Q-VALUE WEIGHTED REGRESSION: REINFORCEMENT LEARNING WITH LIMITED DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Sample efficiency and performance in the offline setting emerged as among the main challenges of deep reinforcement learning. We introduce Q-Value Weighted Regression (QWR), a simple RL algorithm that excels in these aspects. QWR is an extension of Advantage Weighted Regression (AWR), an off-policy actor-critic algorithm that performs very well on continuous control tasks, also in the offline setting, but struggles on tasks with discrete actions and in sample efficiency. We perform a theoretical analysis of AWR that explains its shortcomings and use the insights to motivate QWR theoretically. We show experimentally that QWR matches state-of-the-art algorithms both on tasks with continuous and discrete actions. We study the main hyperparameters of QWR and find that it is stable in a wide range of their choices and on different tasks. In particular, QWR yields results on par with SAC on the MuJoCo suite and – with the same set of hyperparameters – yields results on par with a highly tuned Rainbow implementation on a set of Atari games. We also verify that QWR performs well in the offline RL setting, making it a compelling choice for reinforcement learning in domains with limited data.

1 INTRODUCTION

Deep reinforcement learning has been applied to a large number of challenging tasks, from games (Silver et al., 2017; OpenAI, 2018; Vinyals et al., 2017) to robotic control (Sadeghi & Levine, 2016; OpenAI et al., 2018; Rusu et al., 2016). Since RL makes minimal assumptions on the underlying task, it holds the promise of automating a wide range of applications. However, its widespread adoption has been hampered by a number of challenges. Reinforcement learning algorithms can be substantially more complex to implement and tune than standard supervised learning methods and can have a fair number of hyper-parameters and be brittle with respect to their choices, and may require a large number of interactions with the environment.

These issues are well-known and there has been significant progress in addressing them. The policy gradient algorithm REINFORCE (Williams (1992)) is simple to understand and implement, but is both brittle and requires on-policy data. Proximal Policy Optimization (PPO, Schulman et al. (2017)) is a more stable on-policy algorithm that has seen a number of successful applications despite requiring a large number of interactions with the environment. Soft Actor-Critic (SAC, Haarnoja et al. (2018)) is a much more sample-efficient off-policy algorithm, but it is defined only for continuous action spaces and does not work well in the offline setting, known as batch reinforcement learning, where all samples are provided from earlier interactions with the environment, and the agent cannot collect more samples. Advantage Weighted Regression (AWR, Peng et al. (2019)) is a recent off-policy actor-critic algorithm that works well in the offline setting and is built using only simple and convergent maximum likelihood loss functions, making it easier to tune and debug. It is competitive with SAC given enough time to train, but is less sample-efficient and has not been successful in settings with discrete actions.

We replace the value function training of AWR with Q-value training. Next, we add action sampling to the actor training loop. The resulting algorithm, which we call Q-Value Weighted Regression (QWR) inherits the advantages of AWR but is more sample-efficient and works well with discrete actions and in visual domains, e.g., on Atari games.

To better understand QWR we perform a number of ablations, checking different number of samples in actor training, different advantage estimators, and aggregation functions. These choices affect the

performance of QWR only to a limited extent and it remains stable with each of the choices across the tasks we experiment with.

We run experiments with QWR on the MuJoCo environments and on a subset of the Atari Learning Environment. Since sample efficiency is our main concern, we focus on the difficult case when the number of interactions with the environment is limited – in most our experiments we limit it to 100K interactions. The experiments demonstrate that QWR is indeed more sample-efficient than AWR. On MuJoCo, it performs on par with Soft Actor-Critic (SAC), the current state-of-the-art algorithm for continuous domains. On Atari, QWR performs on par with OTRainbow, a variant of Rainbow highly tuned for sample efficiency. Notably, we use the same set of hyperparameters (except for the network architecture) for both our final MuJoCo and Atari experiments.

2 Q-Value Weighted Regression

2.1 Advantage Weighted Regression

Peng et al. (2019) recently proposed Advantage Weighted Regression (AWR), an off-policy, actor-critic algorithm notable for its simplicity and stability, achieving competitive results across a range of continuous control tasks. It can be expressed as interleaving data collection and two regression tasks performed on the replay buffer, as shown in Algorithm 1.

Algorithm 1 Advantage Weighted Regression.

