Reinforcement Learning with Deep Energy-Based Policies

Tuomas Haarnoja [*1] **Haoran Tang** [*2] **Pieter Abbeel** [1 3 4] **Sergey Levine** [1]

Abstract

We propose a method for learning expressive energy-based policies for continuous states and actions, which has been feasible only in tabular domains before. We apply our method to learning maximum entropy policies, resulting into a new algorithm, called soft Q-learning, that expresses the optimal policy via a Boltzmann distribution. We use the recently proposed amortized Stein variational gradient descent to learn a stochastic sampling network that approximates samples from this distribution. The benefits of the proposed algorithm include improved exploration and compositionality that allows transferring skills between tasks, which we confirm in simulated experiments with swimming and walking robots. We also draw a connection to actor-critic methods, which can be viewed performing approximate inference on the corresponding energy-based model.

1. Introduction

Deep reinforcement learning (deep RL) has emerged as a promising direction for autonomous acquisition of complex behaviors (Mnih et al., 2015; Silver et al., 2016), due to its ability to process complex sensory input (Jaderberg et al., 2016) and to acquire elaborate behavior skills using general-purpose neural network representations (Levine et al., 2016). Deep reinforcement learning methods can be used to optimize deterministic (Lillicrap et al., 2015) and stochastic (Schulman et al., 2015a; Mnih et al., 2016) policies. However, most deep RL methods operate on the conventional deterministic notion of optimality, where the optimal solution, at least under full observability, is always a deterministic policy (Sutton & Barto, 1998). Although stochastic policies are desirable for exploration, this exploration is typically attained heuristically, for example by injecting noise (Silver et al., 2014; Lillicrap et al., 2015; Mnih et al., 2015) or initializing a stochastic policy with high entropy (Kakade, 2002; Schulman et al., 2015a; Mnih et al., 2016).

In some cases, we might actually prefer to learn stochastic behaviors. In this paper, we explore two potential reasons for this: exploration in the presence of multimodal objectives, and compositionality attained via pretraining. Other benefits include robustness in the face of uncertain dynamics (Ziebart, 2010), imitation learning (Ziebart et al., 2008), and improved convergence and computational properties (Gu et al., 2016a). Multi-modality also has application in real robot tasks, as demonstrated in (Daniel et al., 2012). However, in order to learn such policies, we must define an objective that promotes stochasticity.

In which cases is a stochastic policy actually the optimal solution? As discussed in prior work, a stochastic policy emerges as the optimal answer when we consider the connection between optimal control and probabilistic inference (Todorov, 2008). While there are multiple instantiations of this framework, they typically include the cost or reward function as an additional factor in a factor graph, and infer the optimal conditional distribution over actions conditioned on states. The solution can be shown to optimize an entropy-augmented reinforcement learning objective or to correspond to the solution to a maximum entropy learning problem (Toussaint, 2009). Intuitively, framing control as inference produces policies that aim to capture not only the single deterministic behavior that has the lowest cost, but the entire range of low-cost behaviors, explicitly maximizing the entropy of the corresponding policy. Instead of learning the best way to perform the task, the resulting policies try to learn *all* of the ways of performing the task. It should now be apparent why such policies might be preferred: if we can learn all of the ways that a given task might be performed, the resulting policy can serve as a good initialization for finetuning to a more specific behavior (e.g. first learning all the ways that a robot could move forward, and then using this as an initialization to learn separate running and bounding skills); a better exploration mechanism for seeking out the best mode in a multimodal reward landscape; and a more robust behavior in the

[*]Equal contribution [1]UC Berkeley, Department of Electrical Engineering and Computer Sciences [2]UC Berkeley, Department of Mathematics [3]OpenAI [4]International Computer Science Institute. Correspondence to: Haoran Tang <hrtang@math.berkeley.edu>, Tuomas Haarnoja <haarnoja@berkeley.edu>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

face of adversarial perturbations, where the ability to perform the same task in multiple different ways can provide the agent with more options to recover from perturbations.

