Diagnosing Bottlenecks in Deep Q-learning Algorithms

Justin Fu [*1] **Aviral Kumar** [*1] **Matthew Soh** [1] **Sergey Levine** [1]

Abstract

Q-learning methods are a common class of algorithms used in reinforcement learning (RL). However, their behavior with function approximation, especially with neural networks, is poorly understood theoretically and empirically. In this work, we aim to experimentally investigate potential issues in Q-learning, by means of a ”unit testing” framework where we can utilize oracles to disentangle sources of error. Specifically, we investigate questions related to function approximation, sampling error and nonstationarity, and where available, verify if trends found in oracle settings hold true with deep RL methods. We find that large neural network architectures have many benefits with regards to learning stability; offer several practical compensations for overfitting; and develop a novel sampling method based on explicitly compensating for function approximation error that yields fair improvement on high-dimensional continuous control domains.

1. Introduction

Q-learning algorithms, which are based on approximating state-action value functions, are an efficient and commonly used class of RL methods. Q-learning methods have several very appealing properties: they are relatively sample-efficient when compared to policy gradient methods, and they allow for off-policy learning. This makes them an appealing choice for a wide range of tasks, from robotic control (Kalashnikov et al., 2018) and video game AI (Mnih et al., 2015) to off-policy learning from historical data for recommender systems (Shani et al., 2005). However, although the basic tabular Q-learning algorithm is convergent and admits theoretical analysis (Sutton & Barto, 2018), its counterpart with function approximation is poorly understood. In this paper, we aim to investigate the degree to which potential issues with Q-learning manifest in practice. We empirically analyze aspects of the Q-learning method in a *unit testing* framework, where we can employ oracle solvers to obtain ground truth Q-functions and distributions for exact analysis. We investigate the following questions:

1) What is the effect of function approximation on convergence? Many practical RL problems require function approximation to handle large or continuous state spaces. However, the behavior of Q-learning methods under function approximation is not well understood – there are simple counterexamples where the method diverges (Baird, 1995). To investigate these problems, we study the convergence behavior of Q-learning methods with function approximation by varying the function approximator power and analyzing the quality of the solution found. We find, somewhat surprisingly, that divergence rarely occurs, and that function approximation error is not a major problem in Q-learning algorithms when the function approximator is powerful. This makes sense in light of the theory: a high-capacity approximator can perform an accurate projection of the bellman Backup, thus mitigating potential convergence issues due to function approximation. (Section 4)

2) What is the effect of sampling error and overfitting? RL is used to solve problems where we do not have access to the transition function of the MDP. Thus, Q-learning methods need to learn by collecting samples in the environment, and minimizing errors on samples potentially leads to overfitting. We experimentally show that overfitting exists in practice by performing ablation studies on the number of gradient steps, and by demonstrating that oracle based early stopping techniques can be used to improve performance of Q-learning algorithms. (Section 5).

3) What is the effect of distribution shift and a moving target? The standard formulation of Q-learning prescribes an update rule, with no corresponding objective function (Sutton et al., 2009a). This results in a process which optimizes an objective that is non-stationary in two ways: the target values are updated during training (the *moving target* problem), and the distribution under which the Bellman error is optimized changes, as samples are drawn from different policies (the *distribution shift* problem). These properties can make convergence behavior difficult to understand, and prior works have hypothesized

[*]Equal contribution [1]UC Berkeley. Correspondence to: Justin Fu <justinjfu@eecs.berkeley.edu>, Aviral Kumar <aviralk@berkeley.edu>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

that nonstationarity is a source of instability (Mnih et al., 2015; Lillicrap et al., 2015). We develop metrics to quantify the amount of distribution shift and performance change due to non-stationary targets. Surprisingly, we find that in a controlled experiment, distributional shift and non-stationary targets do not correlate with a reduction in performance, and some well-performing methods incur high distribution shift.

4) What is the best sampling or weighting distribution? Deeply tied to the distribution shift problem is the choice of which distribution to sample from. Do moving distributions cause instability, as Q-values trained on one distribution are evaluated under another in subsequent iterations? Researchers have often noted that on-policy samples are typically superior to off-policy samples (Sutton & Barto, 2018), and there are several theoretical results that highlight favorable convergence properties under on-policy samples. However, there is little theoretical guidance on how to pick distributions so as to maximize learning. To this end, we investigate several choices for the sampling distribution. Surprisingly, we find that on-policy training distributions are not always preferable, and that broader, higher-entropy distributions often perform better, regardless of distributional shift. Motivated by our findings, we propose a novel weighting distribution, adversarial feature matching (AFM), which is explicitly compensates for function approximator error, while still producing high-entropy sampling distributions.

In summary, we introduce a unit testing framework for Q-learning to disentangle potential bottlenecks where approximate components are replaced by oracles. This allows for controlled analysis of different sources of error. We study various sources of instability and error in Q-learning algorithms on tabular domains, and show that many of these trends hold true in high dimensional domains. We then propose a novel sampling distribution that improve performance even on high-dimensional tasks.

