SOFT ACTOR-CRITIC: OFF-POLICY MAXIMUM ENTROPY DEEP REINFORCEMENT LEARNING WITH A STOCHASTIC ACTOR

Anonymous authors
Paper under double-blind review

ABSTRACT

Model-free deep reinforcement learning (RL) algorithms have been demonstrated on a range of challenging decision making and control tasks. However, these methods typically suffer from two major challenges: very high sample complexity and brittle convergence properties, which necessitate meticulous hyperparameter tuning. Both of these challenges severely limit the applicability of such methods to complex, real-world domains. In this paper, we propose soft actor-critic, an off-policy actor-critic deep RL algorithm based on the maximum entropy reinforcement learning framework. In this framework, the actor aims to maximize expected reward while also maximizing entropy—that is, succeed at the task while acting as randomly as possible. Prior deep RL methods based on this framework have been formulated as either off-policy Q-learning, or on-policy policy gradient methods. By combining off-policy updates with a stable stochastic actor-critic formulation, our method achieves state-of-the-art performance on a range of continuous control benchmark tasks, outperforming prior on-policy and off-policy methods.

1 INTRODUCTION

Model-free deep reinforcement learning (RL) algorithms have been applied in a range of challenging domains, from games (Mnih et al., 2013; Silver et al., 2016) to robotic control (Schulman et al., 2015). The combination of RL and high-capacity function approximators such as neural networks holds the promise of automating a wide range of decision making and control tasks, but widespread adoption of these methods in real-world domains has been hampered by two major challenges. First, model-free deep RL methods are notoriously expensive in terms of their sample complexity. Even relatively simple tasks can require millions of steps of data collection, and complex behaviors with high-dimensional observations might need substantially more. Second, these methods are often brittle with respect to their hyperparameters: learning rates, exploration constants, and other settings must be set carefully for different problem settings to achieve good results. Both of these challenges severely limit the applicability of model-free deep RL to real-world tasks.

One cause for the poor sample efficiency of deep RL methods is on-policy learning: some of the most commonly used deep RL algorithms, such as TRPO (Schulman et al., 2015) or A3C (Mnih et al., 2016), require new samples to be collected for each gradient step on the policy. This quickly becomes extravagantly expensive, as the number of gradient steps to learn an effective policy increases with task complexity. Off-policy algorithms instead aim to reuse past experience. This is not directly feasible with conventional policy gradient formulations (Schulman et al., 2015; Mnih et al., 2016), but is relatively straightforward for Q-learning based methods (Mnih et al., 2015). Unfortunately, the combination of off-policy learning and high-dimensional, nonlinear function approximation with neural networks presents a major challenge for stability and convergence (Bhatnagar et al., 2009). This challenge is further exacerbated in continuous state and action spaces, where a separate actor network is typically required to perform the maximization in Q-learning. A commonly used algorithm in such settings, deep deterministic policy gradient (DDPG) (Lillicrap et al., 2015), provides for sample-efficient learning, but is notoriously challenging to use due to its extreme brittleness and hyperparameter sensitivity (Duan et al., 2016; Henderson et al., 2017).

We explore how to design an efficient and stable model-free deep RL algorithm for continuous state and action spaces. To that end, we draw on the maximum entropy framework, which aug-

ments the standard maximum reward reinforcement learning objective with an entropy maximization term (Ziebart et al., 2008; Toussaint, 2009; Rawlik et al., 2012; Fox et al., 2016; Haarnoja et al., 2017). Maximum entropy reinforcement learning alters the RL objective, though the original objective can be recovered by using a temperature parameter (Haarnoja et al., 2017). More importantly, the maximum entropy formulation provides a substantial improvement in exploration and robustness: as discussed by Ziebart (2010), maximum entropy policies are robust in the face of modeling and estimation errors, and as demonstrated by Haarnoja et al. (2017), they improve exploration by acquiring diverse behaviors. Prior work has proposed model-free deep RL algorithms for continuous action spaces that perform on-policy learning with entropy maximization (O'Donoghue et al., 2016), as well as off-policy methods based on soft Q-learning (Schulman et al., 2017; Haarnoja et al., 2017). However, the on-policy variants suffer from poor sample complexity for the reasons discussed above, while the off-policy variants require complex approximate inference procedures in continuous action spaces.

In this paper, we demonstrate that we can devise an off-policy maximum entropy actor-critic algorithm, which we call soft actor-critic, which provides for both sample-efficient learning and stability. This algorithm extends readily to very complex, high-dimensional tasks, such as the Humanoid benchmark in OpenAI gym (Brockman et al., 2016), where off-policy methods such as DDPG typically struggle to obtain good results (Gu et al., 2016), while avoiding the complexity and potential instability associated with approximate inference in prior off-policy maximum entropy algorithms based on soft Q-learning (Haarnoja et al., 2017). We present a convergence proof for our algorithm based on a soft variant of policy iteration, and present empirical results that show a substantial improvement in both performance and sample efficiency over both off-policy and on-policy prior methods.

