Revisiting Bellman Errors for Offline Model Selection

Anonymous Author(s)
Affiliation
Address
email

Abstract

It is well-known that the empirical Bellman errors are poor predictors of value function estimation accuracy and policy performance. This has led researchers to abandon offline model selection procedures based on Bellman errors and instead focus on directly estimating the expected return under different policies of interest. The problem with this approach is that it can be very difficult to use an offline dataset generated by one policy to estimate the expected returns of a different policy. In contrast, we argue that Bellman errors can be useful for offline model selection, and that the discouraging results in past literature has been due to estimating and utilizing them incorrectly. We propose a new algorithm, *Supervised Bellman Validation*, that estimates the expected squared Bellman error better than the empirical Bellman errors. We demonstrate the relative merits of our method over competing methods through both theoretical results and empirical results on offline datasets from the Atari benchmark. We hope that our results will challenge current attitudes and spur future research into Bellman errors and their utility in offline model selection.

1 Introduction

Traditionally, reinforcement learning (RL) has assumed access to an online environment or high-fidelity simulator with which to perform repeated interactions [25]. In contrast, offline RL [1, 14, 16] focuses on training an agent solely from a fixed dataset of observed transitions. By not requiring any further online interactions, offline RL algorithms can significantly improve real-world applicability of RL to settings such as autonomous driving [30] and healthcare [23], where large-scale active data collection may be expensive or unsafe but large data banks of previously-logged interactions are still available.

There has been a recent surge of RL algorithms that can train an agent offline with varying success [6, 10]. Most such algorithms rely on online environment interactions to tune their own hyperparameters. However, applying offline RL algorithms in real-world settings necessitates tuning their hyperparameters offline as well (i.e. offline model selection). Prevalent offline model selection approaches are typically based on offline policy evaluation (OPE) algorithms that estimate the expected returns under a given policy [19, 26, 31]. Unfortunately, accurately estimating the expected returns under one policy using data generated by a completely different behavioral policy can be very difficult. Other approaches that have been explored have been based on empirical Bellman errors as an approximation for the true Bellman errors [8, 12, 19]. These works have found empirical Bellman errors to be poor predictors of value model accuracy, leading to a current belief that Bellman errors are not useful in offline model selection [5, 9, 19].

Instead, our work argues that empirical Bellman errors have been found unhelpful in model selection because they are poor proxies for the true Bellman errors. We propose a new algorithm, *Supervised Bellman Validation (SBV)*, that uses supervised learning to derive a Bellman error-based validation loss function and provide a better proxy for the true Bellman errors. We argue several theoretical benefits of SBV over competing methods and apply SBV to offline Atari datasets [1, 10] containing

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

high-dimensional image observations. We explore conditions in which approaches based on Bellman errors can work well for model selection, and demonstrate that SBV performs competitively when these conditions are satisfied. Finally, we discuss implications of our results and suggest potential avenues for future work.

2 Preliminaries

In *offline reinforcement learning (RL)* [14, 16], we have a static dataset $\mathcal{D} = \{(s, a, r, s')\}$ consisting of $|\mathcal{D}|$ transitions (s, a, r, s') where $r \in \mathbb{R}$ is the immediate reward observed after state $s \in \mathcal{S}$ was observed and action $a \in \mathcal{A}$ was taken, before observing the new state $s' \in \mathcal{S}$. We assume the data comes from a Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, R, d_0, \gamma)$ [21] with state space $\mathcal{S}$, action space $\mathcal{A}$, transition probabilities $T(s'|a, s) = \Pr(S_{t+1} = s'|A_t = a, S_t = s)$, reward function $R(S_t, A_t, S_{t+1}) = R_t$, initial state probabilities $d_0(s) = \Pr(S_0 = s)$, and discount factor $\gamma \in (0, 1)$. We can think of the observed state-action pairs in $\mathcal{D}$ as being sampled from (or being identically distributed as) $P^{\mu}_{SA}(s, a) = d^{\mu}(s)\mu(a|s)$ where μ is the *behavioral policy* and d^{μ} is the marginal distribution of states over time points induced by policy μ and MDP $\mathcal{M}$. Likewise, we can think of the observed transitions (s, a, r, s') in $\mathcal{D}$ as being sampled from (or being identically distributed as) $P^{\mu}_{T}(s, a, r, s') = d^{\mu}(s)\mu(a|s)T(s'|a, s)$ where $r = R(s, a, s')$. If $\mathcal{D}$ was generated by a single stationary policy, μ is simply equal to this policy. On the other hand, if $\mathcal{D}$ consists of the intermediate transitions observed from training an online RL algorithm, $\mathcal{D}$ can be thought of as being generated from many different policies and μ becomes a mixture distribution of those policies.