2. Preliminaries

Q-learning algorithms aim to solve a Markov decision process (MDP) by learning the optimal state-action value function, or Q-function. We define an MDP as a tuple $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$. $\mathcal{S}, \mathcal{A}$ represent the state and action spaces, respectively. $T(s'|s,a)$ and $R(s,a)$ represent the dynamics (transition distribution) and reward function, respectively, and $\gamma \in (0,1)$ represents the discount factor. Letting $\rho_0(s)$ denote the initial state distribution, the goal in RL is to find a policy $\pi(a|s)$ that maximizes the expected cumulative discounted rewards, known as the *returns*: $\pi^* = \operatorname*{argmax}_{\pi} E_{s_0\sim\rho_0, s_{t+1}\sim T, a_t\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$ The quantity of interest in many Q-learning methods are the optimal state ($V^*(s)$) and state-action ($Q^*(s,a)$) value functions, which give the expected future return of the optimal policy starting from a particular state or state-action pair. Q-learning algorithms are based on iterating the Bellman backup operator $\mathcal{T}$, defined as

$$(\mathcal{T}Q)(s,a) = R(s,a) + \gamma E_{s'\sim T}[V(s')]$$
$$V(s) = \max_{a'} Q(s,a')$$

Q-iteration is a dynamic programming algorithm that iterates the Bellman backup $Q^{t+1} \leftarrow \mathcal{T}Q^t$. Because the Bellman backup is a γ-contraction in the L_∞ norm, and Q^* is its fixed point, Q-iteration can be shown to converge to Q^* (Sutton & Barto, 2018). A deterministic optimal policy can then be obtained as $\pi^*(s) = \operatorname{argmax}_a Q^*(s,a)$.

When state spaces are large, function approximation is needed to represent the Q-values. This corresponds to *fitted Q-iteration* (FQI) (Ernst et al., 2005), a form of approximate dynamic programming (ADP), which forms the basis of modern deep RL methods such as DQN (Mnih et al., 2015). FQI projects the values of the Bellman backup onto a family of Q-function approximators $\mathcal{Q}$: $Q^{t+1} \leftarrow \Pi_\mu(\mathcal{T}Q^t)$. Π_μ denotes a μ-weighted L_2 projection, which minimizes the *Bellman error* via supervised learning:

$$\Pi_\mu(Q) \overset{\text{def}}{=} \operatorname*{argmin}_{Q'\in\mathcal{Q}} E_{s,a\sim\mu}[(Q'(s,a) - Q(s,a))^2]. \quad (1)$$

The values produced by the Bellman backup, $(\mathcal{T}Q^t)(s,a)$ are commonly referred to as *target values*, and when neural networks are used for function approximation, the previous Q-function $Q^t(s,a)$ is referred to as the *target network*. In this work, we distinguish between the cases when the Bellman error is estimated with Monte-Carlo sampling or computed exactly (see Section 3.1). The sampled variant corresponds to FQI as described in the literature (Ernst et al., 2005; Riedmiller, 2005), while the exact variant is an example of conventional ADP methods (Bertsekas & Tsitsiklis, 1996). Convergence guarantees for Q-iteration do not cleanly translate to FQI. Π_μ is an L_2 projection, but $\mathcal{T}$ is a contraction in the L_∞ norm – this norm mismatch means the composition of the backup and projection is no longer guaranteed to be a contraction under any norm (Bertsekas & Tsitsiklis, 1996), and hence convergence is not guaranteed.

A related branch of Q-learning methods are *online Q-learning* methods, in which Q-values are updated while samples are being collected in the MDP. This includes classic algorithms such as Watkin's Q-learning (Watkins & Dayan, 1992). Online Q-learning methods can be viewed as a form of stochastic approximation applied to Q-iteration and FQI (Bertsekas & Tsitsiklis, 1996), and share many of its theoretical properties (Tsitsiklis, 1994; Szepesvári, 1998). Modern deep RL algorithms such as DQN (Mnih et al., 2015) have characteristics of both online Q-learning and FQI – using replay buffers means the sampling distribution μ changes very little between target updates (see

Section 6.2), and target networks are justified from the viewpoint of FQI. Because FQI corresponds to the case when the sampling distribution is static between target updates, the behavior of modern deep RL methods more closely resembles FQI than a true online method without target networks.

3. Experimental Setup

Our experimental setup is centered around *unit-testing*. We evaluate a spectrum of Q-learning algorithms, starting with exact dynamic programming and replacing exact components with practical approximations, until the algorithm resembles modern deep Q-learning methods.

3.1. Algorithms

We use three different Q-learning variants, each of which controls for a specific source of approximation error – Exact-FQI, Sampling-FQI, and Replay-FQI. We use FQI as a basis for our controlled analysis, as it strongly resembles modern deep RL algorithms while allowing us to separately isolate target values, update rates, and the number of samples used for each iteration. We then confirm that the observed trends hold with several state-of-the-art deep RL methods (SAC (Haarnoja et al., 2017), TD3 (Fujimoto et al., 2018)) on standard benchmark problems.

Exact-FQI (Algorithm 1): Exact-FQI uses known dynamics and reward functions and computes the backup and projection on all state-action tuples, without sampling error. We use Exact-FQI to study convergence, distribution shift (by varying weighting distributions on transitions), and function approximation in the absence of sampling error.

Sampled-FQI (Algorithm 2): Sampled-FQI is a special case of Exact-FQI, where the Bellman error is approximated with Monte-Carlo estimates from a sampling distribution μ, and the Bellman backup is approximated with samples from the dynamics as $r(s,a) + \gamma \max_{a'} Q(s', a')$. We use Sampled-FQI to study effects of overfitting. Sampled-FQI incorporates errors arising from function approximation, sampling, and distribution shift.

Replay-FQI (Algorithm 3): Replay-FQI is a special case of Sampled-FQI that uses a *replay buffer* (Lin, 1992), that saves past transition samples (s, a, s', r), which are used for computing Bellman error. Replay-FQI strongle resembles DQN (Mnih et al., 2015), but lacking the online updates that allow μ to change within an FQI iteration. With large replay buffers, we expect the difference between Replay-FQI and DQN to be minimal as μ changes slowly.

We additionally investigate the following choices of weighting distributions (μ) for the Bellman error:

Unif(s, a): Uniform weights over state-action space.

$\pi(s, a)$: The on-policy state-action marginal induced by π.