Unfortunately, solving such maximum entropy stochastic policy learning problems in the general case is challenging. A number of methods have been proposed, including Z-learning (Todorov, 2007), maximum entropy inverse RL (Ziebart et al., 2008), approximate inference using message passing (Toussaint, 2009), Ψ-learning (Rawlik et al., 2012), and G-learning (Fox et al., 2016), as well as more recent proposals in deep RL such as PGQ (O'Donoghue et al., 2016), but these generally operate either on simple tabular representations, which are difficult to apply to continuous or high-dimensional domains, or employ a simple parametric representation of the policy distribution, such as a conditional Gaussian. Therefore, although the policy is optimized to perform the desired skill in many different ways, the resulting distribution is typically very limited in terms of its representational power, even if the *parameters* of that distribution are represented by an expressive function approximator, such as a neural network.

How can we extend the framework of maximum entropy policy search to arbitrary policy distributions? In this paper, we borrow an idea from energy-based models, which in turn reveals an intriguing connection between Q-learning, actor-critic algorithms, and probabilistic inference. In our method, we formulate a stochastic policy as a (conditional) energy-based model (EBM), with the energy function corresponding to the "soft" Q-function obtained when optimizing the maximum entropy objective. In high-dimensional continuous spaces, sampling from this policy, just as with any general EBM, becomes intractable. We borrow from the recent literature on EBMs to devise an approximate sampling procedure based on training a separate sampling network, which is optimized to produce unbiased samples from the policy EBM. This sampling network can then be used both for updating the EBM and for action selection. In the parlance of reinforcement learning, the sampling network is the actor in an actor-critic algorithm. This reveals an intriguing connection: entropy regularized actor-critic algorithms can be viewed as approximate Q-learning methods, with the actor serving the role of an approximate sampler from an intractable posterior. We explore this connection further in the paper, and in the course of this discuss connections to popular deep RL methods such as deterministic policy gradient (DPG) (Silver et al., 2014; Lillicrap et al., 2015), normalized advantage functions (NAF) (Gu et al., 2016b), and PGQ (O'Donoghue et al., 2016).

The principal contribution of this work is a tractable, efficient algorithm for optimizing arbitrary multimodal stochastic policies represented by energy-based models, as well as a discussion that relates this method to other recent algorithms in RL and probabilistic inference. In our experimental evaluation, we explore two potential applications of our approach. First, we demonstrate improved exploration performance in tasks with multi-modal reward landscapes, where conventional deterministic or unimodal methods are at high risk of falling into suboptimal local optima. Second, we explore how our method can be used to provide a degree of compositionality in reinforcement learning by showing that stochastic energy-based policies can serve as a much better initialization for learning new skills than either random policies or policies pretrained with conventional maximum reward objectives.

2. Preliminaries

In this section, we will define the reinforcement learning problem that we are addressing and briefly summarize the maximum entropy policy search objective. We will also present a few useful identities that we will build on in our algorithm, which will be presented in Section 3.

2.1. Maximum Entropy Reinforcement Learning

We will address learning of maximum entropy policies with approximate inference for reinforcement learning in continuous action spaces. Our reinforcement learning problem can be defined as policy search in an infinite-horizon Markov decision process (MDP), which consists of the tuple $(\mathcal{S}, \mathcal{A}, p_{\mathbf{s}}, r)$, The state space $\mathcal{S}$ and action space $\mathcal{A}$ are assumed to be continuous, and the state transition probability $p_{\mathbf{s}} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, \infty)$ 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 $r : \mathcal{S} \times \mathcal{A} \rightarrow [r_{\min}, r_{\max}]$ 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 $\pi(\mathbf{a}_t|\mathbf{s}_t)$.