$\theta \leftarrow$ random actor parameters
$\phi \leftarrow$ random critic parameters
$\mathcal{D} \leftarrow \emptyset$
for k **in** $0..n_iterations - 1$ **do**
    add trajectories $\{\tau_i\}$ sampled by π_θ to $\mathcal{D}$
    for i **in** $0..n_critic_steps - 1$ **do**
        sample $(\mathbf{s}, \mathbf{a}) \sim \mathcal{D}$
        $\phi \leftarrow \phi - \alpha_V \nabla_\phi \left[||\mathcal{R}_{\mathcal{D}}^{\mathbf{s},\mathbf{a}} - V_\phi(\mathbf{s})||^2 \right]$
    end for
    for i **in** $0..n_actor_steps - 1$ **do**
        sample $(\mathbf{s}, \mathbf{a}) \sim \mathcal{D}$
        $\theta \leftarrow \theta + \alpha_\pi \nabla_\theta \left[\log \pi_\theta(\mathbf{a}|\mathbf{s}) \exp(\frac{1}{\beta}(\mathcal{R}_{\mathcal{D}}^{\mathbf{s},\mathbf{a}} - V_\phi(\mathbf{s})) \right]$
    end for
end for

AWR optimizes *expected improvement* of an actor policy $\pi(\mathbf{a}|\mathbf{s})$ over a sampling policy $\mu(\mathbf{a}|\mathbf{s})$ by regression towards the well-performing actions in the collected experience. Improvement is achieved by weighting the actor loss by exponentiated advantage $A_\mu(\mathbf{s}, \mathbf{a})$ of an action, skewing the regression towards the better-performing actions. The advantage is calculated based on the expected return $\mathcal{R}_\mu^{\mathbf{s},\mathbf{a}}$ achieved by performing action $\mathbf{a}$ in state $\mathbf{s}$ and then following the sampling policy μ. To calculate the advantage, one first estimates the value, $V_\mu(s)$, using a learned critic and then computes $A_\mu(\mathbf{s}, \mathbf{a}) = \mathcal{R}_\mu^{\mathbf{s},\mathbf{a}} - V_\mu(\mathbf{s})$. This results in the following formula for the actor:

$$\arg\max_\pi \mathbb{E}_{\mathbf{s}\sim d_\mu(\mathbf{s})} \mathbb{E}_{\mathbf{a}\sim\mu(\cdot|\mathbf{s})} \left[\log \pi(\mathbf{a}|\mathbf{s}) \exp\left(\frac{1}{\beta} (\mathcal{R}_\mu^{\mathbf{s},\mathbf{a}} - V_\mu(\mathbf{s})) \right) \right], \tag{1}$$

where $d_\mu(\mathbf{s}) = \sum_{t=1}^{\infty} \gamma^{t-1} p(\mathbf{s}_t = \mathbf{s}|\mu)$ denotes the unnormalized, discounted state visitation distribution of the policy μ, and β is a temperature hyperparameter.

The critic is trained to estimate the future returns of the sampling policy μ:

$$\arg\min_V \mathbb{E}_{\mathbf{s}\sim d_\mu(\mathbf{s})} \mathbb{E}_{\mathbf{a}\sim\mu(\cdot|\mathbf{s})} \left[||\mathcal{R}_\mu^{\mathbf{s},\mathbf{a}} - V(\mathbf{s})||^2 \right]. \tag{2}$$

To achieve off-policy learning, the actor and the critic are trained on data collected from a mixture of policies from different training iterations, stored in the replay buffer $\mathcal{D}$.

2.2 ANALYSIS OF AWR WITH LIMITED DATA

While AWR achieves very good results after longer training, it is not very sample efficient, as noted in the future work section of (Peng et al., 2019). To understand this problem, we analyze a single loop of actor training in AWR under a special assumption.

The assumption we introduce, called *state-determines-action*, concerns the content of the replay buffer $\mathcal{D}$ of an off-policy RL algorithm. The replay buffer contains all state-action pairs that the algorithm has visited so far during its interactions with the environment. We say that a replay buffer $\mathcal{D}$ satisfies the *state-determines-action* assumption when for each state s in the buffer, there is a unique action that was taken from it, formally:

$$\text{for all } (s, a), (s', a') \in \mathcal{D} : s = s' \implies a = a'.$$

This assumption may seem very limiting and indeed – it is not true in many experimental runs of RL algorithms. Even a random policy starting from the same state will violate the assumption the second time it collects a trajectory. But in the case of limited data, when only a few trajectories were collected, this assumption may hold, at least for a large subset of the replay buffer, which makes it relevant to the study of sample efficiency.

How does AWR perform under the state-determines-action assumption? In Theorem 1, formulated and proven in Appendix 6.2, we show that the AWR update rule under this assumption will converge towards a policy that assigns probability 1 to the actions already present in the replay buffer, thus cloning the previous behaviors. This is not the desired behavior as an algorithm should consider various actions from each state, at least at the beginning.