$\pi^*(s, a)$: The state-action marginal induced by π^*.

Random(s, a): State-action marginal induced by executing uniformly random actions.

Prioritized(s, a): Weights Bellman errors proportional to $|Q(s, a) - \mathcal{T}Q(s, a)|$. This is similar to prioritized replay (Schaul et al., 2015) with $\beta = 0$.

Replay(s, a) and **Replay10**(s, a): Averaged state-action marginal of all policies (or the last 10) produced during training. This simulates sampling from a replay buffer.

3.2. Domains

We evaluate our methods on suite of tabular environments where we can compute oracle values. We selected 8 tabular domains, each with different qualitative attributes, including: gridworlds of varying sizes and observations, blind Cliffwalk (Schaul et al., 2015), discretized Pendulum and Mountain Car based on OpenAI Gym (Plappert et al., 2018), and a sparsely connected graph. Additional details can be found in Appendix A. In order to provide consistent metrics across domains, we normalize returns and errors involving Q-functions (such as Bellman error) by the returns of the expert policy π^* on each environment.

3.3. Function Approximators

Throughout our experiments, we use 2-layer ReLU networks, denoted by a tuple (N, N) where N represents the number of units in a layer. The "Tabular" architecture refers to the case when no function approximation is used.

3.4. High-Dimensional Testing

In addition to diagnostic experiments on tabular domains, we also wish to see if the observed trends hold true on high-dimensional environments. Thus, we include experiments on continuous control tasks in the OpenAI Gym benchmark (Plappert et al., 2018) (HalfCheetah-v2, Hopper-v2, Ant-v2, Walker2d-v2). In continuous domains, computing the maximum over actions of the Q-value is difficult ($\max_a Q(s, a)$). A common choice in this case is to use an "actor" function to approximate $\arg\max_a Q(s, a)$ (Lillicrap et al., 2015; Fujimoto et al., 2018; Haarnoja et al., 2018). This approach resembles Replay-FQI, but using the actor network in place of the max.

4. Function Approximation and Convergence

This interaction between approximation and convergence has been a long-studied topic in RL and ADP. In control theory, it is closely related to the problems of state-aliasing or interference (Farrell & Berger, 1995). Baird (1995) in-

Algorithm 1 Exact-FQI

```
1: Initialize Q-value approximator $Q_\theta(s,a)$.
2: for step $t$ in $\{1,\dots,N\}$ do
3:     Evaluate $Q_{\theta^t}(s,a)$ at all states.
4:     Compute exact target values at all states.
       $y(s,a) = r(s,a) + \gamma E_{s'}[V_{\theta^t}(s')]$
5:     Minimize projection loss with respect to $\mu$:
       $\underset{\theta}{\text{argmin}}\ E_\mu[(Q_\theta(s,a) - y(s,a))^2]$
6: end for
```

Algorithm 2 Sampled-FQI

```
1: Initialize Q-value approximator $Q_\theta(s,a)$.
2: for step $t$ in $\{1,\dots,N\}$ do
3:     Collect $M$ samples from $\mu$.
4:     Evaluate $Q_{\theta^t}(s,a)$ on samples.
5:     Compute sampled target values on samples.
       $\hat{y}_i = r_i + \gamma V_{\theta^t}(s'_i)$
6:     Minimize projection loss with respect to samples:
       $\underset{\theta}{\text{argmin}}\ \frac{1}{M}\sum_{i=1}^{M}(Q_\theta(s_i,a_i) - y_i)^2$
7: end for
```

Algorithm 3 Replay-FQI

```
1: Initialize Q-value approximator $Q_\theta(s,a)$, replay buffer $\mathcal{B}$.
2: for step $t$ in $\{1,\dots,N\}$ do
3:     Collect $K$ online samples from $\mu$.
4:     Append online samples to buffer $\mathcal{B}$.
5:     Collect $M$ samples from $\mathcal{B}$.
6:     Evaluate $Q_{\theta^t}(s,a)$ on samples.
7:     Compute sampled target values on samples
       $\hat{y}_i = r_i + \gamma V_{\theta^t}(s'_i)$
8:     Minimize projection loss with respect to samples:
       $\underset{\theta}{\text{argmin}}\ \frac{1}{M}\sum_{i=1}^{M}(Q_\theta(s_i,a_i) - y_i)^2$
9: end for
```

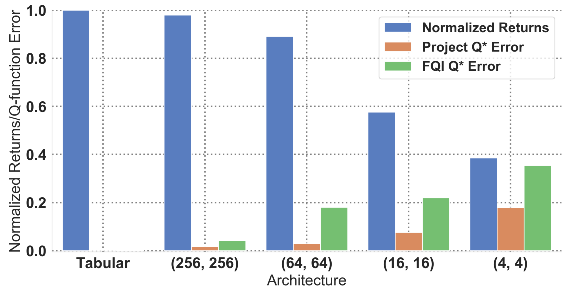
troduces a counterexample in which Watkin's Q-learning with linear approximators causes unbounded divergence. However, several works have noted that divergence need not occur. Munos (2005); Antos et al. (2008); Farahmand et al. (2010) address the norm-mismatch problem by showing convergence guarantees in L_p-norms, at the price of introducing a *concentrability coefficient* that worsens the error bound (and is potentially infinite for deterministic MDPs). In policy evaluation, Tsitsiklis & Van Roy (1997) prove that on-policy TD-learning with linear approximators converges, and methods such as GTD (Sutton et al., 2009a) and ETD (Sutton et al., 2016) have extended results to off-policy cases. In the control scenario, convergent algorithms such as SBEED (Dai et al., 2018) and Greedy-GQ (Maei et al., 2010) have been developed. Concurrently to us,Van Hasselt et al. (2018) experimentally find that unbounded divergence rarely occurs with DQN variants on Atari games.