Our goal is to learn a policy $\pi(\mathbf{a}_t|\mathbf{s}_t)$. We can define the standard reinforcement learning objective in terms of the above quantities as

$$\pi^*_{\text{std}} = \arg\max_\pi \sum_t \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} \left[r(\mathbf{s}_t, \mathbf{a}_t) \right]. \tag{1}$$

Maximum entropy RL augments the reward with an entropy term, such that the optimal policy aims to maximize its entropy at each visited state:

$$\pi^*_{\text{MaxEnt}} = \arg\max_\pi \sum_t \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} \left[r(\mathbf{s}_t, \mathbf{a}_t) + \alpha \mathcal{H}(\pi(\cdot|\mathbf{s}_t)) \right], \tag{2}$$

where α is an optional but convenient parameter that can be used to determine the relative importance of entropy and reward.[1] Optimization problems of this type have been explored in a number of prior works (Kappen, 2005; Todorov,

[1] In principle, $1/\alpha$ can be folded into the reward function, eliminating the need for an explicit multiplier, but in practice, it is often convenient to keep α as a hyperparameter.

2007; Ziebart et al., 2008), which are covered in more detail in Section 4. Note that this objective differs qualitatively from the behavior of Boltzmann exploration (Sallans & Hinton, 2004) and PGQ (O'Donoghue et al., 2016), which greedily maximize entropy at the current time step, but do not explicitly optimize for policies that aim to reach *states* where they will have high entropy in the future. This distinction is crucial, since the maximum entropy objective can be shown to maximize the entropy of the entire trajectory distribution for the policy π, while the greedy Boltzmann exploration approach does not (Ziebart et al., 2008; Levine & Abbeel, 2014). As we will discuss in Section 5, this maximum entropy formulation has a number of benefits, such as improved exploration in multimodal problems and better pretraining for later adaptation.

If we wish to extend either the conventional or the maximum entropy RL 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. In the context of policy search algorithms, the use of a discount factor is actually a somewhat nuanced choice, and writing down the precise objective that is optimized when using the discount factor is non-trivial (Thomas, 2014). We defer the full derivation of the discounted objective to Appendix A, since it is unwieldy to write out explicitly, but we will use the discount γ in the following derivations and in our final algorithm.

2.2. Soft Value Functions and Energy-Based Models

Optimizing the maximum entropy objective in (2) provides us with a framework for training stochastic policies, but we must still choose a representation for these policies. The choices in prior work include discrete multinomial distributions (O'Donoghue et al., 2016) and Gaussian distributions (Rawlik et al., 2012). However, if we want to use a very general class of distributions that can represent complex, multimodal behaviors, we can instead opt for using general energy-based policies of the form

$$\pi(\mathbf{a}_t|\mathbf{s}_t) \propto \exp\left(-\mathcal{E}(\mathbf{s}_t, \mathbf{a}_t)\right), \tag{3}$$

where $\mathcal{E}$ is an energy function that could be represented, for example, by a deep neural network. If we use a universal function approximator for $\mathcal{E}$, we can represent any distribution $\pi(\mathbf{a}_t|\mathbf{s}_t)$. There is a close connection between such energy-based models and *soft* versions of value functions and Q-functions, where we set $\mathcal{E}(\mathbf{s}_t, \mathbf{a}_t) = -\frac{1}{\alpha} Q_{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t)$ and use the following theorem:

Theorem 1. *Let the soft Q-function be defined by*

$$Q^*_{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t) = r_t + \mathbb{E}_{(\mathbf{s}_{t+1}, \ldots) \sim \rho_\pi}\left[\sum_{l=1}^{\infty} \gamma^l \left(r_{t+l} + \alpha \mathcal{H}\left(\pi^*_{\text{MaxEnt}}(\cdot|\mathbf{s}_{t+l})\right)\right)\right], \tag{4}$$

and soft value function by

$$V^*_{\text{soft}}(\mathbf{s}_t) = \alpha \log \int_{\mathcal{A}} \exp\left(\frac{1}{\alpha} Q^*_{\text{soft}}(\mathbf{s}_t, \mathbf{a}')\right) d\mathbf{a}'. \tag{5}$$