2.3 Q-VALUE WEIGHTED REGRESSION

To remedy the issue indicated by Theorem 1, we introduce a mechanism to consider multiple different actions that can be taken from a single state. We calculate the advantage of the sampling policy μ based on a learned Q-function: $A_\mu(\mathbf{s}, \mathbf{a}) = Q_\mu(\mathbf{s}, \mathbf{a}) - \hat{V}_\mu(\mathbf{s})$, where $\hat{V}_\mu(\mathbf{s})$ is the expected return of the policy μ, expressed using Q_μ by expectation over actions: $\hat{V}_\mu(\mathbf{s}) = \mathbb{E}_{a\sim\mu(\cdot|\mathbf{s})} Q_\mu(\mathbf{s}, \mathbf{a})$. We substitute our advantage estimator into the AWR actor formula (Equation 1) to obtain the QWR actor:

$$\arg\max_\pi \mathbb{E}_{s\sim d_\mu(\mathbf{s})} \mathbb{E}_{\mathbf{a}\sim\mu(\cdot|\mathbf{s})} \left[\log \pi(\mathbf{a}|\mathbf{s}) \exp\left(\frac{1}{\beta} (Q_\mu(\mathbf{s}, \mathbf{a}) - \hat{V}_\mu(\mathbf{s})) \right) \right]. \tag{3}$$

Similar to AWR, we implement the expectation over states in Equation 3 by sampling from the replay buffer. However, to estimate the expectation over actions, we average over multiple actions sampled from μ during training. Because the replay buffer contains data from multiple different sampling policies, we store the parameters of the sampling policy $\mu(\mathbf{a}|\mathbf{s})$ conditioned on the current state in the replay buffer and restore it in each training step to compute the loss. This allows us to consider multiple different possible actions for a single state when training the actor, not only the one performed in the collected experience.

The use of a Q-network as a critic provides us with an additional benefit. Instead of regressing it towards the returns of our sampling policy μ, we can train it to estimate the returns of an improved policy $\mu^\star$, in a manner similar to Q-learning. This allows us to optimize expected improvement over $\mu^\star$, providing a better baseline - as long as $\mathbb{E}_{\mathbf{a}\sim\mu^\star(\cdot|\mathbf{s})} Q_\mu(\mathbf{s}, \mathbf{a}) \geq \mathbb{E}_{\mathbf{a}\sim\mu(\cdot|\mathbf{s})} Q_\mu(\mathbf{s}, \mathbf{a})$, the *policy improvement theorem* for stochastic policies (Sutton & Barto, 2018, Section 4.2) implies that the policy $\mu^\star$ achieves higher returns than the sampling policy μ:

$$\mathbb{E}_{\mathbf{a}\sim\mu^\star(\cdot|\mathbf{s})} Q_\mu(\mathbf{s}, \mathbf{a}) \geq V_\mu(\mathbf{s}) \Rightarrow V_{\mu^\star}(\mathbf{s}) \geq V_\mu(\mathbf{s}) \tag{4}$$

$\mu^\star$ need not be parametric - in fact, it is not materialized in any way over the course of the algorithm. The only requirement is that we can estimate the Q backup $\mathbb{E}_{\mathbf{a}\sim\mu^\star(\cdot|\mathbf{s})} Q(\mathbf{s}, \mathbf{a})$. This allows great flexibility in choosing the form of $\mu^\star$. Since we want our method to work also in continuous action spaces, we cannot compute the backup exactly. Instead, we estimate it based on several samples from the sampling policy μ. Our backup has the form $\mathbb{E}_{\mathbf{a}_1,...,\mathbf{a}_k\sim\mu(\cdot|\mathbf{s})} F(\{Q(\mathbf{s}, \mathbf{a}_1), ..., Q(\mathbf{s}, \mathbf{a}_k)\})$. In this work, we extend the term *Q-learning* to mean training a Q-value using such a generalized backup.

To make training of the Q-network more efficient, we use multi-step targets, described in detail in Appendix 6.3. The critic optimization objective using single-step targets is:

$$\arg\min_{Q} \mathbb{E}_{\mathbf{s}\sim d_\mu(\mathbf{s})}\mathbb{E}_{\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}\mathbb{E}_{\mathbf{a'}_1,...,\mathbf{a'}_k\sim\mu(\mathbf{a'}|\mathbf{s'})}\left[||Q^\star - Q(\mathbf{s},\mathbf{a})||^2\right], \quad \text{where } Q^\star = r(\mathbf{s},\mathbf{a}) + \gamma F(\{Q_\mu(\mathbf{s},\mathbf{a}_1),...,Q_\mu(\mathbf{s},\mathbf{a}_k)\}). \tag{5}$$

In this work, we investigate three choices of F: average, yielding $\mu^\star = \mu$; max, where $\mu^\star$ approximates the greedy policy; and log-sum-exp, $F(X) = \tau \log\left[\frac{1}{|X|}\sum_{x\in X}\exp(x/\tau)\right]$, interpolating between average and max with the temperature parameter τ. This leads to three versions of the QWR algorithm: QWR-AVG, QWR-MAX, and QWR-LSE. The last operator, log-sum-exp, is similar to the backup operator used in maximum-entropy reinforcement learning (see e.g. Haarnoja et al. (2018)) and can be thought of as a soft-greedy backup, rewarding both high returns and uncertainty of the policy. It is our default choice and the final algorithm is shown in Algorithm 2.