4.1. How does function approximation affect convergence and solution suboptimality?

The crucial quantities we wish to measure are a trend between function approximation and performance, and a measure for the bias in the learning procedure introduced by function approximation. Using Exact-FQI with uniform weighting (to remove sampling error), we plot the returns of the learned policy, and the L_∞ error between Q^* and the solution found by Exact-FQI ($\lim_{t\to\infty}(\Pi_\mu \mathcal{T})^t Q^0$) and the projection of the optimal solution ($\Pi_\mu Q^*$) in Fig. 1. $\Pi_\mu Q^*$ represents the best solution within model class, in absence of bootstrapping error. Thus, the difference between FQI error and projection error represents the additional bias introduced by FQI (the *inherent Bellman error* of the function class (Munos & Szepesvári, 2008)). This is the potential gap that can be improved via better algorithm design.

We first note the obvious trend that smaller architectures produce lower returns and converge to worse solutions. However, we also find that smaller architectures introduce significant bias in the *learning process*, and there is a significant gap between the solution found by Exact-FQI and the best solution within the model class. One explanation is that when the target is bootstrapped, we must represent all Q-functions along the path to the solution, and not only the final result (Bertsekas & Tsitsiklis, 1996). This observation implies that using large architectures is crucial not only because they have capacity to represent a better solution, but also because they are easier to train using bootstrapping. We also note that divergence rarely occurs in practice, occuring in only 0.9% of our experiments (measured as the Q-values growing larger than 10 times $Q^*(s,a)$).

Figure 1. Normalized returns and Q-function error with function approximation, averaged across domains and seeds. Small architectures show a large gap between the solution found by FQI (FQI Error) and the best solution within model class (Project Error).



For high-dimensional problems, we present experiments on varying the architecture of the Q-network in SAC (Haarnoja et al., 2018) in Appendix Fig. 13. We still observe that large networks have the best performance, and that divergence rarely happens even in high-dimensional continuous spaces. We briefly discuss theoretical intuitions on apparent discrepancy between the lack of unbounded divergence in relation known counterexamples in Appendix B.

5. Sampling Error and Overfitting

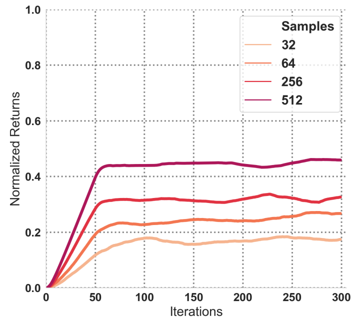


Figure 2. Samples plotted with returns for a 256x256 network. More samples yields better performance.

Approximate dynamic programming assumes that the projection of the Bellman backup (Eqn. 1) is computed exactly, but in reinforcement learning we can normally only compute the *empirical Bellman error* over a finite set of samples. In the PAC framework, overfitting can be quantified by a bounded error in between the empirical and expected loss with high probability, which decays with sample size (Shalev-Shwartz & Ben-David, 2014). Munos & Szepesvári (2008); Maillard et al. (2010); Tosatto et al. (2017) provide such PAC-bounds which account for sampling error in the context of Q-learning and value-based methods, and quantify the quality of the final solution in terms of sample complexity.

We analyze several key points that relate to sampling error. First, we show that Q-learning is prone to overfitting, and that this overfitting has a real impact on performance. We also show that the replay buffer is in fact an effective technique in addressing this issue, and discuss several methods to migitate the effects of overfitting in practice.

5.1. Quantifying Overfitting

We first quantify the amount of overfitting that happens during training, by varying the number of samples. In order provide comparable validation errors across different experiments, we fix a reference sequence of Q-functions, $Q^1, ..., Q^N$, obtained during an arbitrary run of FQI. We then retrace the training sequence, and minimize the projection error $\Pi_\mu(Q^t)$ at each training iteration, using varying amounts of on-policy data or sampling from a replay buffer. We measure the validation error (the expected Bellman error) at each iteration under the on-policy distribution, plotted in Fig. 3. We note the obvious trend that more samples leads to significantly lower validation loss. A more interesting observation is that sampling from the replay buffer results in the lowest on-policy validation loss, despite bias due to distribution mismatch from sampling off-policy data. As we elaborate in Section 6, we believe that replay buffers are effective because they reduce overfitting and have good sample coverage over the state space, not necessarily due to reducing the effects of nonstationarity.

Next, Fig. 2 shows the relationship between number of samples and returns. We see that more samples leads to improved learning speed and a better final solution. A full sweep including architectures is presented in Appendix Fig. 14. Despite overfitting being an issue, larger architectures still perform better because the bias introduced by smaller architectures dominates.

5.2. How can we compensate for overfitting?

Finally, we discuss methods to compensate for overfitting. One common method for reducing overfitting is to regularize the function approximator to reduce its capacity. However, we have seen that weaker architectures can give rise to suboptimal convergence. Instead, we study *early stopping* methods to mitigate overfitting without reducing model size. We first note that the number of gradient steps taken per sample in the projection step has an important effect on performance – too few steps and the algorithm learns slowly, but too many steps and the algorithm may initially learn quickly but overfit. To show this, we run an ablation over the number of gradient steps taken per sample in Replay-FQI and TD3 (TD3 uses 1 by default). Results for FQI are shown in Fig. 4, and for TD3 in Appendix Fig. 15.

In order to understand whether early stopping criteria can reduce overfitting, we employ *oracle* stopping rules to provide an "upper bound" on the best potential improvement. We try two criteria for setting the number of gradient steps: the expected Bellman error and the expected returns of the greedy policy (oracle returns). We implement both methods by running the projection step of FQI to convergence, and retroactively selecting the intermediate Q-function which is judged best by the evaluation metric. Using oracle stopping metrics results in a modest boost in performance in tabular domains (Fig. 5). Thus, we believe that there is promise in further improving such early-stopping methods for reducing overfitting in deep RL algorithms.