Define the *action-value function* for policy π as $Q^{\pi}(s, a) = \mathbb{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t R_t | S_0 = s, A_0 = a]$ where $\mathbb{E}_{\pi}$ denotes expectation over MDP $\mathcal{M}$ and policy π. The optimal policy π^* is the policy whose action-value function equals the optimal action-value function $Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a)$. For any real-valued function of state-action pairs Q (oftentimes called a *Q-function*), the *Bellman operator* $\mathcal{B}^*$ satisfies:

$$\mathcal{B}^* Q(s, a) = \mathbb{E}[R_t + \gamma \max_{a' \in \mathcal{A}} Q(S_{t+1}, a') | S_t = s, A_t = a] \tag{1}$$

It can be shown that $\pi^* = \pi_{Q^*}$, where $\pi_Q(s) = \text{argmax}_a Q(s, a)$ is the greedy policy of Q-function Q, and that $Q = \mathcal{B}^* Q$ if and only if $Q = Q^*$ [25].

For any Q-function Q, density P of state-action pairs and dataset of transitions $\mathcal{S} = \{(s, a, r, s')\}$, let $||Q||^2_P = \mathbb{E}_{(s,a)\sim P}[Q(s, a)^2]$ and $||Q||^2_{\mathcal{S}} = |\mathcal{S}|^{-1} \sum_{(s,a)\in\mathcal{S}} [Q(s, a)^2]$. Define the *Bellman backup function* for Q-function Q as $(\mathcal{B}^* Q)(s, a)$ and its *Bellman error function* as $(Q - \mathcal{B}^* Q)(s, a)$ (see Equation 1). Likewise, define the *empirical Bellman backups* for Q-function Q and dataset $\mathcal{S}$ as $(\mathcal{B}_{\mathcal{S}} Q)(s, a) = r + \gamma \max_{a'} Q(s', a'),\ (s, a, r, s') \in \mathcal{S}$ and their *empirical Bellman errors* as $Q(s, a) - (\mathcal{B}_{\mathcal{S}} Q)(s, a),\ (s, a) \in \mathcal{S}$. Q-learning algorithms update a working model based on its empirical Bellman backups so as to approximate Q^* [18, 29]. Throughout we assume all estimated value functions and policies are fitted by Q-learning algorithms.

3 Related Work

In the *offline model selection* problem, we have a set of candidate value models $\mathcal{Q} = \{Q_1, ..., Q_M\}$, or estimators for Q^*, and our goal is to choose the "best" among them based on some criterion. Here each candidate in $\mathcal{Q}$ was estimated via a Q-learning algorithm with a different hyperparameter configuration. For example, candidates in $\mathcal{Q}$ could include Q-functions obtained from running a single DQN algorithm and stopping at different numbers of training iterations, or it could include Q-functions obtained from running DQN with different Q-network architectures. The most popular approach for offline model selection is to use an *offline policy evaluation (OPE)* algorithm, which estimates the marginal expectation of returns $J(\pi) = \mathbb{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t R_t]$ under policies of interest $\pi \in \{\pi_{Q_1}, \pi_{Q_2}, ..., \pi_{Q_M}\}$ [7]. For example, *importance sampling* estimators directly estimate $J(\pi)$ from $\mathcal{D}$ by using importance weights to adjust for the distribution shift [20, 26]. *Fitted Q-evaluation* indirectly estimates $J(\pi)$ by estimating Q^{π} using another off-policy RL algorithm [15, 19]. Finally, *model-based* approaches indirectly estimates $J(\pi)$ by estimating the transition kernel $T(\cdot|a, s)$ for all $(a, s) \in \mathcal{A} \times \mathcal{S}$ using standard density estimation techniques [4, 24, 31].