Algorithm 2 Q-Value Weighted Regression.

$\theta \leftarrow$ random actor parameters
$\phi \leftarrow$ random critic parameters
$\mathcal{D} \leftarrow \emptyset$
for k **in** $0..n_iterations - 1$ **do**
  add trajectories $\{\tau_i\}$ sampled by π_θ to $\mathcal{D}$
  $\phi_t \leftarrow \phi$
  for i **in** $0..n_critic_steps - 1$ **do**
    if $i \mod update_frequency = 0$ **then**
      $\phi_t \leftarrow \phi$
    end if
    sample $(\mathbf{s}, \mu, \mathbf{a}, \mathbf{r}, \mathbf{s}') \sim \mathcal{D}$
    sample $\mathbf{a'}_0, ..., \mathbf{a'}_{n-1} \sim \mu(\cdot|\mathbf{s}')$
    $Q^\star \leftarrow \mathbf{r} + \gamma F(\{Q_{\phi_t}(\mathbf{s}', \mathbf{a'}_0), ..., Q_{\phi_t}(\mathbf{s}', \mathbf{a'}_{n-1})\})$
    $\phi \leftarrow \phi - \alpha_V \nabla_\phi \left[||Q^\star - Q_\phi(\mathbf{s}, \mathbf{a})||^2\right]$
  end for
  for i **in** $0..n_actor_steps - 1$ **do**
    sample $(\mathbf{s}, \mu, ...) \sim \mathcal{D}$
    sample $\mathbf{a}_0, ..., \mathbf{a}_{n-1} \sim \mu(\cdot|\mathbf{s})$
    $\hat{V} \leftarrow \frac{1}{n}\sum_{j=0}^{n-1} Q_\phi(\mathbf{s}, \mathbf{a}_j)$
    $\theta \leftarrow \theta + \alpha_\pi \nabla_\theta \frac{1}{n}\sum_{j=0}^{n-1}\left[\log \pi_\theta(\mathbf{a}_j|\mathbf{s}) \exp(\frac{1}{\beta}(Q_\phi(\mathbf{s}, \mathbf{a}_j) - \hat{V}))\right]$
  end for
end for

3 Related work

Reinforcement learning algorithms. Recent years have seen great advances in the field of reinforcement learning due to the use of deep neural networks as function approximators. Mnih et al. (2013b) introduced DQN, an off-policy algorithm learning a parametrized Q-value function through updates based on the Bellman equation. The DQN algorithm only computes the Q-value function, it does not learn an explicit policy. In contrast, policy-based methods such as REINFORCE (Williams, 1992) learn a parameterized policy, typically by following the policy gradient (Sutton et al., 1999) estimated through Monte Carlo approximation of future returns. Such methods suffer from high variance, causing low sample efficiency. Actor-critic algorithms, such as A2C and A3C (Sutton et al., 2000; Mnih et al., 2016), decrease the variance of the estimate by jointly learning policy and value functions, and using the latter as an action-independent baseline for calculation of the policy gradient. The PPO algorithm (Schulman et al., 2017) optimizes a clipped surrogate objective in order to allow multiple updates using the same sampled data.

Continuous control. Lillicrap et al. (2015) adapted Q-learning to continuous action spaces. In addition to a Q-value function, they learn a deterministic policy function optimized by backpropagating the gradient through the Q-value function. Haarnoja et al. (2018) introduce Soft Actor-Critic (SAC): a method learning in a similar way, but with a stochastic policy optimizing the Maximum Entropy RL (Levine, 2018) objective. Similarly to our method, SAC also samples from the policy during training.

Advantage-weighted regression. The QWR algorithm is a successor of AWR proposed by Peng et al. (2019), which in turn is based on Reward-Weighted Regression (RWR, Peters & Schaal (2007)) and AC-REPS proposed by Wirth et al. (2016). Mathematical and algorithmical foundations of advantage-weighted regression were developed by Neumann & Peters (2009). The algorithms share the same good theoretical properties: RWR, AC-REPS, AWR, and QWR losses can be mathematically reformulated in terms of KL-divergence with respect to the optimal policy (see formulas (7)-(10) in Peng et al. (2019)). QWR is different from AWR in the following key aspects: instead of empirical returns in the advantage estimation we train a Q function (see formulas 1 and 3 below for precise definition) and use sampling for the actor. QWR is different from AC-REPS as it uses deep learning for function approximation and Q-learning for fitting the critic, see Section 2.