We can draw a few conclusions from these experiments. First, overfitting is indeed an issue with Q-learning, and too many gradient steps or too few samples can lead to poor performance. Second, replay buffers and early stopping can mitigate the effects of overfitting. Third, although overfitting is a problem, large architectures are still preferred, because the bias from function approximation outweighs the increased overfitting from using large models.

6. Non-Stationarity

Instability in Q-learning methods is often attributed to the nonstationarity of the objective (Lillicrap et al., 2015; Mnih et al., 2015). Nonstationarity occurs in two places: in the changing target values $\mathcal{T}Q$, and in a changing weighting distribution μ ("distribution shift"). Note that a non-stationary objective, by itself, is not indicative of instability. For example, gradient descent can be viewed as successively minimizing linear approximations to a func-

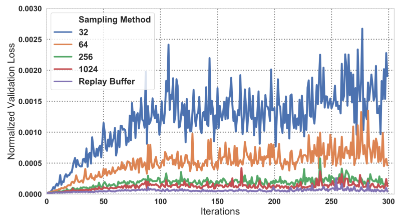


Figure 3. On-policy validation losses for varying amounts of on-policy data (or replay buffer), averaged across environments and seeds. Note that sampling from the replay buffer has lower on-policy validation loss, despite bias from distribution shift.

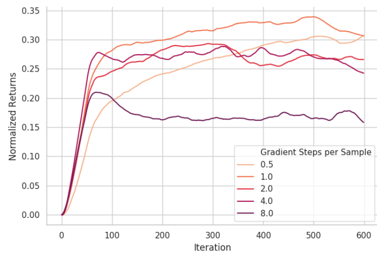


Figure 4. Normalized returns plotted over training iterations (32 samples are taken per iteration), for different ratios of gradient steps per sample using Replay-FQI. We observe that intermediate values of gradient steps work best, and too many gradient steps hinders performance.

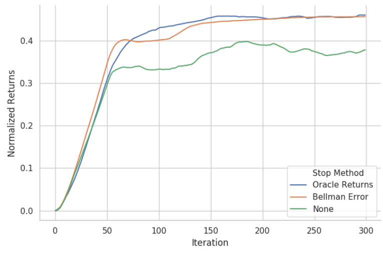


Figure 5. Normalized returns plotted over training iterations (32 samples are taken per iteration), for different early stopping methods using Replay-FQI. We observe that using proper early stopping can result in a modest performance increase.

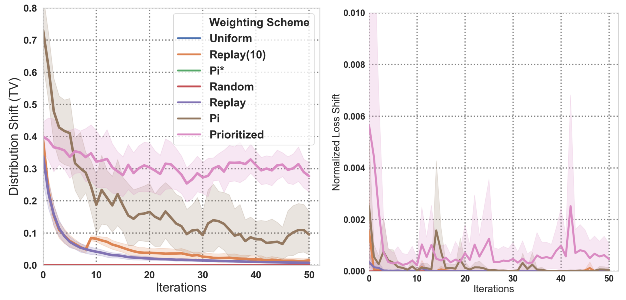
tion: for gradient descent on f with parameter θ and learning rate α, we have the "moving" objective $\theta^{t+1} = \underset{\theta}{\text{argmin}} \left\{\theta^T \nabla_\theta f(\theta^t) - \frac{1}{2\alpha} \|\theta - \theta^t\|_2^2\right\} = \theta^t - \alpha \nabla_\theta f(\theta^t)$. However, the fact that the Q-learning algorithm prescribes an update rule and not a stationary objective complicates analysis. Indeed, the motivation behind algorithms such as GTD (Sutton et al., 2009b;a), approximate linear programming (De Farias, 2002), and residual methods (Baird, 1995; Scherrer, 2010) can be seen as introducing a stationary objective that can be optimized with standard methods such as gradient descent.

6.1. Does a moving target cause instability in the absence of a moving distribution?

To study the moving target problem, we first isolate the speed at which the target changes. To this end, we define the α-smoothed Bellman backup, $\mathcal{T}^\alpha$, which computes an exponentially smoothed update as follows: $\mathcal{T}^\alpha Q = \alpha \mathcal{T} Q + (1-\alpha)Q$. This scheme is inspired by the soft target update used in algorithms such as DDPG (Lillicrap et al., 2015) and SAC (Haarnoja et al., 2017) to improve the stability of learning. Standard Q-iteration uses a "hard" update where $\alpha = 1$. A soft target update weakens the contraction of Q-iteration from γ to $1 - \alpha + \alpha\gamma$ (see Appendix C), so we expect slower convergence, but perhaps it is more stable under heavy function approximation error. We performed experiments with this modified backup using Exact-FQI under the Unif(s, a) weighting distribution.

Our results are presented in Appendix Fig. 10. We find that the most cases, the hard update with $\alpha = 1$ results in the fastest convergence and highest asymptotic performance. However, for smaller architectures, 4×4 and 16×16, lower values of α (such as 0.1) achieve slightly higher asymptotic performance. Thus, while more expressive architectures are still stable under fast-changing targets, we believe that a slowly moving target may have benefits under heavy approximation error. This evidence points to either using large function approximators, in line with the conclusions drawn in the previous sections, or slowing the target updates on problems with high approximation error.

Figure 6. Distribution shift and loss shift plotted against time. Prioritized and on-policy distributions induce the greatest shift, whereas replay buffers greatly reduce the amount of shift.