Existing OPE approaches often have difficulties with accurately estimating $J(\pi)$. In particular, importance sampling usually has prohibitively-large estimation variance, FQE introduces its own hyperparameters that cannot be easily tuned offline and model-based approaches can have great difficulty modelling the entire underlying MDP in complex and high-dimensional settings. For these reasons, all of these methods have achieved highly sub-optimal performance on certain environments

and most of them have not even been attempted on environments with pixel-valued states [7]. In contrast to estimating the expected returns associated with the greedy policies implicit in $\mathcal{Q}$, another approach is to assess candidates $Q_m, 1 \leq m \leq M$ via their *mean squared Bellman error (MSBE)*:

$$||Q_m - \mathcal{B}^* Q_m||^2_{P^\mu_{SA}} = \mathbb{E}_{(s,a)\sim P^\mu_{SA}} \left[\left(Q_m(s,a) - (\mathcal{B}^* Q_m)(s,a)\right)^2 \right] \tag{2}$$

We discuss theoretical justifications for using the Bellman error for model selection in section 5. To estimate the true MSBE (Equation 2), previous literature has used the *empirical mean square Bellman error (EMSBE)* [8, 12, 19]:

$$|Q_m - \mathcal{B}_\mathcal{D} Q_m||^2_\mathcal{D} = \frac{1}{|\mathcal{D}|} \sum_{(s,a,r,s')\in\mathcal{D}} \left[\left(r + \max_{a'\in\mathcal{A}} Q_m(s',a') - Q_m(s,a) \right)^2 \right] \tag{3}$$

The problem is that the EMSBE (Equation 3) is a **biased** estimator of the true MSBE and its minimizer can be an arbitrarily poor estimator of Q^* even in the asymptotic case (see section 5 for details). It is likely for this reason that previous literature has found this to be ineffective at hyperparameter tuning [8, 12].

The work by Farahmand and Szepesvari [5] is more related to our approach in that they propose supervised learning to approximate the true Bellman errors. There are however two major distinctions of Farahmand and Szepesvari [5] over our work: 1) their algorithm is much more sample inefficient as they require partitioning the data into thirds as well as tight probabilistic upper bounds on the *excess risk* of the estimates of $\mathcal{B}^* Q_m$ [27]; 2) they did not investigate the empirical performance of their algorithm or of the MSBE more generally.

4 Supervised Bellman Validation

To understand *Supervised Bellman Validation (SBV)*, consider first the case where $Q^*(s,a)$ is actually **known** for state-action pairs $(s,a) \in \mathcal{D}$ and we wish to select the candidates $Q_m \in \mathcal{Q}$ that best estimate Q^*. An obvious criterion in this case would be the *value function estimation mean squared error (MSE)*:

$$||Q^* - Q_m||^2_{P^\mu_{SA}} = \mathbb{E}_{(s,a)\sim P^\mu_{SA}} \left[\left(Q^*(s,a) - Q_m(s,a)\right)^2 \right]. \tag{4}$$

While P^μ_{SA} is unknown, we can estimate the associated expectation by randomly partition 80% of the trajectories present in $\mathcal{D}$ into a training set $\mathcal{D}_T$ and reserving the remaining 20% of trajectories as a validation set $\mathcal{D}_V$. We can then generate candidates $\mathcal{Q} = \{Q_1, ..., Q_M\}$ by running deep Q-learning algorithms on $\mathcal{D}_T$ with M different hyperparameter configurations, and use $\mathcal{D}_V$ to estimate the value function estimation MSE (Equation 4) as $||Q^* - Q_m||^2_{\mathcal{D}_V}$, for each $1 \leq m \leq M$.

In real-world RL, the targets $Q^*(s,a), (s,a) \in \mathcal{D}$ are not known: this is what separates supervised learning from offline RL. Instead of a criterion based on Equation 4, SBV uses a surrogate criterion based on the MSBE $||Q_m - \mathcal{B}^* Q_m||^2_{P^\mu_{SA}}$ (Equation 2). The relationship between estimation error and Bellman error is discussed more in section 5. Similar to what we would do in supervised learning, SBV creates a training set $\mathcal{D}_T$ and a validation set $\mathcal{D}_V$ by randomly partitioning trajectories from $\mathcal{D}$. SBV then trains M Q-functions $\mathcal{Q} = \{Q_1, ..., Q_M\}$ on $\mathcal{D}_T$. Note that the MSBE actually contains **two** unknown quantities: the population density P^μ_{SA}, and the M Bellman backup functions $\mathcal{B}^* Q_m, 1 \leq m \leq M$. From Equation 1, we see that each $(\mathcal{B}^* Q_m)(s,a)$ is just a conditional expectation. Therefore, the M Bellman backup functions can be estimated by running M regression algorithms on $\mathcal{D}_T$, with the mth such algorithm estimating $\mathcal{B}^* Q_m$ by fitting a function f to minimize the *Bellman backup MSE*:

$$||\mathcal{B}_{\mathcal{D}_T} Q_m - f||^2_{\mathcal{D}_T} = \frac{1}{|\mathcal{D}_T|} \sum_{(s,a,r,s')\in\mathcal{D}_T} \left[\left(r + \gamma \max_{a'} Q_m(s',a') - f(s,a) \right)^2 \right]. \tag{5}$$

