A Appendix

A.1 Experimental Details

We run experiments on 4 MuJoCo environments: *Half-Cheetah*, *Walker*, *Hopper* and *Humanoid* and on 6 Atari games: *Boxing*, *Breakout*, *Freeway*, *Gopher*, *Pong* and *Seaquest*. Our code with the exact configurations we use to reproduce the experiments is available as open source[3]. We use QWR as described in Algorithm 2 with the following hyper-parameters:

- We set `n_iterations` to 100 on MuJoCo and on Atari we stop on each game after reaching 100K interactions with the environment (episodes are limited to 2000 interactions during training).
- We set `n_critic_steps` to 3000 and `update_frequency` to 100, so the Q target network is updated 30 times in each epoch.
- We set `n_actor_steps` to 3000 as well.
- The number of samples is 8 by default and the number of steps for multi-step targets is 3.

When training the networks, we use the Adam optimizer. Learning rate for the critic is set to $5e-4$ and for the actor it is set to $1e-4$ for all our experiments. We use the standard architectures for deep networks. In MuJoCo experiments we use a multi-layer perceptron with two layers 256 neurons each and ReLU activations. In Atari experiments we use the same convolutional architectures as Mnih et al. (2013).

A.2 Formal Analysis of AWR with Limited Data

Since sample efficiency is one of the key challenges in deep reinforcement learning, it would be desirable to have better tools to understand why any RL algorithm – for instance AWR – is sample efficient or not. This is hard to achieve in the general setting, but we identify a key simplifying assumption that allows us to solve AWR analytically and identify the source of its problems.

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 limiting and indeed – it might not true in many experimental runs of RL algorithms. Even a random policy starting from the same state could violate the assumption the second time it collects a trajectory. But in the case of limited data, when only a few trajectories are collected, this assumption may hold, at least for a large subset of the replay buffer. This makes it relevant to the study of sample efficiency – we argue that an algorithm that does not perform well under this assumption will rarely be sample-efficient. We believe that good performance under the *state-determines-action* assumption is also necessary for RL algorithms to scale well to larger real-world settings. As the Greek philosopher Heraclitus said, no man ever steps in the same river twice. In real world the same state is never visited twice to give a chance to take different actions.

A simplifying assumption like *state-determines-action* is useful only if it indeed simplifies the analysis of RL algorithms. We show that in case of AWR it does even more – it allows us to analytically calculate the final policy that the algorithm produces. In the case of AWR, it turns out that the resulting policy yields no improvement over the sampling policy.

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 address this problem, let us analyze a single loop of actor training in AWR:

$$\pi_{\mathcal{D}}^{i+1} \leftarrow \arg\max_{\pi} \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}} \left[\log \pi(\mathbf{a}|\mathbf{s}) \exp\left(\frac{1}{\beta} (\mathcal{R}_{\mathcal{D}}^{\mathbf{s},\mathbf{a}} - V_{\mathcal{D}}^{i}(\mathbf{s})) \right) \right] \tag{6}$$

[3] `url_removed_to_preserve_anonymity`

For simplicity, let us denote

$$\alpha_{\mathcal{D}}^{\mathbf{s},\mathbf{a}} = \exp\left(\frac{1}{\beta}(\mathcal{R}_{\mathcal{D}}^{\mathbf{s},\mathbf{a}} - V_{\mathcal{D}}^{i}(\mathbf{s}))\right) \tag{7}$$

so the above update can be written as:

$$\pi_{\mathcal{D}}^{i+1} \leftarrow \arg\max_{\pi} \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}} \left[\log \pi(\mathbf{a}|\mathbf{s}) \alpha_{\mathcal{D}}^{\mathbf{s},\mathbf{a}}\right] \tag{8}$$

How this update acts on a replay buffer that satisfies the *state-determines-action* assumption? It turns out that we can answer this question analytically using the following theorem.

Theorem 1. *Let a replay buffer $\mathcal{D} \subseteq \mathcal{S} \times \mathcal{A}$ satisfy the state-determines-action assumption and let $\pi_{\mathcal{D}}$ be a policy distribution that clones the behaviour from $\mathcal{D}$, i.e., that assigns to each state s from $\mathcal{D}$ the action a such that $(s, a) \in \mathcal{D}$ with probability 1. Then, under the AWR update, $\pi_{\mathcal{D}}^{i+1} \leftarrow \pi_{\mathcal{D}}$.*

Proof. (For discrete action spaces.) By definition of the AWR update rule, $\pi_{\mathcal{D}}^{i+1} \leftarrow \arg\max_{\pi} \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}[\log \pi(\mathbf{a}|\mathbf{s}) \alpha_{\mathcal{D}}^{\mathbf{s},\mathbf{a}}]$. Recall that the number $\alpha_{\mathcal{D}}^{\mathbf{s},\mathbf{a}}$ is non-negative as it is an exponent of another number. Thus the value $\log \pi(\mathbf{a}|\mathbf{s}) \alpha_{\mathcal{D}}^{\mathbf{s},\mathbf{a}}$ can be at most 0, since the logarithm of a probability is negative or 0. The policy $\pi_{\mathcal{D}}$ assigns probability 1 to the action a in state s, so it does reach the value 0 and therefore reaches $\arg\max_{\pi}$ as required. □

As we can see from the above theorem, the AWR update rule will insist on cloning the action taken in the replay buffer as long as it satisfies the *state-determines-action* assumption. In a deterministic environment, the new policy $\pi_{\mathcal{D}}$ will not add any new data to the buffer, only replay a trajectory already in it. So the whole AWR loop will end with the policy $\pi_{\mathcal{D}}$, which yields no improvement.

