q_i)) = 0$ as long as $Q_i = Q^{\pi_i}$, so `BVFT-PE-loss` is really a loss function for *policy evaluation*, hence the name `BVFT-PE`. (We will actually provide the theoretical guarantees of `BVFT-PE` for policy evaluation at the end of this section; see Theorem 4.) While the loss recovers `BVFT-loss` when $\pi = \pi_Q$,

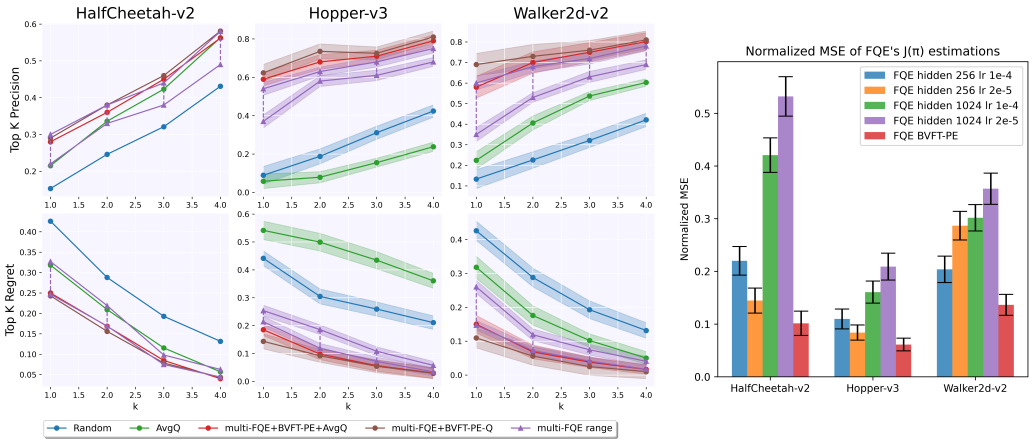

Figure 5: **Left:** Policy selection with multiple FQE instances in Mujoco. To avoid cluttering, we show the range of FQE performance across different hyperparameters (with upper and lower bounds connected by vertical dashed lines), and have removed error bars from HalfCheetah for better visibility (their widths are similar to the other two figures). The curves are cluttered in HalfCheetah and both our strategies perform similarly to the upper bound. **Right:** OPE accuracy for using `BVFT-PE` for hyperparameter tuning in FQE, where $J(\pi)$ is approximated by $\mathbb{E}_{s\sim d_0}[Q(s,\pi(s))]$.

more generally there can exist the degenerate cases of $Q = Q^\pi$ but π itself being a poor policy, where `BVFT-PE-loss`$((\pi, Q))$ is still 0. We address this issue by subtracting a Q term from the `BVFT-PE-loss`: `BVFT-PE-Q` $:=$ `BVFT-PE-loss` $- \lambda\mathbb{E}_\mu[Q]$, and the structure of this loss resembles a telescoping identity for $J(\pi)$ commonly used in the OPE literature; see Appendix A.1 for details.

Step 2: Re-fitting Q with OPE The above derivations address some of the basic theoretical issues, but still implicitly assume that at least one good π_i is paired with a $Q_i \approx Q^{\pi_i}$. In actor-critic algorithms, however, we often observe that the actor π converges way before the critic Q does, leaving us with a poorly estimated Q. Indeed, we have already seen in Figure 4 that `BVFT-PE` and `BVFT-PE-Q` are still not effective when applied to the candidate (π, Q) pairs.

A natural solution to the problem is to use OPE algorithms to refit Q^π to replace the critic Q, in order to provide a more accurate value function. We implement this using FQE as the OPE algorithm, though in principle we can use any OPE algorithm that provides an estimate of Q^π, including kernel loss [FLL19], MQL [UHJ20], or even a model-based method through planning. Figure 4 shows that `BVFT-PE` and `BVFT-PE-Q` enjoy strong empirical performance. However, the baseline of simply ranking the policies according to FQE itself ("FQE+AvgQ") is equally effective, which suggests that most of the performance gains should be attributed to FQE. That said, even if one's conclusion is that "OPE-based policy selection is more superior in continuous-action domains", we are just all the way back to where we started in Section 1:

How can we tune the hyperparameters for OPE itself?