Denote the fitted models from our regression algorithms as $\widehat{\mathcal{B}}^* Q_1, ..., \widehat{\mathcal{B}}^* Q_M$. The MSBE for each candidate $Q_m \in \mathcal{Q}$ can then be estimated as:

$$||Q_m - \widehat{\mathcal{B}}^* Q_m||^2_{\mathcal{D}_V} = \frac{1}{|\mathcal{D}_V|} \sum_{(s,a)\in\mathcal{D}_V} \left[\left(Q_m(s,a) - (\widehat{\mathcal{B}}^* Q_m)(s,a) \right)^2 \right]. \tag{6}$$

Our method is summarized in Algorithm 1. Here a hyperparameter configuration H_m specifies all relevant hyperparameters of the Q-learning algorithm, such as the specific training algorithm (e.g.

Algorithm 1: Supervised Bellman Validation (SBV)

Input: Set of observed transitions $\mathcal{D} = \{(s, a, r, s')\}$, hyperparameter configurations to evaluate $\mathcal{H} = \{H_1, ..., H_m\}$
Randomly partition trajectories in $\mathcal{D}$ to training set $\mathcal{D}_T$ and validation set $\mathcal{D}_V$
for *configuration* $m \in \{1, ..., M\}$ **do**
  Estimate Q^* as Q_m by running a Q-learning algorithm on $\mathcal{D}_T$ with configuration H_m
  Estimate $\mathcal{B}^* Q_m$ as $\widehat{\mathcal{B}}^* Q_m$ by minimizing $||\mathcal{B}_{\mathcal{D}_T} Q_m - f||^2_{\mathcal{D}_T}$ wrt f (see Equation 5)
  Estimate the MSBE of Q_m as $||Q_m - \widehat{\mathcal{B}}^* Q_m||^2_{\mathcal{D}_V}$ (see Equation 6).
end
Output Q_{m^*} as our estimate of Q^* where $m^* = \text{argmin}_{1 \leq m \leq M} ||Q_m - \widehat{\mathcal{B}}^* Q_m||^2_{\mathcal{D}_V}$

double DQN [11] or duelling DQN [28]), the Q-network architecture and the number of training iterations.

To instantiate SBV, Algorithm 2 provides a practical implementation of SBV for tuning the number of training iterations used by DQN. Here the *Q-Network* is a neural network Q-function Q_θ with trainable parameters θ that is trained by DQN to approximate Q^*, and $Q_{\theta_{k+1}}$ denotes the trained Q-Network after $k+1$ iterations of the DQN algorithm. Moreover, the *Bellman Network* is a neural network model $\mathcal{B}_\phi$ with trainable parameters ϕ that is updated by SBV after every DQN training iteration to approximate $\mathcal{B}^* Q_\theta$ for the current Q-Network Q_θ. We let $\mathcal{B}_{\phi_{k+1}}$ denote the Bellman network updated by SBV in order to approximate $\mathcal{B}^* Q_{\theta_{k+1}}$.

While the Q-network hyperparameters can be tuned via SBV, the Bellman network itself has hyperparameters which must be tuned. As the Bellman network is solving a series of supervised learning problems, the relevant hyperparameters can easily be tuned offline by minimizing validation Bellman backup MSE on $\mathcal{D}_V$ across the various sets of targets. The ability of SBV to fully tune its own hyperparameters offline gives it real-world applicability and sets it apart from other approaches like fitted Q-evaluation, whose hyperparameters have traditionally been chosen based on online evaluations or previous literature on the environments of interest [19].

5 Theoretical Results

We begin by introducing some novel theoretical results to support our algorithm, the proofs of which can be found in section A of the Supplementary Material. Our first theorem, given below, implies that a candidate value model Q_m with sufficiently small Bellman errors should be sufficiently accurate and have a greedy policy that gives sufficiently high return. Moreover, if the Bellman errors $Q_m - \mathcal{B}^* Q_m$ are unknown but are estimated with sufficient accuracy, the estimated Bellman errors will still be informative in lower-bounding the estimation accuracy and policy value.