Then the optimal policy for (2) is given by

$$\pi^*_{\text{MaxEnt}}(\mathbf{a}_t|\mathbf{s}_t) = \exp\left(\tfrac{1}{\alpha}(Q^*_{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t) - V^*_{\text{soft}}(\mathbf{s}_t))\right). \tag{6}$$

Proof. See Appendix A.1 as well as (Ziebart, 2010). □

Theorem 1 connects the maximum entropy objective in (2) and energy-based models, where $\frac{1}{\alpha} Q_{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t)$ acts as the negative energy, and $\frac{1}{\alpha} V_{\text{soft}}(\mathbf{s}_t)$ serves as the log-partition function. As with the standard Q-function and value function, we can relate the Q-function to the value function at a future state via a soft Bellman equation:

Theorem 2. *The soft Q-function in (4) satisfies the soft Bellman equation*

$$Q^*_{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t) = r_t + \gamma\, \mathbb{E}_{\mathbf{s}_{t+1} \sim p_{\mathbf{s}}}\left[V^*_{\text{soft}}(\mathbf{s}_{t+1})\right], \tag{7}$$

where the soft value function V^*_{soft} is given by (5).

Proof. See Appendix A.2, as well as (Ziebart, 2010). □

The soft Bellman equation is a generalization of the conventional (hard) equation, where we can recover the more standard equation as $\alpha \to 0$, which causes (5) to approach a hard maximum over the actions. In the next section, we will discuss how we can use these identities to derive a Q-learning style algorithm for learning maximum entropy policies, and how we can make this practical for arbitrary Q-function representations via an approximate inference procedure.

3. Training Expressive Energy-Based Models via Soft Q-Learning

In this section, we will present our proposed reinforcement learning algorithm, which is based on the soft Q-function described in the previous section, but can be implemented via a tractable stochastic gradient descent procedure with approximate sampling. We will first describe the general case of soft Q-learning, and then present the inference procedure that makes it tractable to use with deep neural network representations in high-dimensional continuous state and action spaces. In the process, we will relate this Q-learning procedure to inference in energy-based models and actor-critic algorithms.

3.1. Soft Q-Iteration

We can obtain a solution to (7) by iteratively updating estimates of V^*_{soft} and Q^*_{soft}. This leads to a fixed-point iteration that resembles Q-iteration:

Theorem 3. *Soft Q-iteration. Let $Q_{\text{soft}}(\cdot, \cdot)$ and $V_{\text{soft}}(\cdot)$ be bounded and assume that $\int_{\mathcal{A}} \exp(\frac{1}{\alpha} Q_{\text{soft}}(\cdot, \mathbf{a}')) d\mathbf{a}' < \infty$*

and that $Q^*_{\text{soft}} < \infty$ exists. Then the fixed-point iteration

$$Q_{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t) \leftarrow r_t + \gamma\, \mathbb{E}_{\mathbf{s}_{t+1} \sim p_{\mathbf{s}}} \left[V_{\text{soft}}(\mathbf{s}_{t+1}) \right], \ \forall \mathbf{s}_t, \mathbf{a}_t \quad (8)$$

$$V_{\text{soft}}(\mathbf{s}_t) \leftarrow \alpha \log \int_{\mathcal{A}} \exp\left(\frac{1}{\alpha} Q_{\text{soft}}(\mathbf{s}_t, \mathbf{a}')\right) d\mathbf{a}', \ \forall \mathbf{s}_t \quad (9)$$

converges to Q^*_{soft} and V^*_{soft}, respectively.

Proof. See Appendix A.2 as well as (Fox et al., 2016). □

We refer to the updates in (8) and (9) as the soft Bellman backup operator that acts on the soft value function, and denote it by $\mathcal{T}$. The maximum entropy policy in (6) can then be recovered by iteratively applying this operator until convergence. However, there are several practicalities that need to be considered in order to make use of the algorithm. First, the soft Bellman backup cannot be performed exactly in continuous or large state and action spaces, and second, sampling from the energy-based model in (6) is intractable in general. We will address these challenges in the following sections.