2 Related Work

Our soft actor-critic algorithm incorporates three key ingredients: an actor-critic architecture with separate policy and value function networks, an off-policy formulation that enables reuse of previously collected data for efficiency, and entropy maximization to enable stability and exploration. We review prior works that draw on some of these ideas in this section. Actor-critic algorithms are typically derived starting from policy iteration, which alternates between *policy evaluation*—computing the value function for a policy—and *policy improvement*—using the value function to obtain a better policy (Barto et al., 1983; Sutton & Barto, 1998). In large-scale reinforcement learning problems, it is typically impractical to run either of these steps to convergence, and instead the value function and policy are optimized jointly. In this case, the policy is referred to as the actor, and the value function as the critic. Many actor-critic algorithms build on the standard, on-policy policy gradient formulation to update the actor (Peters & Schaal, 2008; Schulman et al., 2015; Mnih et al., 2016). This tends to improve stability, but results in very poor sample complexity.

There have been efforts to increase the sample efficiency while retaining the robustness properties by incorporating off-policy samples and by using higher order variance reduction techniques (O'Donoghue et al., 2016; Gu et al., 2016). However, fully off-policy algorithms still attain better efficiency. A particularly popular off-policy actor-critic variant is based on the deterministic policy gradient (Silver et al., 2014) and its deep counterpart, DDPG (Lillicrap et al., 2015). This method uses a Q-function estimator to enable off-policy learning, and a deterministic actor that maximizes this Q-function. As such, this method can be viewed both as a deterministic actor-critic algorithm and an approximate Q-learning algorithm. Unfortunately, the interplay between the deterministic actor network and the Q-function typically makes DDPG extremely difficult to stabilize and brittle to hyperparameter settings (Duan et al., 2016; Henderson et al., 2017). As a consequence, it is difficult to extend DDPG to very complex, high-dimensional tasks, and on-policy policy gradient methods still tend to produce the best results in such settings (Gu et al., 2016). Our method instead combines off-policy actor-critic training with a *stochastic* actor, and further aims to maximize the entropy of this actor with an entropy maximization objective. We find that this actually results in a substantially more stable and scalable algorithm that, in practice, exceeds both the efficiency and final performance of DDPG.

Maximum entropy reinforcement learning optimizes policies to maximize both the expected return and the expected entropy of the policy. This framework has been used in many contexts, from

inverse reinforcement learning (Ziebart et al., 2008) to optimal control (Todorov, 2008; Toussaint, 2009; Rawlik et al., 2012). More recently, several papers have noted the connection between Q-learning and policy gradient methods in the framework of maximum entropy learning (O'Donoghue et al., 2016; Haarnoja et al., 2017; Nachum et al., 2017a; Schulman et al., 2017). While most of the prior works assume a discrete action space, Nachum et al. (2017b) approximate the maximum entropy distribution with a Gaussian and Haarnoja et al. (2017) with a sampling network trained to draw samples from the optimal policy. Although the soft Q-learning algorithm proposed by Haarnoja et al. (2017) has a value function and actor network, it is not a true actor-critic algorithm: the Q-function is estimating the optimal Q-function, and the actor does not directly affect the Q-function except through the data distribution. Hence, Haarnoja et al. (2017) motivates the actor network as an approximate sampler, rather than the actor in an actor-critic algorithm. Crucially, the convergence of this method hinges on how well this sampler approximates the true posterior. In contrast, we prove that our method converges to the optimal policy from a given policy class, regardless of the policy parameterization. Furthermore, these previously proposed maximum entropy methods generally do not exceed the performance of state-of-the-art off-policy algorithms, such as DDPG, when learning from scratch, though they may have other benefits, such as improved exploration and ease of finetuning. In our experiments, we demonstrate that our soft actor-critic algorithm does in fact exceed the performance of state-of-the-art off-policy deep RL methods by a wide margin.

3 PRELIMINARIES

In this section, we introduce notation and summarize the standard and maximum entropy reinforcement learning frameworks.

3.1 NOTATION

We address policy learning in continuous action spaces. To that end, we consider infinite-horizon Markov decision processes (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, p_{\mathbf{s}}, r)$, where the state space $\mathcal{S}$ and the action space $\mathcal{A}$ are assumed to be continuous, and the unknown state transition probability $p_{\mathbf{s}} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, \infty)$ represents the probability density of the next state $\mathbf{s}_{t+1} \in \mathcal{S}$ given the current state $\mathbf{s}_t \in \mathcal{S}$ and action $\mathbf{a}_t \in \mathcal{A}$. The environment emits a reward $r : \mathcal{S} \times \mathcal{A} \to [r_{\min}, r_{\max}]$ on each transition, which we will abbreviate as $r_t \triangleq r(\mathbf{s}_t, \mathbf{a}_t)$ to simplify notation. We will also use $\rho_\pi(\mathbf{s}_t)$ and $\rho_\pi(\mathbf{s}_t, \mathbf{a}_t)$ to denote the state and state-action marginals of the trajectory distribution induced by a policy $\pi(\mathbf{a}_t|\mathbf{s}_t)$.