Several recent works have developed algorithms similar to QWR. We provide a brief overview and ways of obtaining them from the QWR pseudocode (Algorithm 2). AWR can be recovered by learning a value function $V(s)$ as a critic (Line 14) and sampling actions from the replay buffer (lines 12 and 18 in Algorithm 2). AWAC (Nair et al., 2020) modifies AWR by learning a Q-function for the critic. We get it from QWR by sampling actions from the replay buffer (lines 12 and 18). Note that compared to AWAC, by sampling multiple actions for each state, QWR is able to take advantage of Q-learning to improve the critic. CRR (Wang et al., 2020) augments AWAC with training a distributional Q-function in Line 14 and substituting different functions for computing advantage weights in Line 20 [1]. Again, compared to CRR, QWR samples multiple actions for each state, and so can take advantage of Q-learning. In a way similar to QWR, MPO (Abdolmaleki et al., 2018) samples actions during actor training to improve generalization. Compared to QWR, it introduces a dual function for dynamically tuning β in Line 20, adds a prior regularization for policy training and trains the critic using Retrace (Munos et al., 2016) targets in line 13. QWR can be thought of as a significant simplification of MPO, with addition of Q-learning to provide a better baseline for the actor. Additionally, the classical DQN (Mnih et al., 2013a) algorithm for discrete action spaces can be recovered from QWR by removing the actor training loop (lines 16-21), computing a maximum over all actions in Q-network training (line 13) and using an epsilon-greedy policy w.r.t. the Q-network for data collection.

Offline reinforcement learning. Offline RL is the main topic of the survey Levine et al. (2020). The authors state that "offline reinforcement learning methods equipped with powerful function approximation may enable data to be turned into generalizable and powerful decision making engines". We see this as one of the major challenges of modern RL and this work contributes to this challenge. Many current algorithms perform to some degree in offline RL, e.g., variants of DDPG and DQN developed by Fujimoto et al. (2018); Agarwal et al. (2019), as well as the MPO algorithm by Abdolmaleki et al. (2018) are promising alternatives to AWR and QWR analyzed in this work.

ABM (Siegel et al., 2020) is a method of extending RL algorithms based on policy networks to offline settings. It first learns a prior policy network on the offline dataset using a loss similar to Equation 1, and then learns the final policy network using any algorithm, adding an auxiliary term penalizing KL-divergence from the prior policy. CQL (Kumar et al., 2020) is a method of extending RL algorithms based on Q-networks to offline settings by introducing an auxiliary loss. To compute the loss, CQL samples actions on-line during training of the Q-network, similar to Line 14 in QWR. EMaQ (Ghasemipour et al., 2020) learns an ensemble of Q-functions using an Expected-Max backup operator and uses it during evaluation to pick the best action. The Q-network training part is similar to QWR with $F = \max$ in line 13 in Algorithm 2.

The imitation learning algorithm MARWIL by Wang et al. (2018) confirms that the advantage-weighted regression performs well in the context of complex games.

[1] CRR sets the advantage weight function f to be a hyperparameter in $\log \pi_\theta(a_j|s) f(Q_\phi(s, a_j) - \hat{V})$ (Line 20). In QWR, $f(x) = \exp(x/\beta)$.

Algorithm	Half-Cheetah	Walker	Hopper	Humanoid
QWR-LSE	2323 ± 332	$\mathbf{1301 \pm 445}$	$\mathbf{1758 \pm 735}$	511 ± 57
QWR-MAX	2250 ± 254	1019 ± 1185	1187 ± 345	503 ± 49
QWR-AVG	1691 ± 682	1052 ± 231	420 ± 65	455 ± 41
AWR	-0.4 ± 0	67 ± 11	110 ± 81	500 ± 4
SAC	$\mathbf{5492 \pm 8}$	493 ± 6	1197 ± 175	$\mathbf{645 \pm 27}$
PPO	51 ± 41	-14 ± 98	15 ± 75	72 ± 18

Table 1: Comparison of variants of QWR with AWR (Peng et al., 2019), SAC (Haarnoja et al., 2018) and PPO (Schulman et al., 2017) on 4 MuJoCo environments at 100K samples.

Algorithm	Boxing	Breakout	Freeway	Gopher	Pong	Seaquest
QWR-LSE	**4.6**	**8**	21.2	**776**	-7.6	308
QWR-MAX	-1.8	0.8	16.8	580	$\mathbf{-2}$	252
QWR-AVG	-0.8	1.4	19.2	548	-9	296
PPO	-3.9	5.9	8	246	-20.5	**370**
Rainbow	2.5	1.9	**27.9**	349.5	-19.3	354.1
MPR	16.1	14.2	23.1	341.5	-10.5	361.8
MPR-aug	30.5	15.6	24.6	593.4	-3.8	603.8
SimPLe	9.1	16.4	20.3	845.6	12.8	683.3
Random	0.1	1.7	0	257.6	-20.7	68.4

Table 2: Comparison of variants of QWR with the sample-efficient variant of Rainbow (Hessel et al., 2017; van Hasselt et al., 2019), MPR (Schwarzer et al., 2020), SimPLe (Kaiser et al., 2019) and random scores on 6 Atari games at 100K samples. We report results of the the augmented and on-augmented version of the MPR algorithm. Since MPR and SimPLe are based on learning a model of the environment, we do not consider them when choosing the best scores.