Step 3: Re-fitting Q with *Multiple* OPE Algorithms Our last idea is retrospectively simple, yet allows us to combine the strengths of `BVFT` and OPE and get the best of both worlds. Suppose we have L OPE algorithms which we wish to select from. We will simply run each OPE algorithm on each policy, producing $m \times L$ candidate (π, Q) pairs in total, in the form of $\{(\pi_i, Q_i^l)\}$, where Q_i^l is the estimation of Q^{π_i} by the l-th OPE algorithm. There are two strategies we can proceed with:
Strategy 1: Run `BVFT-PE-Q` on all $m \times L$ pairs of (π, Q). Rank the m policies according to their highest position in the ranking produced by `BVFT-PE-Q`.
Strategy 2: Within each π_i, use `BVFT-PE` to select $Q_i^{l_i^\star} \approx Q^\pi$, then rank the policies based on the predictions of $Q_i^{l_i^\star}$ (e.g., using AvgQ).

Empirical Evaluation We use these strategies in the same experiment setting as Figure 4R, with multiple instances of FQE using different hyperparameters (see legend of Figure 5R). The results are shown in Figure 5L, where both strategies perform similarly to or better than the best FQE instance. We emphasize that this is highly nontrivial, because `BVFT-PE` and `BVFT-PE-Q` do not observe the *identities* of the FQE algorithms across different policies and runs (in fact, each FQE algorithm is represented as merely $2mn$ numbers), so matching the best FQE performance is strong evidence for

BVFT algorithms' model selection capabilities. Between the two BVFT strategies, Strategy 1 (using BVFT-PE-Q) slightly outperforms Strategy 2, but comes with an additional hyperparameter λ; we tuned it on Hopper and use the same constant in all experiments. In comparison, Strategy 2 is only slightly worse and does not have its own hyperparameters, which is an advantage.

Hyperparameter Tuning in OPE In Strategy 2, we are basically using OPE as the policy-selection algorithm, and BVFT-PE as a subroutine for hyperparameter tuning for OPE itself. Since OPE is an important component of the offline RL pipeline in its own right, our procedure for tuning OPE algorithms using BVFT-PE is also of independent interest. To this end, we conduct additional experiments in Mujoco to test the OPE accuracy of FQE with different hyperparameters. As Figure 5R shows, FQE with hyperparameters tuned by BVFT-PE consistently outperforms the *best fixed* set of hyperparameters across all 3 Mujoco domains. This implies that BVFT-PE can select the best hyperparameters for each policy individually, which is an appealing property. We also supplement the empirical results with a theoretical guarantee for selecting Q^π out of candidate functions for a fixed π, which follows from similar proof techniques as [XJ20, Theorem 2]; see Appendix B for the proof.

Theorem 4. *Let C be the same as in [XJ20, Theorem 2], which characterizes the exploratoriness of the data distribution. Consider any policy π and candidate Q-functions, $\{Q^l\}_{l=1}^L$, with $Q^l \in [0, \frac{R_{\max}}{1-\gamma}]$. Let $(\pi, \hat{Q})$ be the pair that minimizes the empirical version of `BVFT-PE-loss` applied to $\{(\pi, Q^l)\}_{l=1}^L$ with $\epsilon_{dct} = \frac{\epsilon R_{\max}}{8\sqrt{C}}$. Then, using a dataset of size $\tilde{O}\left(\frac{C^2 \ln(L/\delta)}{\epsilon^4(1-\gamma)^4}\right)$ where $\tilde{O}$ suppresses logarithmic terms, w.p. $\geq 1-\delta$, $\sup_{\nu:\|\nu/\mu\|_\infty \leq C} \|\hat{Q} - Q^\pi\|_{2,\nu} \leq \epsilon \cdot \frac{R_{\max}}{1-\gamma} + \frac{(2+4\sqrt{C}) \min_l \|Q^l - Q^\pi\|_\infty}{1-\gamma}$.*

As the theorem implies, when one of $\{Q^l\}_{l=1}^L$ is a good approximation of Q^π (i.e., $\min_l \|Q^l - Q^\pi\|_\infty$ is small), BVFT-PE is able to identify $\hat{Q} \approx Q^\pi$ with a polynomial sample complexity.

6 Discussion and Conclusion

We present BVFT and its variants based on recent theoretical advances [XJ20], which are (nearly) hyperparameter-free algorithms for policy selection in offline RL and empirically effective in discrete-action benchmarks. When combined with OPE algorithms such as FQE, variants of BVFT are also competitive in continuous-action benchmarks, and such a combination addresses the weaknesses of BVFT (relying on the existence of good value functions among the candidates) and those of OPE (having untunable hyperparameters) and gets the best of both worlds.