3.2 MAXIMUM ENTROPY REINFORCEMENT LEARNING

The standard objective used in reinforcement learning is to maximize the expected sum of rewards $\sum_t \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi}[r_t]$. We will consider a more general maximum entropy objective, which favors stochastic policies by augmenting the objective with the expected entropy of the policy over $\rho_\pi(\mathbf{s}_t)$:

$$J(\pi) = \sum_{t=0}^{T-1} \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi}\left[r(\mathbf{s}_t, \mathbf{a}_t) + \alpha\mathcal{H}(\pi(\,\cdot\,|\mathbf{s}_t))\right]. \tag{1}$$

The temperature parameter α determines the relative importance of the entropy term against the reward, and thus controls the stochasticity of the optimal policy. The maximum entropy objective differs from the standard maximum expected reward objective used in conventional reinforcement learning, though the conventional objective can be recovered in the limit as $\alpha \to 0$. For the rest of this paper, we will omit writing the temperature explicitly, as it can always be subsumed into the reward by scaling it by α^{-1}. The maximum entropy objective has a number of conceptual and practical advantages. First, the policy is incentivized to explore more widely, while giving up on clearly unpromising avenues. Second, the policy can capture multiple modes of near-optimal behavior. In particular, in problem settings where multiple actions seem equally attractive, the policy will commit equal probability mass to those actions. Lastly, prior work has observed substantially improved exploration from this objective (Haarnoja et al., 2017; Schulman et al., 2017), and in our experiments, we observe that it considerably improves learning speed over state-of-art methods that optimize the conventional objective function. If we wish to extend the objective to infinite horizon problems, it is convenient to also introduce a discount factor γ to ensure that the sum of expected

rewards and entropies is finite. Writing down the precise maximum entropy objective for the infinite horizon discounted case is more involved (Thomas, 2014) and is deferred to Appendix A.

Just as in standard reinforcement learning, maximum entropy reinforcement learning can make use of Q-functions and value functions, and as we note in Section 4, the soft Q-function of a stochastic policy π satisfies the soft Bellman equation

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \, \mathbb{E}_{\mathbf{s}_{t+1} \sim p_\mathbf{s}} \left[V^\pi(\mathbf{s}_{t+1}) \right], \tag{2}$$

where the soft value $V^\pi(\mathbf{s}_t)$ is given by

$$V^\pi(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi} \left[Q^\pi(\mathbf{s}_t, \mathbf{a}_t) - \log \pi(\mathbf{a}_t | \mathbf{s}_t) \right]. \tag{3}$$

Prior methods have proposed directly solving for the optimal Q-function, from which the optimal policy can be recovered (Ziebart et al., 2008; Fox et al., 2016; Haarnoja et al., 2017). In the next section, we will discuss how we can devise a soft actor-critic algorithm through a policy iteration formulation, where we instead evaluate the Q-function of the current actor, and update the actor through an *off-policy* gradient update. Though such algorithms have previously been proposed for conventional reinforcement learning, our method is, to our knowledge, the first off-policy actor-critic method in the maximum entropy reinforcement learning framework.

4 From Soft Policy Iteration to Soft Actor-Critic

Our off-policy soft actor-critic algorithm can be derived starting from the maximum entropy variant of the policy iteration method. In this section, we will first present this derivation, verify that the corresponding algorithm converges to an optimal policy from its function class, and then present a practical deep reinforcement learning algorithm based on this theory.

4.1 Derivation of Soft Policy Iteration

We will begin by deriving soft policy iteration, a general algorithm for learning optimal maximum entropy policies by alternating policy evaluation and policy improvement in the maximum entropy framework. Our derivation is based on a tabular setting, to enable theoretical analysis and convergence guarantees, and we extend this method into the general continuous setting in the next section. We will show that soft policy iteration converges to the optimal policy within a set of policies which might correspond, for instance, to a set of parameterized functions.

In the policy evaluation step of soft policy iteration, we wish to compute the value of a policy, $\pi(\mathbf{a}_t|\mathbf{s}_t)$, according to the maximum entropy objective in Equation 1. For a fixed policy, the soft Q-value can be computed iteratively, by starting from any function $Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ and iteratively applying a modified version of the Bellman backup operator:

$$Q \leftarrow \mathcal{T}^\pi Q, \tag{4}$$

where $\mathcal{T}^\pi$ is the soft Bellman backup operator defined by

$$\mathcal{T}^\pi Q = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \, \mathbb{E}_{\mathbf{s}_{t+1} \sim p_\mathbf{s}, \mathbf{a}_{t+1} \sim \pi} \left[Q(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) - \log \pi(\mathbf{a}_{t+1} | \mathbf{s}_{t+1}) \right]. \tag{5}$$

Indeed, we can show that repeated application of Equation 4 will converge to the soft value of the fixed policy π, as formalized in Lemma 1.