4 EXPERIMENTS

Neural architectures. In all MuJoCo experiments, for both value and policy networks, we use multi-layer perceptrons with two layers 256 neurons each, and ReLU activations. In all Atari experiments, for both value and policy networks, we use the same convolutional architectures as in Mnih et al. (2013a). To feed actions to the network, we embed them using one linear layer, connected to the rest of the network using the formula $o \cdot \tanh(a)$ where o is the processed observation and a is the embedded action. This is followed by the value or policy head. For the policy, we parameterize either the log-probabilities of actions in case of discrete action spaces, or the mean of a Gaussian distribution in case of continuous action spaces, while keeping variance constant.

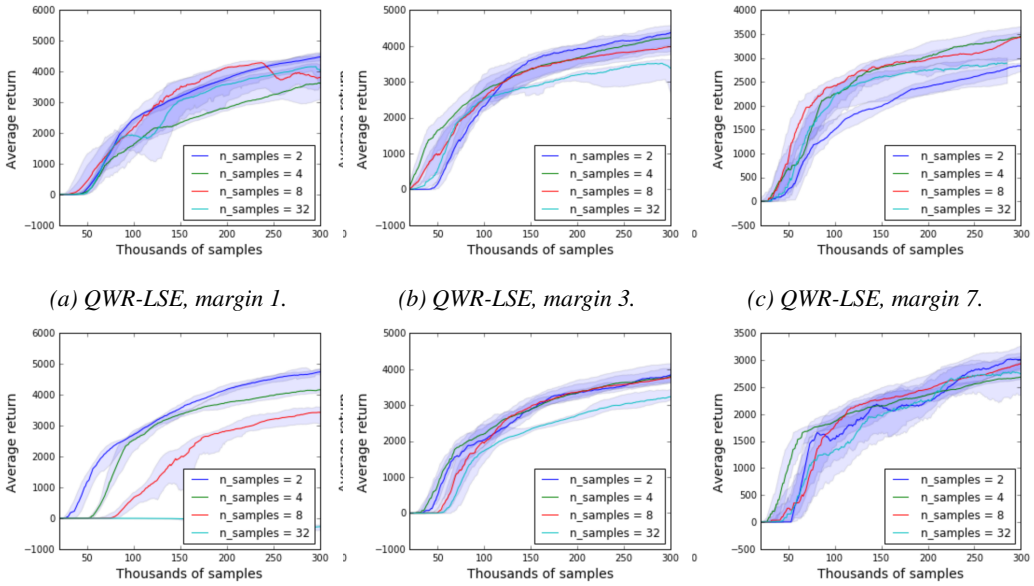
4.1 SAMPLE EFFICIENCY

Since we are concerned with sample efficiency, we focus our first experiments on the case when the number of interactions with the environment is limited. To use a single number that allows comparisons with previous work both on MuJoCo and Atari, we decided to restrict the number of interactions to 100K. This number is high enough, that the state-of-the-art algorithms such as SAC reach very good performance.

We run experiments on 4 MuJoCo environments and 6 Atari games, evaluating three versions of QWR with the 3 backup operators introduced in Section 2.3: QWR-LSE (using log-sum-exp), QWR-MAX (using maximum) and QWR-AVG (using average). For all those experiments, we set the Q target truncation horizon T to 3 and the number of action samples k to 8. We discuss the choice of these values and show ablations below, while more experimental details are given in Appendix 6.1.

In Tables 1 and 2 we present the final numbers at 100K samples for the considered algorithms and environments. To put them within a context, we also provide numbers for SAC, PPO, the variant of Rainbow tuned for sample efficiency, MPR and SimPLe.

On all considered MuJoCo tasks, QWR exceeds the performance of AWR and PPO. The better sample efficiency is particularly well visible in the case of Walker, where each variant of QWR performs



(a) QWR-LSE, margin 1. *(b) QWR-LSE, margin 3.* *(c) QWR-LSE, margin 7.*

(d) QWR-MAX, margin 1. *(e) QWR-MAX, margin 3.* *(f) QWR-MAX, margin 7.*

Figure 1: Ablation of QWR with respect to the margin, the number of action samples and the method of training the critic. The results are shown on the Half-Cheetah environment. The plots show the median of 5 runs with the shaded area denoting the interquartile range.

better than any baseline considered. On Hopper, QWR-LSE - the best variant - outpaces all baselines by a large margin. On Humanoid, it comes close to SAC - the state of the art on MuJoCo.