6.2. Does distribution shift impact performance?

To study the distribution shift problem, we exactly compute the amount of distribution shift between iterations in total-variation distance, $D_{TV}(\mu^{t+1}||\mu^t)$ and the "loss shift": $\mathbb{E}_{\mu^{t+1}}[(Q^t - \mathcal{T}Q^t)^2] - \mathbb{E}_{\mu^t}[(Q^t - \mathcal{T}Q^t)^2]$. The loss shift quantifies the Bellman error objective when evaluated under a new distribution - if the distribution shifts to previously unseen states, we would expect a highly inaccurate Q-value in such states, leading to high loss shift.

We run our experiments using Exact-FQI with a 256x256 layer architecture, and plot the distribution discrepancy and the loss discrepancy in Fig. 6. We find that Prioritized(s, a) has the greatest shift, followed by on-policy variants. Replay buffers greatly reduce distribution shift compared to on-policy learning, which is similar to the de-correlation argument cited for its use by Mnih et al. (2015). However, we find that this metric correlates little with the performance of FQI (Fig. 7). For example, prioritized weighting performs well yet has high distribution shift.

Overall, our experiments indicate that nonstationarities in both distributions and target values, when isolated, do not cause significant stability issues. Instead, other factors such as sampling error and function approximation appear to have more significant effects on performance.

7. Sampling Distributions

Off-policy data has been cited as one of the "deadly triads" for Q-learning (Sutton & Barto, 2018), which has potential to cause instabilities in learning. On-policy distributions (Tsitsiklis & Van Roy, 1997) and fixed behavior distributions (Sutton et al., 2009b; Maei et al., 2010) have often been targeted for theoretical convergence analysis, and many works use importance sampling to correct for off-policyness (Precup et al., 2001; Munos et al., 2016) However, to our knowledge, there is relatively little guidance which compares how different weighting distributions compare in terms of convergence rate and final solutions.

Nevertheless, several works give hypotheses on good choices for weighting distributions. (Munos, 2005) provides an error bound which suggests that "more uniform" distributions can guarantee better worst-case performance. (Geist et al., 2017) suggests that when the state-distribution is fixed, the action distribution should be weighted by the optimal policy for residual Bellman errors. In deep RL, prioritized replay (Schaul et al., 2015), and mixing replay buffer with on-policy data (Hausknecht & Stone, 2016; Zhang & Sutton, 2017) have been found to be effective.

7.1. What Are the Best Weighting Distributions in Absence of Sampling Error?

We begin by studying the effect of weighting distributions when disentangled from sampling error. We run Exact-FQI with an ablation over architectures and weighting distributions and report our results in Fig. 8. Unif(s, a) $Replay(s, a)$, and Prioritized(s, a) consistently result in the highest returns across all architectures. We believe that these results are in favor of the *uniformity* hypothesis: the best distributions spread weight across larger support of the state-action space. For example, a replay buffer contains state-action tuples from many policies, and therefore would be expected to have wider support than the state-action distribution of a single policy. We can see this general trend in Fig. 9.

7.2. Designing a Better Off-Policy Distribution: Adversarial Feature Matching

In our final study, we attempt to design a better weighting distribution using insights from previous sections that can be integrated into deep RL methods. We refer to this method as adversarial feature-matching (AFM). We draw upon three specific insights outlined in previous analysis. First, the function approximator should be incentivized to maximize its ability to distinguish states to minimize function approximation bias (Section 4). Second, the weighting distribution should emphasize areas where the Q-function incurs high Bellman error, to minimize the discrepancy between L_2 norm error and L_∞ norm error. Third, high-entropy weighting distributions tend to be higher performant. The first insight was also demonstrated in (Liu et al., 2018) where enforcing sparsity in the Q-function was found to provide locality in the Q-function which prevented catastrophic interference and provided better values for bootstrapping.

We propose to model our problem as a minimax game, where the weighting distribution is a parameterized adversary $p_\phi(s, a)$ which tries to *maximize* the Bellman error, while the Q-function ($Q_\theta(s, a)$) tries to minimize it. Note that in the unconstrained setting, this game is equivalent to minimizing the L_∞ norm error in its dual-norm representation. However, in practical settings where minimizing stochastic approximations of the L_∞ norm can be difficult for neural networks, it is crucial to introduce constraints to weaken the adversary. These constraints also make the adversary closer to the uniform distribution while allowing it to be sufficiently different at specific state-action pairs.

We use a feature matching constraint which enforces the expected feature vectors, $\mathbb{E}[\Phi(s)]$, under $p_\phi(s, a)$ to *roughly* match the expected feature vector under uniform sampling from the replay buffer. We can express the output of a neural network Q-function as $Q_\theta(s, a) = w_a^T \Phi_\theta(s)$ or, in the continuous case, as $Q_\theta(s, a) = w^T \Phi_\theta(s, a)$, where the feature vector $\Phi_\theta(s), \Phi_\theta(s, a)$ represent the the output of all but the final layer. Intuitively, this constraint restricts the adversary to distributing probability mass among states that are perceptually similar to the Q-function. The objective is

$$\min_{\theta, w} \max_{\phi} \mathbb{E}_{p_\phi(s,a)}[(Q_{w,\theta}(s, a) - y(s, a))^2]$$
$$s.t. \ ||\mathbb{E}_{p_\phi(s,a)}[\Phi(s)] - \frac{\sum_i \Phi(s_i)}{N}|| \leq \varepsilon$$

Note that $\Phi(s)$ is a function of θ but, while solving the maximization, θ is assumed to be a constant. This is equivalent to solving only the inner maximization with a constraint, and empirically provides better stability. Implementation details for AFM are provided in **Appendix D**. The $\frac{\sum_i \Phi(s_i)}{N}$ denotes an estimator for the true expectation under some sampling distribution, such as a uniform distribution over

Figure 7. Average distribution shift across time for different weighting distributions, plotted against returns for a 256x256 model. We find that distribution shift does not have strong correlation with returns.