Lemma 1 (Soft Policy Evaluation)**.** *Consider the soft Bellman backup operator $\mathcal{T}^\pi$ in Equation 5 and a mapping $Q^k : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ and define $Q^{k+1} = \mathcal{T}^\pi Q^k$. Then the sequence Q^k will converge to the soft value of π as $k \rightarrow \infty$.*

Proof. See the appendix. □

In the policy improvement step, we update the policy towards the exponential of the new Q-function. This particular choice of update can be guaranteed to result into an improved policy in terms of its soft value, as we show in this section. Since in practice we prefer policies that are tractable, we will additionally restrict the policy to some set of policies Π, which can correspond, for example, to a parameterized family of distributions such as Gaussians. To account for the constraint that $\pi \in \Pi$, we project the improved policy into the desired set of policies. While in principle we could choose

Algorithm 1: Soft Policy Iteration

while $\pi_{\text{new}}(\mathbf{a}_t|\mathbf{s}_t) \neq \pi_{\text{old}}(\mathbf{a}_t|\mathbf{s}_t)$ *for some* $(\mathbf{a}_t, \mathbf{s}_t) \in \mathcal{A} \times \mathcal{S}$ **do**
  $Q^0 \leftarrow Q^{\pi_{\text{old}}}$
  while $Q^{k+1}(\mathbf{s}_t, \mathbf{a}_t) > Q^k(\mathbf{s}_t, \mathbf{a}_t)$ *for some* $(\mathbf{a}_t, \mathbf{s}_t) \in \mathcal{A} \times \mathcal{S}$ **do**
    $Q^{k+1} \leftarrow \mathcal{T}^{\pi_{new}} Q^k$
    $k \leftarrow k + 1$
  end
  $\pi_{\text{old}} \leftarrow \pi_{\text{new}}$
  $\pi_{\text{new}} \leftarrow \arg\min_{\pi' \in \Pi} \mathrm{D}_{\mathrm{KL}}\left(\pi' \parallel \exp\left(Q^{\pi_{\text{old}}} - \log Z^{\pi_{\text{old}}}\right)\right)$
end

any projection, it will turn out to be convenient to use the information projection defined in terms of the Kullback-Leibler divergence. In the other words, in the policy improvement step, we update the policy according to

$$\pi_{\text{new}} = \arg\min_{\pi' \in \Pi} \mathrm{D}_{\mathrm{KL}}\left(\pi'(\,\cdot\,|\mathbf{s}_t) \parallel \exp\left(Q^{\pi_{\text{old}}}(\mathbf{s}_t, \,\cdot\,) - \log Z^{\pi_{\text{old}}}(\mathbf{s}_t)\right)\right). \tag{6}$$

For this choice of projection, we can show that the new, projected policy has a higher value than the old policy with respect to the objective in Equation 1. We formalize this result in Lemma 2.

Lemma 2 (Soft Policy Improvement)**.** *Let* $\pi_{\text{old}} \in \Pi$ *and let* π_{new} *be the optimizer of the minimization problem defined in Equation 6. Then* $Q^{\pi_{\text{new}}}(\mathbf{s}_t, \mathbf{a}_t) \geq Q^{\pi_{\text{old}}}(\mathbf{s}_t, \mathbf{a}_t)$ *for all* $(\mathbf{s}_t, \mathbf{a}_t) \in \mathcal{S} \times \mathcal{A}$*.*

Proof. See the appendix. □

The full soft policy iteration algorithm, which we summarize in Algorithm 1 alternates between the soft policy evaluation and the soft policy improvement steps, and it will provably converge to the optimal maximum entropy policy among the policies in Π (Theorem 1). Although this algorithm will provably find the optimal solution, we can perform it in its exact form only in the tabular case. Therefore, we will next approximate the algorithm for continuous domains, where we need to rely on function approximator to represent the Q-values, and running the two steps until convergence would be computationally too expensive. The approximation gives rise to a new practical algorithm, called soft actor-critic.

Theorem 1 (Soft Policy Iteration)**.** *Algorithm 1 converges, and at the convergence,* $\pi_{\text{new}}(\mathbf{a}_t|\mathbf{s}_t) = \max_{\pi' \in \Pi} Q^{\pi'}(\mathbf{s}_t, \mathbf{a}_t)$ *for all* $(\mathbf{s}_t, \mathbf{a}_t) \in \mathcal{S} \times \mathcal{A}$*.*

Proof. See the appendix. □

4.2 Soft Actor-Critic

As discussed above, large, continuous domains require us to derive a practical approximation to soft policy iteration. To that end, we will use function approximators for both the Q-function and policy, and instead of running evaluation and improvement to convergence, alternate between optimizing both networks with stochastic gradient descent. For the remainder of this paper, we will consider a parameterized state value function $V_\psi(\mathbf{s}_t)$, soft Q-function $Q_\theta(\mathbf{s}_t, \mathbf{a}_t)$, and a tractable policy $\pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$. The parameters of these networks are ψ, θ, and ϕ. In the following, we will derive update rules for these parameter vectors.

The state value function approximates the soft value. There is no need in principle to include a separate function approximator for the state value, since it is related to the Q-function and policy according to $Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$. This quantity can be evaluated using actions sampled from π_ϕ, but in practice, including a separate function approximator for the soft value substantially stabilizes training, and is convenient to train simultaneously with the other networks. The soft value function is trained to minimize the squared residual error

$$J_V(\psi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}}\left[\frac{1}{2}\left(V_\psi(\mathbf{s}_t) - \mathbb{E}_{\mathbf{a}_t \sim \pi_\phi}\left[Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)\right]\right)^2\right], \tag{7}$$

where $\mathcal{D}$ is the distribution of previously sampled states and actions, or a replay buffer. The gradient of Equation 7 can be estimated with an unbiased estimator

$$\hat{\nabla}_\psi J_V(\psi) = \nabla_\psi V_\psi(\mathbf{s}_t) \left(V_\psi(\mathbf{s}_t) - Q_\theta(\mathbf{s}_t, \mathbf{a}_t) + \log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t) \right), \tag{8}$$