How come AWR works so well in practice, even with not-that-many interactions? First of all, note that for the above effect to occur in practice, the neural network used for AWR actor must be large enough and trained long enough to memorize the data from the replay buffer. Even when this is the case, the policy it learns must be allowed to express distributions that assign probability 1 to a single action. This holds for environments with discrete actions, but it is not true when using Gaussian distributions with fixed variance in continuous action spaces. But even in that case, using an algorithm that corrects this problem leads to improved sample efficiency, as we show below.

A.2.1 Continuous action spaces

Note that this theorem is also useful in the case of learning continuous policies. For instance, a Gaussian policy with parameterized variance can learn to decrease the variance to be arbitrarily close to zero, in the limit approaching the distribution defined in the theorem. While the theorem still holds, the notions of AWR and the proof need to be clarified in the continuous setting. First of all, let us clarify the notation of Equation 8:

$$\pi_{\mathcal{D}}^{i+1} \leftarrow \arg\max_{\pi} \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}} \left[\log \pi(\mathbf{a}|\mathbf{s}) \alpha_{\mathcal{D}}^{\mathbf{s},\mathbf{a}}\right]$$

For discrete actions, $\pi(\mathbf{a}|\mathbf{s})$ is a discrete probability distribution, so $\pi(\mathbf{a}|\mathbf{s})$ is non-zero for some action $\mathbf{a}$. For continuous distributions $\pi(\mathbf{a}|\mathbf{s})$ may be 0, making the values undefined in that strict sense (and taking the expectation over $\mathcal{D}$ doesn't change that as $\mathcal{D}$ is a finite set). The way to work around this problem is to let $\pi(\mathbf{a}|\mathbf{s})$ be a probability density function instead.

The notions of Theorem 1 must also be clarified in this case: the distribution that assigns probability 1 to a single action does not have a probability density function, but the Dirac delta can be used as its probability density in a generalized way. There is a need to also modify the proof of Theorem 1, as we cannot anymore rely on $\log \pi(\mathbf{a}|\mathbf{s}) \leq 0$, as probability density functions can take arbitrarily large values. But any standard probability density function $\pi(\mathbf{a}|\mathbf{s})$ can be improved by increasing the density at $\mathbf{a}$ and decreasing it everywhere else. So the maximum can only be realized when $\pi(\mathbf{a}|\mathbf{s}) = \infty$, as for the Dirac delta.

To clarify the above considerations in more specific terms, let us consider the common practical case where we only consider π that are probability density functions of the Gaussian distribution. In that case, each $\pi(\cdot|\mathbf{s})$ is characterized by $\sigma(s)$ and $\mu(s)$ of the Gaussian in state s, that is

$$\pi(\mathbf{a}|\mathbf{s}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{a-\mu}{\sigma})^2}$$

and

$$\log \pi(a|s) = \log(\frac{1}{\sigma\sqrt{2\pi}}) + \log(e^{-\frac{1}{2}(\frac{a-\mu}{\sigma})^2}) =$$
$$= -\log(\sigma\sqrt{2\pi}) - \frac{1}{2}(\frac{a-\mu}{\sigma})^2.$$

This is maximized by choosing $\mu = a$ and reducing σ towards 0: at $\sigma = e^{-n}$ we get

$$\log \pi(a|s) = -\log(\sigma\sqrt{2\pi}) - \frac{1}{2}(\frac{a-\mu}{\sigma})^2 =$$
$$= -\log(\sqrt{2\pi}e^{-n}) - 0 =$$
$$= n - \log\sqrt{2\pi},$$

which approaches infinity as we increase n. This corresponds to a sequence of Gaussian distributions with mean at the action a and variance decreasing towards 0, converging to the Dirac delta, which assigns the whole probability mass to a single action.

A.3 Formal Analysis of QWR with Limited Data

To see how QWR performs under limited data, we are going to formulate a positive theorem showing that it achieves the policy improvement that AWR aims for even in a limited data setting. Note that this time we allow replay buffers that do not not necessarily satisfy the *state-determines-action* assumption. But, for clarity, we make a simplifying assumption that the replay buffer has been sampled by a single policy μ.

Recall the QWR update rule:

$$\pi_{\mathcal{D}}^{i+1} \leftarrow \arg\max_{\pi} \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\mathbb{E}_{\mathbf{a}\sim\mu(\cdot|\mathbf{s})}\left[\log \pi(\mathbf{a}|\mathbf{s}) \exp\left(\frac{1}{\beta}(Q_\mu(\mathbf{s},\mathbf{a}) - \hat{V}_\mu(\mathbf{s}))\right)\right] \tag{9}$$

where $\hat{V}_\mu(\mathbf{s}) = \mathbb{E}_{a\sim\mu(\cdot|\mathbf{s})}Q_\mu(\mathbf{s},\mathbf{a})$ and $\mathcal{D}$ is the set of states in the replay buffer.