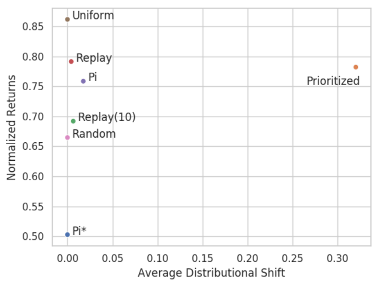


Figure 8. Weighting distribution versus architecture in Exact-FQI. Replay(s, a) consistently provides the highest performance. Note that Adversarial Feature Matching is comparable to Replay(s, a), but surprisingly better for small networks.

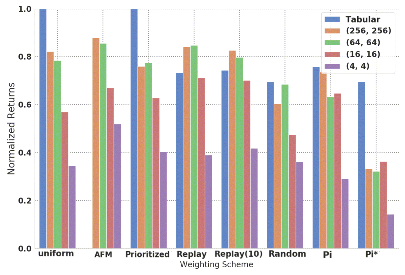
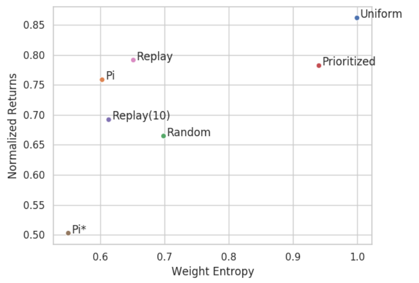


Figure 9. Normalized returns plotted against normalized entropy for different weighting distributions. All experiments use Exact-FQI with a 256x256 network. We see a general trend that high-entropy distributions lead to greater performance.



all states and actions (in exact FQI) or the replay buffer distribution. So, $\frac{\sum_i \Phi(s_i)}{N} \approx E_{p_{rb}}[\Phi]$ holds when using a replay buffer.

In tabular domains with Exact-FQI, we find that AFM performs at par with the top performing weighting distributions, such as Unif(s, a) and *better* than Prioritized(s, a) (Fig. 8). This confirms that adaptive prioritization works better than Prioritized(s, a). Another benefit of AFM is its robustness to function approximation and the performance gains in the case of small architectures (say, $(4, 4)$) are particularly noticeable. (Fig. 8)

In tabular domains with Replay-FQI (Table 1), we also compare AFM to prioritized replay (PER) (Schaul et al., 2015), where AFM and PER perform similarly in terms of normalized returns. We also evaluate a variant of AFM (AFM+Sampling in Table 1) which changes which samples instead of reweighting. We further evaluate AFM on MuJoCo tasks with the TD3 algorithm (Fujimoto et al., 2018) and the entropy constrained SAC algorithm (Haarnoja et al., 2018). The results are presented in the appendix. We find AFM improving performance of the algorithm with three MuJoCo tasks (Ant, Hopper and Cheetah) and two algorithms (TD3 and SAC).

8. Conclusions and Discussion

From our analysis, we have several broad takeaways for the design of deep Q-learning algorithms.

Potential convergence issues with Q-learning do not seem to be endemic empirically, but function approximation still has a strong impact on the solution to which these methods converge. The bias introduced from small architectures is magnified by bootstrapping error. Expressive architectures in general suffer less from bootstrapping error, converge faster, and more stable with moving targets.

Sampling distribution	Norm. Returns (16, 16)	Norm. Returns (64, 64)
None	0.18	0.23
Uniform(s, a)	0.19	0.25
$\pi(s, a)$	**0.45**	0.39
$\pi^*(s, a)$	0.30	0.21
Prioritized(s, a)	0.17	0.33
PER (Schaul et al., 2015)	0.42	**0.49**
AFM (Ours)	0.41	**0.48**
AFM + Sampling (Ours)	0.43	**0.51**

Table 1. Average Performance of various sampling distributions for (16, 16) and (64, 64) neural nets in the setting with replay buffers averaged across 5 random seeds. PER, our AFM and on-policy sampling perform roughly at par on benchmark tasks in expectation when using (16, 16) architectures. However, note that $\pi(s, a)$ is generally computationally intractable.

Sampling error can cause substantial overfitting problems with Q-learning. However, replay buffers and early stopping can mitigate this problem, and the biases incurred from small function approximators outweigh any benefits they may have in terms of overfitting. We believe a good strategy is to use large architectures but tune the number of gradient steps used per sample, or intelligently use early stopping to dynamically control the number of gradient steps.

The choice of sampling or weighting distribution has significant effect on solution quality, even in the absence of sampling error. Surprisingly, we do not find on-policy distributions to be the most performant, but rather methods which have high state-entropy are highly effective. Based on these insights, we propose a new weighting algorithm which balances high-entropy and state aliasing, AFM, that yields fair improvements in both tabular and continuous domains with state-of-the-art off-policy RL algorithms.

Finally, we note that there are several topics that we did not investigate, such as overestimation bias and multi-step returns. We believe that understanding these issues could also be benefit from unit-testing.

Acknowledgements

We thank Vitchyr Pong and Kristian Hartikainen for providing us with implementations of RL algorithms. We thank Csaba Szepesvári and Chelsea Finn for comments on an earlier draft of this paper. SL thanks George Tucker for helpful discussion. We thank Google, NVIDIA, and Amazon for providing computational resources. This research was supported by Berkeley DeepDrive, NSF IIS-1651843 and IIS-1614653, the DARPA Assured Autonomy program, and ARL DCIST CRA W911NF-17-2-0181.