where the actions are sampled according to the current policy, instead of the replay buffer. The soft Q-function parameters can be trained to minimize the soft Bellman residual

$$J_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\mathcal{D}} \left[\frac{1}{2} \left(Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \left(r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \, \mathbb{E}_{\mathbf{s}_{t+1}\sim p_\mathbf{s}} \left[V_\psi(\mathbf{s}_t) \right] \right) \right)^2 \right], \tag{9}$$

which again can be optimized with stochastic unbiased gradients

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(\mathbf{a}_t, \mathbf{s}_t) \left(Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - r(\mathbf{s}_t, \mathbf{a}_t) - \gamma V_\psi(\mathbf{s}_t) \right). \tag{10}$$

Finally, the policy parameters can be learned by directly minimizing the KL-divergence in Equation 6, which we reproduce here in parametrized form for completeness

$$J_\pi(\phi) = \mathrm{D}_{\mathrm{KL}} \left(\pi_\phi(\, \cdot \, |\mathbf{s}_t) \parallel \exp\left(Q_\theta(\mathbf{s}_t, \, \cdot \,) - \log Z_\theta(\mathbf{s}_t) \right) \right). \tag{11}$$

There are several options for minimizing J_π, depending on the choice of the policy class. For simple distributions, such as Gaussians, we can use the reparametrization trick, which allows us to back-propagate the gradient from the critic network and leads to a DDPG-style estimator. However, if the policy depends on discrete latent variables, such as is the case for mixture models, the reparametrization trick cannot be used. We therefore propose to use a likelihood ratio gradient estimator:

$$\nabla_\phi J_\pi(\phi) = \mathbb{E}_{\mathbf{a}_t\sim\pi_\phi} \left[\nabla_\phi \log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t) \left(\log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t) + 1 - Q_\theta(\mathbf{s}_t, \mathbf{a}_t) + \log Z_\theta(\mathbf{s}_t) + b(\mathbf{s}_t) \right) \right], \tag{12}$$

where $b(\mathbf{s}_t)$ is a state-dependent baseline (Peters & Schaal, 2008). We can approximately center the learning signal and eliminate the intractable log-partition function by choosing $b(\mathbf{s}_t) = V_\psi(\mathbf{s}_t) - \log Z_\theta(\mathbf{s}_t) - 1$, which yields the final gradient estimator

$$\hat{\nabla}_\phi J_\pi(\phi) = \nabla_\phi \log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t) \left(\log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t) - Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - V_\psi(\mathbf{s}_t) \right). \tag{13}$$

The complete algorithm is described in Algorithm 2. The method alternates between collecting experience from the environment with the current policy and updating the function approximators using the stochastic gradients on batches sampled from a replay buffer. This is feasible because both value estimators and the policy can be trained entirely on off-policy data. The algorithm is agnostic to the parameterization of the policy, as long as it can be evaluated for any arbitrary state-action tuple. We will next suggest a practical parameterization for the policy, based on Gaussian mixtures.

Algorithm 2: Soft Actor-Critic

```
Initialize parameter vectors ψ, θ, φ.
for each iteration do
    for each environment step do
        a_t ~ π_φ(a_t|s_t)
        s_{t+1} ~ p_s(s_{t+1}|s_t, a_t)
        D ← D ∪ {(s_t, a_t, r(s_t, a_t), s_{t+1})}.
    end
    for each gradient step do
        ψ ← ψ − λ_V ∇̂_ψ J_V(ψ)
        θ ← θ − λ_Q ∇̂_θ J_Q(θ)
        φ ← φ − λ_π ∇̂_φ J_π(φ)
    end
end
```

4.3 Soft Actor-Critic with Gaussian Mixtures

Although we could use a simple policy represented by a Gaussian, as is common in prior work, the maximum entropy framework aims to maximize the randomness of the learned policy. Therefore, a more expressive but still tractable distribution can endow our method with more effective exploration and robustness, which are the typically cited benefits of entropy maximization (Ziebart, 2010). To that end, we propose a practical multimodal representation based on a mixture of K Gaussians. This can approximate any distribution to arbitrary precision as $K \to \infty$, but even for practical numbers of mixture elements, it can provide a very expressive distribution in medium-dimensional action spaces. Although the complexity of evaluating or sampling from the resulting distribution scales linearly in K, our experiments indicates that a relatively small number of mixture components, typically just two or four, is sufficient to achieve high performance, thus making the algorithm scalable to complex domains with high-dimensional action spaces.