QWR surpasses PPO and Rainbow in 4 out of 6 Atari games. In Gopher and Pong QWR wins even against the augmented and non-augmented versions of the model-based MPR algorithm.

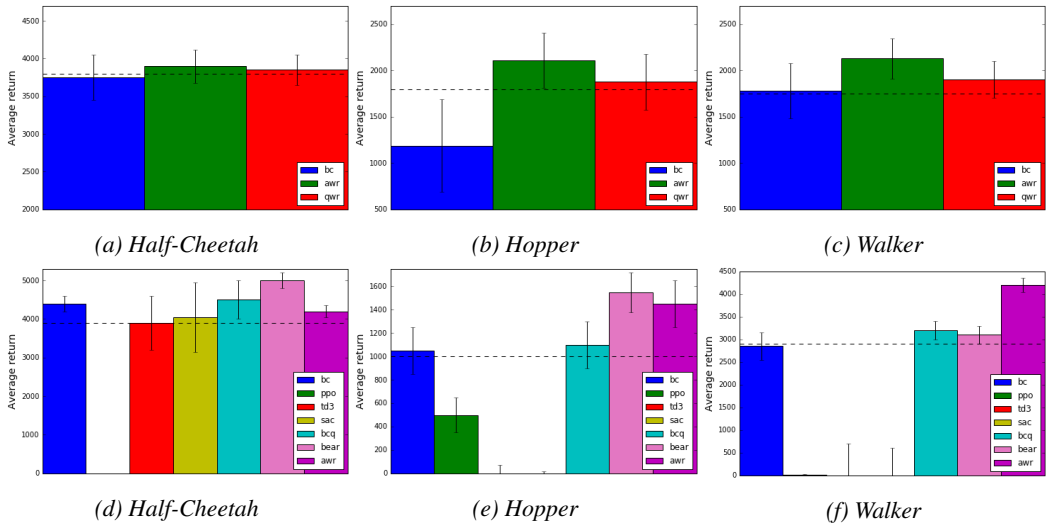
4.2 Ablations

In Figure 1 we provide an ablation of QWR with respect to the backup method F, multistep target horizon T ("margin") and the number of action samples k to consider when training the actor and the critic. As you can see, the algorithm is fairly robust to the choice of these hyperparameters.

In total, the log-sum-exp backup (LSE) achieves best results – compare Figure 1b and Figure 1e. Max backup performs well with margin 1, but is more sensitive to higher numbers of samples – compare Figure 1d and Figure 1e. The log-sum-exp backup is less vulnerable to this effect – compare Figure 1a and Figure 1d. Higher margins decrease performance – see Figure 1c and Figure 1b. We conjecture this to be due to stale action sequences in the replay buffer biasing the multi-step targets. Again, the log-sum-exp backup is less prone to this issue – compare Figure 1c to Figure 1f.

4.3 Offline RL

Both QWR and AWR are capable of handling expert data. AWR was shown to behave in a stable way when provided only with a number of expert trajectories (see Figure 7 in Peng et al. (2019)) without additional data collection. In this respect, the performance of AWR is much more robust than the performance of PPO and SAC. In Figure 2 we show the same result for QWR – in terms of re-using the expert trajectories, it performs as well as AWR. The QWR trainings based on offline data were remarkably stable and worked well across all environments we have tried.



(a) Half-Cheetah *(b) Hopper* *(c) Walker*

(d) Half-Cheetah *(e) Hopper* *(f) Walker*

Figure 2: Figures 2a, 2b and 2b show offline trainings based on 50 trajectories of length 1000 sampled from expert policies. The horizontal lines mark the score of the expert policy. The vertical lines mark the standard deviation. Figures 2d, 2e and 2e are copied from Peng et al. (2019) to cover a broader family of algorithms and show that offline training fails for many RL algorithms.

5 Discussion and Future Work

We present Q-value Weighted Regression (QWR), an off-policy actor-critic algorithm that extends Advantage Weighted Regression with Q-learning. It is more sample-efficient and works well with discrete actions and in visual domains, e.g., on Atari games. QWR consists of two interleaved steps of supervised training: the critic learning the Q function with an L2 loss and the actor learning the policy with weighted regression. Thanks to this clear structure it is simple to implement and debug. It is also stable in a wide range of hyperparameter choices and works well in the offline setting.

Importantly, we designed QWR thanks to a theoretical analysis that revealed why AWR may not work when there are limits on data collection in the environment. Our analysis for the limited data regime is based on the *state-determines-action* assumption that allows to fully solve AWR analytically while still being realistic and indicative of the performance of this algorithm with few samples. We believe that using the *state-determines-action* assumption can yield important insights into other RL algorithms as well.