We conclude the paper with discussions of the limitations of our approach and open questions:

Sample efficiency Section 5 uses OPE to fit Q^π, which can be data intensive. A plausible solution is to re-use the training data for OPE, as suggested by [Pai+20]. However, data reuse can cause serious issues in realworld domains where the amount of training data taken for granted in deep RL is not available. How to improve the sample efficiency of BVFT + OPE is an important open question.

Computational complexity The $O(m^2)$ complexity of BVFT can be prohibitive if we wish to compare hundreds of models. A plausible solution is divide-and-conquer, i.e., eliminating bad models by running BVFT in smaller groups. It will be interesting to see if this compromises performance.

Applicability It is important to understand what type of domains BVFT is particularly suited for. Despite having provided partial answers (e.g., the discussion of discrete actions vs. continuous actions), we still need a more thorough answer based on comprehensive evaluation across more diverse domains and comparison to a wider range of baselines, which is beyond the scope of this paper since we focus on algorithm development and some of our contributions are orthogonal to existing approaches (e.g., BVFT + OPE). We look forward to investigating this question empirically in the future with the help of the public benchmarks that have become available very recently [Fu+21].

Data with insufficient coverage While results in Appendix D show that BVFT is insensitive to moderate changes in data exploratoriness, we will likely need to incorporate some form of pessimism–which is shown to be important for training [KRNJ20; JYW20; RZMJR21]—when the data coverage is seriously lacking. This can be, however, quite challenging, as BVFT uses a dynamic function space for projection ($\mathcal{G}_{i,j}$) that varies from candidate to candidate, and states considered covered due to generalization effects under one function space may be considered lacking data under a different one. How to resolve this issue and design a pessimistic version of BVFT is an interesting question.

References

[ASM08] András Antos, Csaba Szepesvári, and Rémi Munos. "Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path". In: *Machine Learning* 71.1 (2008), pp. 89–129.

[Bai95] Leemon Baird. "Residual algorithms: Reinforcement learning with function approximation". In: *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 30–37.

[BNVB13] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. "The arcade learning environment: An evaluation platform for general agents". In: *Journal of Artificial Intelligence Research* 47 (2013), pp. 253–279.

[Bro+16] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. *OpenAI Gym*. 2016. eprint: `arXiv:1606.01540`.

[CJ19] Jinglin Chen and Nan Jiang. "Information-Theoretic Considerations in Batch Reinforcement Learning". In: *Proceedings of the 36th International Conference on Machine Learning*. 2019, pp. 1042–1051.

[Die00] Thomas G Dietterich. "Hierarchical reinforcement learning with the MAXQ value function decomposition". In: *Journal of artificial intelligence research* 13 (2000), pp. 227–303.

[FCGP19] Scott Fujimoto, Edoardo Conti, Mohammad Ghavamzadeh, and Joelle Pineau. *Benchmarking Batch Deep Reinforcement Learning Algorithms*. 2019. arXiv: `1910.01708 [cs.LG]`.

[FKNTL21] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. *D4RL: Datasets for Deep Data-Driven Reinforcement Learning*. 2021. arXiv: `2004.07219 [cs.LG]`.

[FLL19] Yihao Feng, Lihong Li, and Qiang Liu. "A kernel loss for solving the bellman equation". In: *Advances in Neural Information Processing Systems*. 2019, pp. 15430–15441.

[FMP19] Scott Fujimoto, David Meger, and Doina Precup. *Off-Policy Deep Reinforcement Learning without Exploration*. 2019. arXiv: `1812.02900 [cs.LG]`.

[FS11] Amir-massoud Farahmand and Csaba Szepesvári. "Model selection in reinforcement learning". In: *Machine learning* 85.3 (2011), pp. 299–332.

[FSM10] Amir-massoud Farahmand, Csaba Szepesvári, and Rémi Munos. "Error Propagation for Approximate Policy and Value Iteration". In: *Advances in Neural Information Processing Systems*. 2010, pp. 568–576.

[Fu+21] Justin Fu, Mohammad Norouzi, Ofir Nachum, George Tucker, Ziyu Wang, Alexander Novikov, Mengjiao Yang, Michael R Zhang, Yutian Chen, Aviral Kumar, et al. "Benchmarks for Deep Off-Policy Evaluation". In: *arXiv preprint arXiv:2103.16596* (2021).

[GGMVS20] Diksha Garg, Priyanka Gupta, Pankaj Malhotra, Lovekesh Vig, and Gautam Shroff. "Batch-Constrained Distributional Reinforcement Learning for Session-based Recommendation". In: *arXiv preprint arXiv:2012.08984* (2020).

[Gor95] Geoffrey J Gordon. "Stable function approximation in dynamic programming". In: *Proceedings of the twelfth international conference on machine learning*. 1995, pp. 261–268.