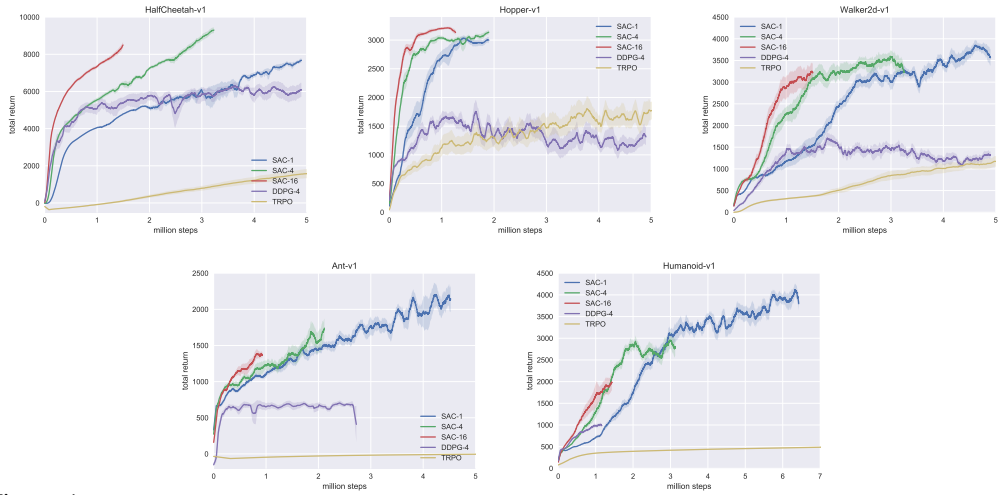

Figure 1: Training curves on continuous control benchmarks. Note that SAC attains the best results in all tasks, compared to both on-policy and off-policy methods. Furthermore, additional gradient steps per time step increase performance up to 16 steps.

We define the policy as

$$\pi_\phi(\mathbf{a}_t|\mathbf{s}_t) = \frac{1}{\sum_i w_i^\phi} \sum_{i=1}^{K} w_i^\phi(\mathbf{s}_t) \mathcal{N}\left(\mathbf{a}_t; \mu_i^\phi(\mathbf{s}_t), \Sigma_i^\phi(\mathbf{s}_t)\right), \tag{14}$$

where w_i^ϕ, μ_i^ϕ, Σ_i^ϕ are the unnormalized mixture weights, means, and covariances, respectively, which all can depend on $\mathbf{s}_t$ in complex ways if expressed as neural networks. Note that, in contrast to soft Q-learning (Haarnoja et al., 2017), our algorithm does *not* assume that the policy can accurately approximate the optimal exponentiated Q-function distribution. The convergence result for soft policy iteration holds even for policies that are restricted to a policy class, in contrast to prior methods of this type.

5 Experiments

The goal of our experimental evaluation is to understand how the sample complexity and stability of our method compares with prior off-policy and on-policy deep reinforcement learning algorithms. To that end, we evaluate on a range of challenging continuous control tasks from the OpenAI gym benchmark suite (Brockman et al., 2016). Although the easier tasks in this benchmark suite can be solved by a wide range of different algorithms, the more complex benchmarks, such as the 17 dimensional Humanoid-v1 task, are exceptionally difficult to solve with off-policy algorithms (Duan et al., 2016). The stability of the algorithm also plays a large role in performance: easier tasks make it more practical to tune hyperparameters to achieve good results, while the already narrow basins of effective hyperparameters become prohibitively small for the more sensitive algorithms on the most high-dimensional benchmarks, leading to poor performance (Gu et al., 2016).

In our comparisons, we compare to trust region policy optimization (TRPO) (Schulman et al., 2015), a stable and effective on-policy policy gradient algorithm, as well as deep deterministic policy gradient (DDGP) (Lillicrap et al., 2015), an algorithm that is regarded as one of the more efficient off-policy deep RL methods (Duan et al., 2016). Although DDPG is very efficient, it is also known to be more sensitive to hyperparameter settings, which limits its effectiveness on complex tasks (Gu et al., 2016; Henderson et al., 2017). Figure 5 shows the total average reward of evaluation rollouts during training for the various methods. Exploration noise was turned off for evaluation for DDPG and TRPO. For SAC, which does not explicitly inject exploration noise, we heuristically chose the action to be the mean of the mixture component that achieves the highest Q-value.

Comparative evaluation. The results show that, overall, SAC substantially outperforms DDPG on all of the benchmark tasks, in terms of both sample efficiency and final performance, and learns substantially faster than TRPO. On the hardest tasks, Humanoid-v1 and HumanoidStandup-v1, DDPG is unable to make any progress, a result that is corroborated by prior work (Gu et al., 2016; Duan et al., 2016). SAC still learns substantially faster than TRPO on these tasks, likely as a consequence of improved stability. Poor stability is exacerbated by high-dimensional and complex tasks.

The quantitative results attained by SAC in our experiments also compare very favorably to results reported by other methods in prior work (Duan et al., 2016; Gu et al., 2016; Henderson et al., 2017), indicating that both the sample efficiency and final performance of SAC on these benchmark tasks exceeds the state of the art. In Figure 2, we show the performance for multiple individual seeds of both DDPG and SAC (Figure 5 shows the average over seeds). The individual seeds attain much more consistent performance with SAC, while DDPG exhibits very high variability across seeds, indicating substantially worse stability. Note that the different seeds of SAC are even more consistent than TRPO, an algorithm that is generally known to be among the more stable (though less efficient) deep RL methods. Even though TRPO does not make perceivable progress for some tasks within the range depicted in the figures, it will eventually solve all of them after substantially more iterations.