QWR already achieves state-of-the-art results in settings with limited data and we believe that it can be further improved in the future. The critic training could benefit from the advances in Q-learning methods such as double Q-networks (van Hasselt et al., 2015) or Polyak averaging (Polyak, 1990), already used in SAC. Distributional Q-learning Bellemare et al. (2017) and the use of ensembles like REM Agarwal et al. (2020) could yield further improvements.

Notably, the QWR results we present are achieved with the same set of hyperparameters (except for the network architecture) both for MuJoCo environments and for Atari games. This is rare among deep reinforcement learning algorithms, especially among ones that strive for sample-efficiency. Combined with its stability and good performance in offline settings, this makes QWR a compelling choice for reinforcement learning in domains with limited data.

References

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Rémi Munos, Nicolas Heess, and Martin A. Riedmiller. Maximum a posteriori policy optimisation. *CoRR*, abs/1806.06920, 2018. URL `http://arxiv.org/abs/1806.06920`.

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. Striving for simplicity in off-policy deep reinforcement learning. *CoRR*, abs/1907.04543, 2019. URL `http://arxiv.org/abs/`

1907.04543.

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning, 2020.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning, 2017.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. *CoRR*, abs/1812.02900, 2018. URL http://arxiv.org/abs/1812.02900.

Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl, 2020.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Daniel Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. *CoRR*, abs/1710.02298, 2017. URL http://arxiv.org/abs/1710.02298.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H. Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model-based reinforcement learning for atari. *CoRR*, abs/1903.00374, 2019. URL http://arxiv.org/abs/1903.00374.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning, 2020.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review, 2018.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020.

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, 2015.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013a.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013b.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning, 2016.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. Safe and efficient off-policy reinforcement learning. *CoRR*, abs/1606.02647, 2016. URL http://arxiv.org/abs/1606.02647.

Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets, 2020.

Gerhard Neumann and Jan R. Peters. Fitted q-iteration by advantage weighted regression. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou (eds.), *Advances in Neural Information Processing Systems 21*, pp. 1177–1184. Curran Associates, Inc., 2009. URL http://papers.nips.cc/paper/3501-fitted-q-iteration-by-advantage-weighted-regression.pdf.

OpenAI. Openai five. https://blog.openai.com/openai-five/, 2018.

OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub W. Pachocki, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *CoRR*, abs/1808.00177, 2018. URL http://arxiv.org/abs/1808.00177.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning, 2019.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pp. 745–750, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933. doi: 10.1145/1273496.1273590. URL https://doi.org/10.1145/1273496.1273590.

Boris T. Polyak. New method of stochastic approximation type. 1990.

Andrei A. Rusu, Matej Vecerík, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets. *CoRR*, abs/1610.04286, 2016. URL http://arxiv.org/abs/1610.04286.

Fereshteh Sadeghi and Sergey Levine. (cad)^{2}rl: Real single-image flight without a single real image. *CoRR*, abs/1611.04201, 2016. URL http://arxiv.org/abs/1611.04201.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with momentum predictive representations, 2020.

Noah Y. Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning, 2020.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815, 2017. URL http://arxiv.org/abs/1712.01815.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pp. 1057–1063, Cambridge, MA, USA, 1999. MIT Press.

Richard S Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. A. Solla, T. K. Leen, and K. Müller (eds.), *Advances in Neural Information Processing Systems 12*, pp. 1057–1063. MIT Press, 2000.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning, 2015.

Hado van Hasselt, Matteo Hessel, and John Aslanides. When to use parametric models in reinforcement learning? *CoRR*, abs/1906.05243, 2019. URL http://arxiv.org/abs/1906.05243.

Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, John Quan, Stephen Gaffney, Stig Petersen, Karen Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy P. Lillicrap, Kevin Calderone, Paul Keet, Anthony Brunasso, David Lawrence, Anders Ekermo, Jacob Repp, and Rodney Tsing. Starcraft II: A new challenge for reinforcement learning. *CoRR*, abs/1708.04782, 2017. URL http://arxiv.org/abs/1708.04782.

Qing Wang, Jiechao Xiong, Lei Han, peng sun, Han Liu, and Tong Zhang. Exponentially weighted imitation learning for batched historical data. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 6288–6297. Curran Associates, Inc., 2018.

Ziyu Wang, Alexander Novikov, Konrad Zolna, Jost Tobias Springenberg, Scott Reed, Bobak Shahriari, Noah Siegel, Josh Merel, Caglar Gulcehre, Nicolas Heess, and Nando de Freitas. Critic regularized regression, 2020.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256, May 1992. ISSN 0885-6125. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Christian Wirth, Johannes Fürnkranz, and Gerhard Neumann. Model-free preference-based reinforcement learning. In Dale Schuurmans and Michael P. Wellman (eds.), *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pp. 2222–2228. AAAI Press, 2016. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12247.