[Gul+21] Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Tom Le Paine, Sergio Gomez Colmenarejo, Konrad Zolna, Rishabh Agarwal, Josh Merel, Daniel Mankowitz, Cosmin Paduraru, Gabriel Dulac-Arnold, Jerry Li, Mohammad Norouzi, Matt Hoffman, Ofir Nachum, George Tucker, Nicolas Heess, and Nando de Freitas. *RL Unplugged: A Suite of Benchmarks for Offline Reinforcement Learning*. 2021. arXiv: `2006.13888 [cs.LG]`.

[JL16] Nan Jiang and Lihong Li. "Doubly robust off-policy value evaluation for reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2016, pp. 652–661.

[JYW20] Ying Jin, Zhuoran Yang, and Zhaoran Wang. "Is Pessimism Provably Efficient for Offline RL?" In: *arXiv preprint arXiv:2012.15085* (2020).

[KRNJ20] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. "Morel: Model-based offline reinforcement learning". In: *arXiv preprint arXiv:2005.05951* (2020).

[KZTL20] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. "Conservative q-learning for offline reinforcement learning". In: *arXiv preprint arXiv:2006.04779* (2020).

[Lil+19] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. *Continuous control with deep reinforcement learning*. 2019. arXiv: 1509.02971 [cs.LG].

[LKTF20] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. "Offline reinforcement learning: Tutorial, review, and perspectives on open problems". In: *arXiv preprint arXiv:2005.01643* (2020).

[LLTZ18] Qiang Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. "Breaking the curse of horizon: Infinite-horizon off-policy estimation". In: *Advances in Neural Information Processing Systems*. 2018, pp. 5361–5371.

[LMS15] Lihong Li, Rémi Munos, and Csaba Szepesvári. "Toward Minimax Off-policy Value Estimation". In: *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*. 2015.

[LVY19] Hoang M. Le, Cameron Voloshin, and Yisong Yue. *Batch Policy Learning under Constraints*. 2019. arXiv: 1903.08738 [cs.LG].

[LWL06] Lihong Li, Thomas J Walsh, and Michael L Littman. "Towards a unified theory of state abstraction for MDPs". In: *Proceedings of the 9th International Symposium on Artificial Intelligence and Mathematics*. 2006, pp. 531–539.

[Mni+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. "Human-level control through deep reinforcement learning". In: *nature* 518.7540 (2015), pp. 529–533.

[NCDL19] Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. *DualDICE: Behavior-Agnostic Estimation of Discounted Stationary Distribution Corrections*. 2019. arXiv: 1906.04733 [cs.LG].

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

[RZMJR21] Paria Rashidinejad, Banghua Zhu, Cong Ma, Jiantao Jiao, and Stuart Russell. "Bridging Offline Reinforcement Learning and Imitation Learning: A Tale of Pessimism". In: *arXiv preprint arXiv:2103.12021* (2021).

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[TET12] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

[UHJ20] Masatoshi Uehara, Jiawei Huang, and Nan Jiang. "Minimax weight and q-function learning for off-policy evaluation". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 9659–9668.

[UIJKSX21] Masatoshi Uehara, Masaaki Imaizumi, Nan Jiang, Nathan Kallus, Wen Sun, and Tengyang Xie. "Finite sample analysis of minimax offline reinforcement learning: Completeness, fast rates and first-order efficiency". In: *arXiv preprint arXiv:2102.02981* (2021).

[VLJY19] Cameron Voloshin, Hoang M Le, Nan Jiang, and Yisong Yue. "Empirical study of off-policy policy evaluation for reinforcement learning". In: *arXiv preprint arXiv:1911.06854* (2019).

[XJ20] Tengyang Xie and Nan Jiang. *Batch Value-function Approximation with Only Realizability*. 2020. arXiv: 2008.04990 [cs.LG].

[YDNTS20] Mengjiao Yang, Bo Dai, Ofir Nachum, George Tucker, and Dale Schuurmans. "Offline Policy Selection under Uncertainty". In: *arXiv preprint arXiv:2012.06919* (2020).

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [No] We provide complete proofs of Proposition 3 and Theorem 4, which are new results. Earlier propositions are simplifications of existing theoretical results to help readers understand the theoretical intuitions, and we provide proof sketches and point the readers to the relevant literature for the full versions of the theorems and proofs.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] We briefly mention computational considerations of our methods in Section 6. Also, a large part of the computational resources go into the training algorithms, which produces the candidate models our algorithms will select from and is part of the experiment setup, which we did not keep track of.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]