Gradient steps per time step. For both DDPG and SAC, we experimented with the number of actor and critic gradient steps per time step of the algorithm. Prior work has observed that increasing the number of gradient steps for DDPG tends to improve sample efficiency (Gu et al., 2017; Popov et al., 2017). Our experiments agree with this observation. We empirically found that 4 gradient updates per time step resulted in the best performance for DDPG. Interestingly, SAC was able to effectively benefit from substantially larger numbers of gradient updates, with performance increasingly steadily until 16 gradient updates per step. In Figure 5, we show performance for SAC with 1, 4, and 16 gradient steps.

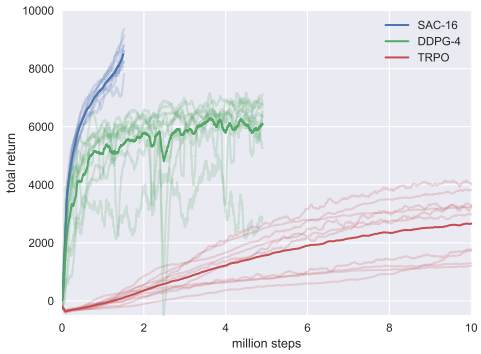

Figure 2: Performance of the individual seeds on the half-cheetah benchmark. Note that all of the SAC seeds attain good performance, while DDPG and TRPO exhibit considerable variability across seeds. The TRPO distribution is in fact clearly bimodal, with one set of successful seeds, and one set of seeds that learn much slower. Several of the DDPG seeds oscillate and fail to improve after the first five million steps. The independent curves show the moving average of a hundred evaluation episodes.

Mixture components. We experimented with different numbers of mixture components for the Gaussian mixture policy for SAC. We found that the number of mixture elements generally did not have a very large effect on performance, though 2 or 4 components typically attained the best results.

Importance of entropy regularization. It is in principle possible to implement a "hard" variant of SAC, by dropping the entropy regularization term. The result is a standard off-policy actor-critic algorithm that uses a Q-function for off-policy learning. This algorithm closely resembles DDPG, but with a stochastic policy trained via likelihood ratio gradients. We found that this method achieved very poor performance across all tasks, and were unable to tune it to train effectively. This suggests that the inclusion of entropy regularization is a critical component of our method.

6 Conclusion

We presented soft actor-critic (SAC), an off-policy maximum entropy deep reinforcement learning algorithm that provides sample-efficient learning while retaining the benefits of entropy maximization and stability. Our theoretical results derive soft policy iteration, which we show to converge to the optimal policy. From this result, we can formulate a soft actor-critic algorithm, and we empirically show that it outperforms state-of-the-art model-free deep RL methods, including the off-policy DDPG algorithm and the on-policy TRPO algorithm. In fact, the sample efficiency of this approach actually exceeds that of DDPG by a substantial margin. Our results suggest that stochastic, entropy maximizing reinforcement learning algorithms can provide a promising avenue for improved robustness and stability, and further exploration of maximum entropy methods, including methods that incorporate second order information (e.g., trust regions (Schulman et al., 2015) or Kronecker factorization (Johns et al., 2007)) or more expressive policy classes is an exciting avenue for future work.

REFERENCES

A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.

S. Bhatnagar, D. Precup, D. Silver, R. S Sutton, H. R Maei, and C. Szepesvári. Convergent temporal-difference learning with arbitrary smooth function approximation. In *Advances in Neural Information Processing Systems*, pp. 1204–1212, 2009.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI gym. *arXiv preprint arXiv:1606.01540*, 2016.

Y. Duan, R. Chen, X. Houthooft, J. Schulman, and P. Abbeel. Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016.

R. Fox, A. Pakman, and N. Tishby. Taming the noise in reinforcement learning via soft updates. In *Conf. on Uncertainty in Artificial Intelligence*, 2016.

S. Gu, T. Lillicrap, Z. Ghahramani, R. E. Turner, and S. Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. *arXiv preprint arXiv:1611.02247*, 2016.

S. Gu, E. Holly, T. Lillicrap, and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 3389–3396. IEEE, 2017.

T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*, 2017.

P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560*, 2017.

J. Johns, S. Mahadevan, and C. Wang. Compact spectral bases for value function approximation using kronecker factorization. In *AAAI*, volume 7, pp. 559–564, 2007.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Int. Conf. on Machine Learning*, 2016.

O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans. Bridging the gap between value and policy based reinforcement learning. *arXiv preprint arXiv:1702.08892*, 2017a.

O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans. Trust-PCL: An off-policy trust region method for continuous control. *arXiv preprint arXiv:1707.01891*, 2017b.

B. O'Donoghue, R. Munos, K. Kavukcuoglu, and V. Mnih. PGQ: Combining policy gradient and Q-learning. *arXiv preprint arXiv:1611.01626*, 2016.

J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

I. Popov, N. Heess, T. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez, and M. Riedmiller. Data-efficient deep reinforcement learning for dexterous manipulation. *arXiv preprint arXiv:1704.03073*, 2017.

K. Rawlik, M. Toussaint, and S. Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. *Proceedings of Robotics: Science and Systems VIII*, 2012.

J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz. Trust region policy optimization. In *Int. Conf on Machine Learning*, pp. 1889–1897, 2015.

J. Schulman, P. Abbeel, and X. Chen. Equivalence between policy gradients and soft Q-learning. *arXiv preprint arXiv:1704.06440*, 2017.