3.2. Soft Q-Learning

This section discusses how the Bellman backup in Theorem 3 can be implemented in a practical algorithm that uses a finite set of samples from the environment, resulting in a method similar to Q-learning. Since the soft Bellman backup is a contraction (see Appendix A.2), the optimal value function is the fixed point of the Bellman backup, and we can find it by optimizing for a Q-function for which the soft Bellman error $|\mathcal{T}Q - Q|$ is minimized at all states and actions. While this procedure is still intractable due to the integral in (9) and the infinite set of all states and actions, we can express it as a stochastic optimization, which leads to a stochastic gradient descent update procedure. We will model the soft Q-function with a function approximator with parameters θ and denote it as $Q^\theta_{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t)$.

To convert Theorem 3 into a stochastic optimization problem, we first express the soft value function in terms of an expectation via importance sampling:

$$V^\theta_{\text{soft}}(\mathbf{s}_t) = \alpha \log \mathbb{E}_{q_{\mathbf{a}'}} \left[\frac{\exp\left(\frac{1}{\alpha} Q^\theta_{\text{soft}}(\mathbf{s}_t, \mathbf{a}')\right)}{q_{\mathbf{a}'}(\mathbf{a}')} \right], \quad (10)$$

where $q_{\mathbf{a}'}$ can be an arbitrary distribution over the action space. Second, by noting the identity $g_1(x) = g_2(x)\ \forall x \in \mathbb{X} \Leftrightarrow \mathbb{E}_{x \sim q}\left[(g_1(x) - g_2(x))^2\right] = 0$, where q can be any strictly positive density function on $\mathbb{X}$, we can express the soft Q-iteration in an equivalent form as minimizing

$$J_Q(\theta) = \mathbb{E}_{\mathbf{s}_t \sim q_{\mathbf{s}_t}, \mathbf{a}_t \sim q_{\mathbf{a}_t}} \left[\frac{1}{2} \left(\hat{Q}^{\bar{\theta}}_{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t) - Q^\theta_{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t) \right)^2 \right], \quad (11)$$

where $q_{\mathbf{s}_t}, q_{\mathbf{a}_t}$ are positive over $\mathcal{S}$ and $\mathcal{A}$ respectively, $\hat{Q}^{\bar{\theta}}_{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t) = r_t + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p_{\mathbf{s}}}[V^{\bar{\theta}}_{\text{soft}}(\mathbf{s}_{t+1})]$ is a *target* Q-value, with $V^{\bar{\theta}}_{\text{soft}}(\mathbf{s}_{t+1})$ given by (10) and θ being replaced by the target parameters, $\bar{\theta}$.

This stochastic optimization problem can be solved approximately using stochastic gradient descent using sampled states and actions. While the sampling distributions $q_{\mathbf{s}_t}$ and $q_{\mathbf{a}_t}$ can be arbitrary, we typically use real samples from rollouts of the current policy $\pi(\mathbf{a}_t|\mathbf{s}_t) \propto \exp\left(\frac{1}{\alpha} Q^\theta_{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t)\right)$. For $q_{\mathbf{a}'}$ we have more options. A convenient choice is a uniform distribution. However, this choice can scale poorly to high dimensions. A better choice is to use the current policy, which produces an unbiased estimate of the soft value as can be confirmed by substitution. This overall procedure yields an iterative approach that optimizes over the Q-values, which we summarize in Section 3.4.

However, in continuous spaces, we still need a tractable way to sample from the policy $\pi(\mathbf{a}_t|\mathbf{s}_t) \propto \exp\left(\frac{1}{\alpha} Q^\theta_{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t)\right)$, both to take on-policy actions and, if so desired, to generate action samples for estimating the soft value function. Since the form of the policy is so general, sampling from it is intractable. We will therefore use an approximate sampling procedure, as discussed in the following section.