Theorem 1. *Let $\widehat{\mathcal{B}}^* Q_m$ be an estimate of $\mathcal{B}^* Q_m$ and assume that $||Q_m - \widehat{\mathcal{B}}^* Q_m||_\infty \leq \epsilon$ and $||\widehat{\mathcal{B}}^* Q_m - \mathcal{B}^* Q_m||_\infty \leq \delta$. Then $||Q - Q^*||_\infty \leq \frac{1}{1-\gamma}(\epsilon + \delta)$ and $||V^{\pi^*} - V^{\pi_Q}||_\infty \leq \frac{2}{(1-\gamma)^2}(\epsilon + \delta)$ where $V^\pi(s) = \mathbb{E}_{a \sim \pi(\cdot|s)} [Q^\pi(s, a)]$.*

Algorithm 2: Applying Early Stopping to DQN with SBV

Input: Set of observed transitions $\mathcal{D} = \{(s, a, r, s')\}$
Randomly partition trajectories in $\mathcal{D}$ to training set $\mathcal{D}_T$ and validation set $\mathcal{D}_V$
Initialize deep Q-network Q_{θ_0} with trainable parameters θ_0 and Bellman network $\mathcal{B}_{\phi_0}$ with trainable parameters ϕ_0
for *iteration* $k \in \{0, ..., K-1\}$ **do**
  Update θ_{k+1} from θ_k by running DQN on $\mathcal{D}_T$ for one iteration
  Update ϕ_{k+1} from ϕ_k via gradient descent to minimize $||\mathcal{B}_{\mathcal{D}_T} Q_{\theta_{k+1}} - \mathcal{B}_\phi||^2_{\mathcal{D}_T}$ wrt ϕ (see 5)
  Estimate of the MSBE of $Q_{\theta_{k+1}}$ as $||Q_{\theta_{k+1}} - \mathcal{B}_{\phi_{k+1}}||^2_{\mathcal{D}_V}$ (see 6)
end
Estimate Q^* as Q_{k^*} and π^* as $\pi_{Q_{k^*}}$ where $k^* = \text{argmin}_{1 \leq k \leq K} ||Q_{\theta_k} - \mathcal{B}_{\phi_k}||^2_{\mathcal{D}_V}$

Let F^μ be the cumulative distribution function (CDF) associated with P_T^μ and let $F^\mathcal{D}$ be the *empirical distribution function (EDF)* of transitions associated with $\mathcal{D}$ [13]. Under general conditions, $F^\mathcal{D}$ converges to F^μ as $|\mathcal{D}|$ grows large. To study the bias and asymptotic properties of the EMSBE (Equation 3) and SBV (Algorithm 1), we focus on the setting where $|\mathcal{D}| = \infty$ and $F^\mathcal{D} = F^\mu$. Theorem 2 states that the EMSBE has non-negligible bias and that its minimum need not be close to Q^* even with infinite samples. In contrast, Theorem 3 states that SBV correctly recovers the true MSBE and correctly selects the true optimal action-value function Q^*, at least in the asymptotic case. These theorems help explain why SBV performs well empirically, both overall and relative to the EMSBE (see section 6).

Theorem 2. *Assume that $F^\mathcal{D} = F^\mu$. Then $\text{argmin}_{Q:\mathcal{S}\times\mathcal{A}\to\mathbb{R}} ||Q - \mathcal{B}_\mathcal{D} Q||^2_\mathcal{D} \neq Q^*$ in general and:*

$$||Q_m - \mathcal{B}_\mathcal{D} Q_m||^2_\mathcal{D} - ||Q_m - \mathcal{B}^* Q_m||^2_{P^\mu_{SA}} = \mathbb{E}_{(S_t,A_t)\sim P^\mu_{SA}} \left\{ Var\left[R_t + \max_{a'\in\mathcal{A}} Q_m(S_{t+1}, a') | S_t, A_t \right] \right\}$$

.