References

Antos, A., Szepesvári, C., and Munos, R. Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71(1):89–129, 2008.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *International Conference on Machine Learning (ICML)*, pp. 214–223, 2017.

Baird, L. Residual Algorithms : Reinforcement Learning with Function Approximation. In *International Conference on Machine Learning (ICML)*, 1995.

Bertsekas, D. P. and Tsitsiklis, J. N. Neuro-dynamic programming. *Athena Scientific*, 1996.

Dai, B., Shaw, A., Li, L., Xiao, L., He, N., Liu, Z., Chen, J., and Song, L. Sbeed: Convergent reinforcement learning with nonlinear function approximation. In *International Conference on Machine Learning*, pp. 1133–1142, 2018.

Daskalakis, C., Ilyas, A., Syrgkanis, V., and Zeng, H. Training GANs with optimism. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SJJySbbAZ.

De Farias, D. P. *The linear programming approach to approximate dynamic programming: Theory and application*. PhD thesis, 2002.

Ernst, D., Geurts, P., and Wehenkel, L. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

Farahmand, A.-m., Szepesvári, C., and Munos, R. Error propagation for approximate policy and value iteration. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

Farrell, J. A. and Berger, T. On the effects of the training sample density in passive learning control. In *American Control Conference*, 1995.

Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, pp. 1587–1596, 2018.

Geist, M., Piot, B., and Pietquin, O. Is the bellman residual a bad proxy? In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 3205–3214. 2017.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning (ICML)*, 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018. URL http://arxiv.org/abs/1801.01290.

Hausknecht, M. and Stone, P. On-policy vs. off-policy updates for deep reinforcement learning. In *Deep Reinforcement Learning: Frontiers and Challenges, IJCAI*, 2016.

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., and Levine, S. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. In *CoRL*, volume 87 of *Proceedings of Machine Learning Research*, pp. 651–673. PMLR, 2018.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2015.

Lin, L.-J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

Liu, V., Kumaraswamy, R., Le, L., and White, M. The utility of sparse representations for control in reinforcement learning. *CoRR*, abs/1811.06626, 2018. URL http://arxiv.org/abs/1811.06626.

Maei, H. R., Szepesvári, C., Bhatnagar, S., and Sutton, R. S. Toward off-policy learning control with function approximation. In *International Conference on Machine Learning (ICML)*, 2010.

Maillard, O.-A., Munos, R., Lazaric, A., and Ghavamzadeh, M. Finite-sample analysis of bellman residual minimization. In *Asian Conference on Machine Learning (ACML)*, pp. 299–314, 2010.

Metelli, A. M., Papini, M., Faccio, F., and Restelli, M. Policy optimization via importance sampling. *CoRR*, abs/1809.06098, 2018. URL http://arxiv.org/abs/1809.06098.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, feb 2015. ISSN 0028-0836.

Munos, R. Error bounds for approximate value iteration. In *AAI Conference on Artificial intelligence (AAAI)*, pp. 1006–1011. AAAI Press, 2005.

Munos, R. and Szepesvári, C. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9 (May):815–857, 2008.

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

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., Kumar, V., and Zaremba, W. Multi-goal reinforcement learning: Challenging robotics environments and request for research, 2018.

Precup, D., Sutton, R. S., and Dasgupta, S. Off-policy temporal difference learning with function approximation. In *International Conference on Machine Learning (ICML)*, pp. 417–424, 2001.

Riedmiller, M. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pp. 317–328. Springer, 2005.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. *International Conference on Learning Representations (ICLR)*, 2015.

Scherrer, B. Should one compute the temporal difference fix point or minimize the bellman residual? the unified oblique projection view. In *International Conference on Machine Learning (ICML)*, pp. 959–966, 2010.

Shalev-Shwartz, S. and Ben-David, S. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Shani, G., Heckerman, D., and Brafman, R. I. An mdp-based recommender system. *Journal of Machine Learning Research*, 6(Sep):1265–1295, 2005.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. Second edition, 2018.

Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, C., and Wiewiora, E. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *International Conference on Machine Learning (ICML)*, 2009a.

Sutton, R. S., Maei, H. R., and Szepesvári, C. A convergent o(n) temporal-difference algorithm for off-policy learning with linear function approximation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2009b.

Sutton, R. S., Mahmood, A. R., and White, M. An emphatic approach to the problem of off-policy temporal-difference learning. *The Journal of Machine Learning Research*, 17 (1):2603–2631, 2016.

Szepesvári, C. The asymptotic convergence-rate of q-learning. In *Advances in Neural Information Processing Systems*, pp. 1064–1070, 1998.

Tosatto, S., Pirotta, M., D'Eramo, C., and Restelli, M. Boosted fitted q-iteration. In *International Conference on Machine Learning (ICML)*, pp. 3434–3443. JMLR. org, 2017.

Tsitsiklis, J. N. Asynchronous stochastic approximation and q-learning. *Machine learning*, 16(3):185–202, 1994.

Tsitsiklis, J. N. and Van Roy, B. Analysis of temporal-diffference learning with function approximation. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1075–1081, 1997.

Tuomas Haarnoja, Aurick Zhou, K. H. G. T. S. H. J. T. V. K. H. Z. A. G. P. A. and Levine, S. Soft actor-critic algorithms and applications. Technical report, 2018.

Van Hasselt, H., Doron, Y., Strub, F., Hessel, M., Sonnerat, N., and Modayil, J. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Yazıcı, Y., Foo, C.-S., Winkler, S., Yap, K.-H., Piliouras, G., and Chandrasekhar, V. The unusual effectiveness of averaging in GAN training. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SJgw_sRqFQ.

Zhang, S. and Sutton, R. S. A deeper look at experience replay. *CoRR*, abs/1712.01275, 2017. URL http://arxiv.org/abs/1712.01275.