3.3. Approximate Sampling and Stein Variational Gradient Descent (SVGD)

In this section we describe how we can approximately sample from the soft Q-function. Existing approaches that sample from energy-based distributions generally fall into two categories: methods that use Markov chain Monte Carlo (MCMC) based sampling (Sallans & Hinton, 2004), and methods that learn a stochastic sampling network trained to output approximate samples from the target distribution (Zhao et al., 2016; Kim & Bengio, 2016). Since sampling via MCMC is not tractable when the inference must be performed online (e.g. when executing a policy), we will use a sampling network based on Stein variational gradient descent (SVGD) (Liu & Wang, 2016) and amortized SVGD (Wang & Liu, 2016). Amortized SVGD has several intriguing properties: First, it provides us with a stochastic sampling network that we can query for extremely fast sample generation. Second, it can be shown to converge to an accurate estimate of the posterior distribution of an EBM. Third, the resulting algorithm, as we will show later, strongly resembles actor-critic algorithm, which provides for a simple and computationally efficient implementation and sheds light on the connection between our algorithm and prior actor-critic methods.

Formally, we want to learn a state-conditioned stochastic neural network $\mathbf{a}_t = f^\phi(\xi; \mathbf{s}_t)$, parametrized by ϕ, that maps noise samples ξ drawn from a normal Gaussian, or other arbitrary distribution, into unbiased action samples from the target EBM corresponding to Q^θ_{soft}. We denote the induced distribution of the actions as $\pi^\phi(\mathbf{a}_t|\mathbf{s}_t)$, and we want to find parameters ϕ so that the induced distribution

approximates the energy-based distribution in terms of the KL divergence

$$J_\pi(\phi; \mathbf{s}_t) = \mathrm{D}_{\mathrm{KL}}\left(\pi^\phi(\cdot|\mathbf{s}_t) \,\middle\|\, \exp\left(\frac{1}{\alpha}\left(Q^\theta_{\mathrm{soft}}(\mathbf{s}_t, \cdot) - V^\theta_{\mathrm{soft}}\right)\right)\right). \tag{12}$$

Suppose we "perturb" a set of independent samples $\mathbf{a}_t^{(i)} = f^\phi(\xi^{(i)}; \mathbf{s}_t)$ in appropriate directions $\Delta f^\phi(\xi^{(i)}; \mathbf{s}_t)$, the induced KL divergence can be reduced. Stein variational gradient descent (Liu & Wang, 2016) provides the most greedy directions as a functional

$$\Delta f^\phi(\cdot; \mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi^\phi}\left[\kappa(\mathbf{a}_t, f^\phi(\cdot; \mathbf{s}_t)) \nabla_{\mathbf{a}'} Q^\theta_{\mathrm{soft}}(\mathbf{s}_t, \mathbf{a}')\big|_{\mathbf{a}'=\mathbf{a}_t} + \alpha \, \nabla_{\mathbf{a}'} \kappa(\mathbf{a}', f^\phi(\cdot; \mathbf{s}_t))\big|_{\mathbf{a}'=\mathbf{a}_t}\right], \tag{13}$$

where κ is a kernel function (typically Gaussian, see details in Appendix D.1). To be precise, Δf^ϕ is the optimal direction in the reproducing kernel Hilbert space of κ, and is thus not strictly speaking the gradient of (12), but it turns out that we can set $\frac{\partial J_\pi}{\partial \mathbf{a}_t} \propto \Delta f^\phi$ as explained in (Wang & Liu, 2016). With this assumption, we can use the chain rule and backpropagate the Stein variational gradient into the policy network according to

$$\frac{\partial J_\pi(\phi; \mathbf{s}_t)}{\partial \phi} \propto \mathbb{E}_\xi\left[\Delta f^\phi(\xi; \mathbf{s}_t) \frac{\partial f^\phi(\xi; \mathbf{s}_t)}{\partial \phi}\right], \tag{14}$$

and use any gradient-based optimization method to learn the optimal sampling network parameters. The sampling network f^ϕ can be viewed as an actor in an actor-critic algorithm. We will discuss this connection in Section 4, but first we will summarize our complete maximum entropy policy learning algorithm.