Theorem 3. *Assume the following: (1) $F^\mathcal{D} = F^\mu$; (2) $\mathcal{D}_T$ and $\mathcal{D}_V$ partition the trajectories present in $\mathcal{D}$; (3) $\mathcal{B}^* Q_m$ is estimated as $\widehat{\mathcal{B}}^* Q_m = \text{argmin}_{f:\mathcal{S}\times\mathcal{A}\to\mathbb{R}} ||\mathcal{B}_{\mathcal{D}_T} Q_m - f||^2_{\mathcal{D}_T}$. Then under mild technical conditions and with probability one, $\widehat{\mathcal{B}}^* Q_m = \mathcal{B}^* Q_m$, $||Q_m - \widehat{\mathcal{B}}^* Q_m||^2_{\mathcal{D}_V} = ||Q_m - \mathcal{B}^* Q_m||^2_{P^\mu_{SA}}$ and $||Q_m - \widehat{\mathcal{B}}^* Q_m||^2_{\mathcal{D}_V} = 0$ if and only if $Q_m = Q^*$.*

6 Experiments

We evaluated Supervised Bellman Validation (SBV; Algorithm 1) on four offline DQN-Replay datasets [1] associated with the following Atari games [2]: Pong, Breakout, Asterix and Seaquest. Atari environments are widely studied in deep RL and possess high-dimensional states, stochastic nonlinear transition dynamics and long-term consequences from taking particular actions. They also possess discrete action spaces amenable to Q-learning algorithms. These properties makes Atari environments challenging and applicable candidates for benchmarking real-world performance of SBV. Following [1], we uniformly sub-sampled 25% of our DQN-replay datasets to obtain four training datasets with 10M transitions and four associated validation datasets with 2.5M transitions. For each training dataset, we evaluate a Q-network after each training iteration across two deep Q-learning configurations, which we dub DQN (S) and DDQN (L). DQN (S) uses the standard DQN architecture [18] and the Adam optimizer following Agarwal et al. [1]. The DDQN (L) configuration is described in section B of the Supplementary Material, and involves a deeper network architecture and double Q-learning target updates [11]. Each training configuration was run for 50 iterations, resulting in 100 candidate Q-networks to evaluate for each training dataset.

We implemented SBV in a manner similar to Algorithm 2 to improve computational efficiency. Moreover, the Bellman network training configuration was tuned offline so as to minimize validation error across the 400 evaluated Q-functions and four Atari games, as discussed in section 4. To demonstrate the efficacy of SBV, we compare it to EMSBE (Equation 3) and *weighted per-decision importance sampling (WIS)* [20]. Full implementation details of SBV and WSE can be found in section C of the Supplementary Material. Due to the computational demands of implementing Fitted Q-evaluation (FQE) and the inability to tune its hyperparameters offline, we have omitted results for FQE for now. We also do not implement model-based approaches here: accurately estimating the underlying MDP of Atari environments offline would be completely novel and extremely challenging, warranting a paper in and of itself.

In Table 1, we calculate the mean return of the top-5 policies according to each method for each environment, and standardize the mean returns to a $[0, 1]$ range. We can see here that SBV consistently performs better than competing methods with respect to top-5 policy returns. EMSBE does not perform as well as SBV, but it still performs better than WIS.

In Figure 1, we plot learning curves for the best configuration for each environment, and indicate the iteration where training was stopped by each method. We can see that SBV performs significantly better than competing methods on average. Moreover, only SBV performs as well or better than applying no early stopping across all environments.

7 Discussion and Future Work

Our theoretical and empirical results challenge the current belief in the RL community that Bellman errors are not helpful for offline model selection. In addition to the Atari results described in section 6,

Table 1: Standardized Top-5 Policy Mean Returns. A mean return of 0% implies that the method choose the worst five policies possible on the given dataset (based on online evaluations), while a mean return of 100% implies that the method choose the best five policies possible.

Method	Pong	Breakout	Asterix	Seaquest
WIS [20]	41%	40%	50%	13%
EMSBE (Equation 3)	77%	43%	51%	44%
SBV (Ours)	**94%**	**79%**	**76%**	**65%**

we also plan to apply SBV to offline RL problems in physics and healthcare involving much smaller offline datasets [17, 22] to further demonstrate the real-world applicability and robust performance of our method. SBV only requires learning a univariate regression function, making it 1) more statistically efficient than either offline RL or model-based learning; 2) straightforward to tune SBV's own hyperparameters to the dataset at hand; and 3) less biased and more sample-efficient than using no function approximation whatsoever. These features make SBV a competitive candidate for offline model selection.