Let $\pi_\mu^\star(\mathbf{a}|\mathbf{s}) \propto \mu(\mathbf{a}|\mathbf{s})\exp\left(\frac{1}{\beta}\left(\mathcal{R}_\mu^{\mathbf{s},\mathbf{a}} - V_\mu(\mathbf{s})\right)\right)$, where V_μ is the state value function of μ. This is the policy optimizing the expected improvement over the sampling policy μ, subject to a KL constraint – the same as in Equation 36 in Peng et al. (2019).

Since $\pi_\mu^\star$ is the target policy resulting from the AWR derivation, we know from Peng et al. (2019) that AWR will update towards this policy in the limit, when the replay buffer is large enough. But from Theorem 1 we know that it will fail to perform this update when the state-determines-action assumption holds. Below we show that QWR will perform the same desirable update for any replay buffer, as long as we restrict the attention to states in the buffer.

Theorem 2. *Let $\mathcal{D} \subseteq \mathcal{S}$ be a finite sample from $d_\mu(\mathbf{s})$ - the undiscounted state distribution of a policy $\mu(\mathbf{a}|\mathbf{s})$. Let Q_μ be the state-action value function for μ, so $Q_\mu(\mathbf{s},\mathbf{a}) = \mathcal{R}_\mu^{\mathbf{s},\mathbf{a}}$ for any state and action. Then, under the QWR actor update, $\pi_{\mathcal{D}}^{i+1} = \pi_\mu^\star|_{\mathcal{D}}$ - the policy $\pi_\mu^\star$ restricted to the set of states $\mathcal{D}$.*

Proof. From the definition of $\hat{V}_\mu(\mathbf{s})$ we have $\hat{V}_\mu(\mathbf{s}) = \mathbb{E}_{a\sim\mu(\cdot|\mathbf{s})}Q_\mu(\mathbf{s},\mathbf{a}) = V_\mu(\mathbf{s})$. Since $Q_\mu(\mathbf{s},\mathbf{a}) = \mathcal{R}_\mu^{\mathbf{s},\mathbf{a}}$,

$$\pi_{\mathcal{D}}^{i+1} \leftarrow \arg\max_{\pi} \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\mathbb{E}_{\mathbf{a}\sim\mu(\cdot|\mathbf{s})}\left[\log \pi(\mathbf{a}|\mathbf{s}) \exp\left(\frac{1}{\beta}(\mathcal{R}_\mu^{\mathbf{s},\mathbf{a}} - V_\mu(\mathbf{s}))\right)\right]. \tag{10}$$

We can now change the measure using the definition of $\pi^\star$:

$$\pi_{\mathcal{D}}^{i+1} \leftarrow \arg\max_{\pi} \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\mathbb{E}_{\mathbf{a}\sim\pi^\star(\cdot|\mathbf{s})} \log \pi(\mathbf{a}|\mathbf{s}). \tag{11}$$

The inner expectation, up to a normalizing constant, is the negative cross-entropy between $\pi^\star$ and π. Since cross-entropy between two distributions is minimized when the distributions are equal, and our optimization problem can be solved for every state from $\mathcal{D}$ separately, the optimum is reached at $\pi = \pi_\mu^\star|_{\mathcal{D}}$. □

As we can see, the QWR update rule reaches the desired target policy even under limited data. This stands in contrast to AWR, which requires repeating states in the replay buffer, as shown in Theorem 1.

A.4 Multi-step targets

To make the training of the Q-value network more efficient, we implement an approach inspired by widely-used multi-step Q-learning (Mnih et al., 2016). We consider targets for the Q-value network computed over multiple different time horizons:

$$Q^{\star}_{\mu,t}(\mathbf{s}_1, \mathbf{a}_1) = \sum_{i=1}^{t} \gamma^{i-1}\mathbf{r}_i + \gamma^t \mathbb{E}_{\mathbf{a}'_1,\dots,\mathbf{a}'_n \sim \mu(\cdot|\mathbf{s}_{t+1})} F(\{Q_\mu(\mathbf{s}_{t+1}, \mathbf{a}'_1), \dots, Q_\mu(\mathbf{s}_{t+1}, \mathbf{a}'_n)\}) \tag{12}$$

where $\mathbf{s}_i$, $\mathbf{a}_i$, $\mathbf{r}_i$ are the states, actions and rewards in a collected trajectory, respectively. We aggregate those multi-step targets using a truncated TD(λ) estimator (Sutton and Barto, 2018, p. 236):

$$Q^{\star}_{\mu}(\mathbf{s}, \mathbf{a}) = (1 - \lambda) \sum_{t=1}^{T} \lambda^{t-1} Q^{\star}_{\mu,t}(\mathbf{s}, \mathbf{a}) \tag{13}$$