D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *Int. Conf on Machine Learning*, 2014.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, Jan 2016. ISSN 0028-0836. Article.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

P. Thomas. Bias in natural actor-critic algorithms. In *Int. Conf. on Machine Learning*, pp. 441–448, 2014.

E. Todorov. General duality between optimal control and estimation. In *IEEE Conf. on Decision and Control*, pp. 4286–4292. IEEE, 2008.

M. Toussaint. Robot trajectory optimization using approximate inference. In *Int. Conf. on Machine Learning*, pp. 1049–1056. ACM, 2009.

B. D. Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. PhD thesis, 2010.

B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, pp. 1433–1438, 2008.

A INFINITE HORIZON DISCOUNTED MAXIMUM ENTROPY OBJECTIVE

$$J(\pi) = \sum_{t=0}^{\infty} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} \left[\sum_{l=t}^{\infty} \gamma^{l-t} \, \mathbb{E}_{\mathbf{s}_l \sim p_{\mathbf{s}}, \mathbf{a}_l \sim \pi} \left[r(\mathbf{s}_t, \mathbf{a}_t) + \alpha \mathcal{H}(\pi(\cdot \,|\mathbf{s}_t)) | \mathbf{s}_t, \mathbf{a}_t \right] \right] \tag{15}$$

B PROOFS

B.1 LEMMA 1

Proof. Define the entropy augmented reward as $r_{\mathcal{H}}(\mathbf{s}_t, \mathbf{a}_t) \triangleq r(\mathbf{s}_t, \mathbf{a}_t) + \mathcal{H}\left(\pi(\cdot \,|\mathbf{s}_t)\right)$ and rewrite the update rule as

$$Q(\mathbf{s}_t, \mathbf{a}_t) \leftarrow r_{\mathcal{H}}(\mathbf{s}_t, \mathbf{a}_t) + \gamma \, \mathbb{E}_{\mathbf{a} \sim \pi} \left[Q(\mathbf{s}_{t+1}, \mathbf{a}) \right] \tag{16}$$

and apply the standard convergence results for policy evaluation (Sutton & Barto, 1998). □

B.2 LEMMA 2

Proof. We will drop the state and action arguments from the following derivation for improved readability. Let $\pi_{\text{old}} \in \Pi$ and let $Q^{\pi_{\text{old}}}$ and $V^{\pi_{\text{old}}}$ be the corresponding soft state-action value and soft state value, and let π_{new} be defined as

$$\begin{aligned} \pi_{\text{new}} &= \arg\min_{\pi' \in \Pi} \mathrm{D}_{\mathrm{KL}} \left(\pi' \parallel \exp\left(Q^{\pi_{\text{old}}} - \log Z^{\pi_{\text{old}}} \right) \right) \\ &= \arg\min_{\pi' \in \Pi} J_{\pi_{\text{old}}}(\pi'). \end{aligned} \tag{17}$$

It must be the case that $J_{\pi_{\text{old}}}(\pi_{\text{new}}) \leq J_{\pi_{\text{old}}}(\pi_{\text{old}})$, since we can always choose $\pi_{\text{new}} = \pi_{\text{old}} \in \Pi$. Hence

$$\mathbb{E}_{\pi_{\text{new}}} \left[\log \pi_{\text{new}} - Q^{\pi_{\text{old}}} + \log Z^{\pi_{\text{old}}} \right] \leq \mathbb{E}_{\pi_{\text{old}}} \left[\log \pi_{\text{old}} - Q^{\pi_{\text{old}}} + \log Z^{\pi_{\text{old}}} \right], \tag{18}$$

or

$$\mathbb{E}_{\pi_{\text{new}}} \left[Q^{\pi_{\text{old}}} - \log \pi_{\text{new}} \right] \geq V^{\pi_{\text{old}}}. \tag{19}$$

Next, consider the soft Bellman equation:

$$\begin{aligned} Q^{\pi_{\text{old}}} &= r + \gamma \, \mathbb{E}_{p_{\mathbf{s}}} \left[V^{\pi_{\text{old}}} \right] \\ &\leq r + \gamma \, \mathbb{E}_{p_{\mathbf{s}}} \left[\mathbb{E}_{\pi_{\text{new}}} \left[Q^{\pi_{\text{old}}} - \log \pi_{\text{new}} \right] \right] \\ &\vdots \\ &\leq Q^{\pi_{\text{new}}}. \end{aligned} \tag{20}$$

We keep expanding $Q^{\pi_{\text{old}}}$ on the RHS, which converges to $Q^{\pi_{\text{new}}}$ by Lemma 1. □

B.3 THEOREM 1

Proof. Let π_i be the policy at iteration i. By Lemma 2, the sequence Q^{π_i} is monotonically increasing. Since Q is bounded, the sequence converges to some π^*. We will still need to show that π^* is indeed optimal. At convergence, it must be case that $J_{\pi^*}(\pi^*) < J_{\pi^*}(\pi)$ for all $\pi \in \Pi$, $\pi \neq \pi^*$. Using the same iterative argument as in the proof of Lemma 2, we get $Q^{\pi^*} > Q^{\pi}$, that is, the soft value of any other policy in Π is lower than that of the converged policy. Hence π^* is optimal in Π. □