Our empirical results on Atari games suggest that Bellman error-based model selection can perform well in real-world settings provided certain conditions are met. For instance, model selection algorithms based on the MSBE will not perform well if they do not accurately estimate the MSBE. For this reason, the EMSBE is an ineffective model selector and SBV performance is dependent on estimating $\mathcal{B}^* Q_m$ with sufficient accuracy for all $1 \leq m \leq M$. In the case of our experiments, tuning the Bellman network to the datasets at hand so as to minimize validation MSE was a critical component in achieving adequate performance with SBV. Model selection algorithms such as SBV that estimate the MSBE also fail if the MSBE itself is a poor model selector. For example, for the MSBE to perform well as a model selector, we require there to be sufficient diversity in the observed state-action pairs (see the proof of Theorem 3 given in section A of the Supplementary Material to better understand the importance of this condition).

Finally, Theorem 1 and empirical investigations detailed in section D of the Supplementary Material suggest that the MSBE provides a tight lower bound on estimation error and a loose lower bound on policy value. Consequently, the lower the MSBE value, the more informative it will be in terms of policy performance. For example, if $J(\pi^*) = \mathbb{E}_{\pi^*}[\sum_{t \geq 0} \gamma^t R_t] = 10$, value models with an MSBE of 0.1 may have universally high estimation error but associated policy values in the wide range of $[1, 10]$. On the other hand, value models with an MSBE of only 0.01 would likely have policy values residing in a much tighter range, such as $[9, 10]$ (they will also all have much lower estimation error). The MSBE will thus be an effective model selector when the set of candidate value functions $\mathcal{Q}$ include value functions that have sufficiently small Bellman errors. In real-world settings, we can ensure that our candidate set $\mathcal{Q}$ includes estimates with sufficiently small MSBE by exploring a sufficiently large number of training configurations and/or by exploring training configurations which have been shown in previous work to achieve robust performance in a variety of scenarios.

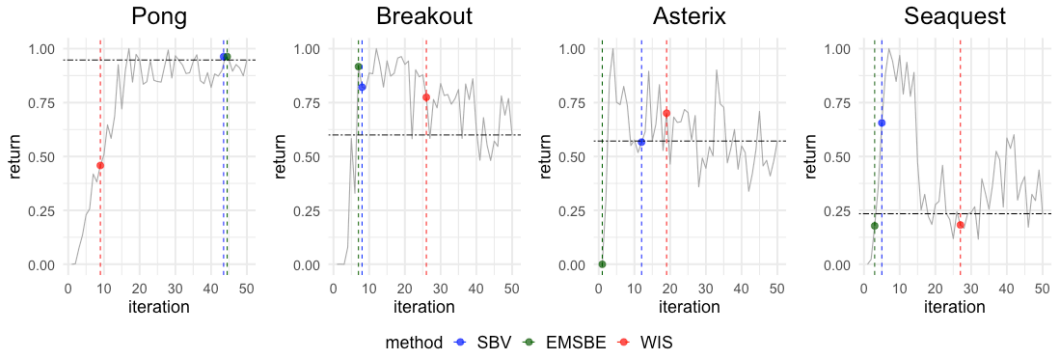

Figure 1: Learning Curves for the Best Configuration. Returns are standardized to a $[0, 1]$ range with zero being the lowest return observed for a given run and one being the highest return observed. The dashed horizontal line represents performance when no early stopping is applied. The vertical lines represent the iterations where training was stopped according to different methods.

Our work opens up several avenues for future research. For example, one avenue for future work is to investigate how to achieve sufficient performance with SBV while reducing computational demands. Reducing the number of configurations we have to explore before achieving sufficient performance with SBV would be helpful here, as would modifying the Bellman network training configuration to improve computational efficiency. Moreover, our results focus on discrete control problems and Q-learning algorithms. In contrast, continuous control problems and deep actor-critic algorithms require two neural network models: a Q-network Q_θ and a *policy model* π_α. suggests that a $\pi_\alpha \approx \pi^*$, the γ-contraction properties of the Bellman operator and policy iteration theory [3] suggests that we would need to ensure that both $Q_\theta \approx \mathcal{B}^{\pi_\alpha} Q_\theta$ and that $\pi_\alpha \approx \pi_{Q_\theta}$. While SBV could help ensure that the first condition is satisfied, extending SBV to deal with the second condition is left for future work. Finally, recall that a major shortcoming of FQE is that it has its own hyperparameters that cannot be easily tuned offline. in theory, one can use SBV to tune the hyperparameters of FQE and then use FQE to tune the hyperparameters of deep Q-learning or deep actor-critic. The main issue here is the computational demand: both SBV and FQE are already computationally demanding procedures in their own right, and combining them in a computationally feasible manner is nontrivial.

References

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

[2] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013. doi: 10.1613/jair.3912.

[3] Lucian Busoniu, Robert Babuska, Bart De Schutter, and Damien Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, 2010.

[4] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NeurIPS*, 2018.

[5] Amir Massoud Farahmand and Csaba Szepesvari. Model selection in reinforcement learning. *Machine Learning*, 85:299–332, 2010.

[6] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[7] Justin Fu, Mohammad Norouzi, Ofir Nachum, G. Tucker, Ziyun Wang, Alexander Novikov, Mengjiao Yang, Michael R. Zhang, Yutian Chen, Aviral Kumar, Cosmin Paduraru, Sergey Levine, and Tom Le Paine. Benchmarks for deep off-policy evaluation. In *ICLR*, 2021.

[8] Scott Fujimoto, David Meger, Doina Precup, Ofir Nachum, and Shixiang Shane Gu. Why should i trust you, bellman? the bellman error is a poor replacement for value error. *ArXiV*, abs/2201.12417, 2022.

[9] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow*. O'Reilly, second edition, 2019.

[10] Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Thomas Paine, Sergio Gómez, Konrad Zolna, Rishabh Agarwal, Josh S Merel, Daniel J Mankowitz, Cosmin Paduraru, et al. Rl unplugged: A suite of benchmarks for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:7248–7259, 2020.

[11] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, page 2094–2100, 2016.

[12] Alex Irpan, Kanishka Rao, Konstantinos Bousmalis, Chris Harris, Julian Ibarz, and Sergey Levine. Off-policy evaluation via off-policy classification. In *NeurIPS*, 2019.

[13] Samuel Kotz, Campbell B. Read, N. Balakrishnan, Brani Vidakovic, and Norman L. Johnson. *Encyclopedia of Statistical Sciences*. John Wiley & Sons, 2006.

[14] Michail G. Lagoudakis and Ronald E. Parr. Least-squares policy iteration. *J. Mach. Learn. Res.*, 4:1107–1149, 2003.

[15] Hoang Minh Le, Cameron Voloshin, and Yisong Yue. Batch policy learning under constraints. In *ICML*, 2019.

[16] Sergey Levine, Aviral Kumar, G. Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *ArXiv*, abs/2005.01643, 2020.

[17] Daniel J. Luckett, Eric B. Laber, Anna R. Kahkoska, David M. Maahs, Elizabeth J. Mayer-Davis, and Michael R. Kosorok. Estimating dynamic treatment regimes in mobile health using v-learning. *Journal of the American Statistical Association*, 115:692–706, 2020.

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[19] Tom Le Paine, Cosmin Paduraru, Andrea Michi, Caglar Gulcehre, Konrad Zolna, Alexander Novikov, Ziyun Wang, and Nando de Freitas. Hyperparameter selection for offline reinforcement learning. *ArXiv*, abs/2007.09055, 2020.

[20] Doina Precup, Richard S. Sutton, and Satinder Singh. Eligibility traces for off-policy policy evaluation. In *ICML*, 2000.

[21] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[22] Jette Randløv and Preben Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. In *ICML*, 1998.

[23] Susan M. Shortreed, Eric B. Laber, Daniel J. Lizotte, T. Scott Stroup, Joelle Pineau, and Susan A. Murphy. Informing sequential clinical decision-making through reinforcement learning: an empirical study. *Machine Learning*, 84:109–136, 2010.

[24] Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *ICML*, 1990.

[25] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[26] Philip S. Thomas and Emma Brunskill. Data-efficient off-policy policy evaluation for reinforcement learning. In *ICML*, 2016.

[27] Vladimir N Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

[28] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. In *ICML*, 2016.

[29] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992. doi: 10.1007/bf00992698.

[30] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving video database with scalable annotation tooling. *ArXiv*, abs/1805.04687, 2018.

[31] Michael R Zhang, Thomas Paine, Ofir Nachum, Cosmin Paduraru, George Tucker, ziyu wang, and Mohammad Norouzi. Autoregressive dynamics models for offline policy evaluation and optimization. In *ICLR*, 2021.