3.4. Algorithm Summary

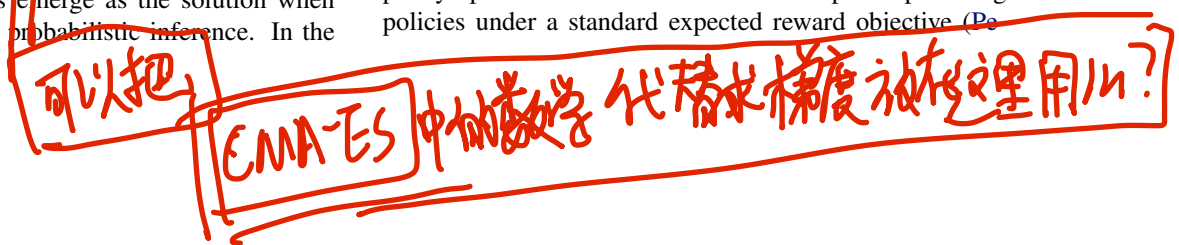
To summarize, we propose the soft Q-learning algorithm for learning maximum entropy policies in continuous domains. The algorithm proceeds by alternating between collecting new experience from the environment, and updating the soft Q-function and sampling network parameters. The experience is stored in a replay memory buffer $\mathcal{D}$ as standard in deep Q-learning (Mnih et al., 2013), and the parameters are updated using random minibatches from this memory. The soft Q-function updates use a delayed version of the target values (Mnih et al., 2015). For optimization, we use the ADAM (Kingma & Ba, 2015) optimizer and empirical estimates of the gradients, which we denote by $\hat{\nabla}$. The exact formulae used to compute the gradient estimates is deferred to Appendix C, which also discusses other implementation details, but we summarize an overview of soft Q-learning in Algorithm 1.

Algorithm 1 Soft Q-learning

$\theta, \phi \sim$ some initialization distributions.
Assign target parameters: $\bar{\theta} \leftarrow \theta$, $\bar{\phi} \leftarrow \phi$.
$\mathcal{D} \leftarrow$ empty replay memory.
for each epoch **do**
  for each t **do**
    Collect experience
    Sample an action for $\mathbf{s}_t$ using f^ϕ:
      $\mathbf{a}_t \leftarrow f^\phi(\xi; \mathbf{s}_t)$ where $\xi \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$.
    Sample next state from the environment:
      $\mathbf{s}_{t+1} \sim p_{\mathbf{s}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$.
    Save the new experience in the replay memory:
      $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$.
    Sample a minibatch from the replay memory
    $\{(\mathbf{s}_t^{(i)}, \mathbf{a}_t^{(i)}, r_t^{(i)}, \mathbf{s}_{t+1}^{(i)})\}_{i=0}^N \sim \mathcal{D}$.
    Update the soft Q-function parameters
    Sample $\{\mathbf{a}^{(i,j)}\}_{j=0}^M \sim q_{\mathbf{a}'}$ for each $\mathbf{s}_{t+1}^{(i)}$.
    Compute empirical soft values $\hat{V}^{\bar{\theta}}_{\mathrm{soft}}(\mathbf{s}_{t+1}^{(i)})$ in (10).
    Compute empirical gradient $\hat{\nabla}_\theta J_Q$ of (11).
    Update θ according to $\hat{\nabla}_\theta J_Q$ using ADAM.
    Update policy
    Sample $\{\xi^{(i,j)}\}_{j=0}^M \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ for each $\mathbf{s}_t^{(i)}$.
    Compute actions $\mathbf{a}_t^{(i,j)} = f^\phi(\xi^{(i,j)}, \mathbf{s}_t^{(i)})$.
    Compute Δf^ϕ using empirical estimate of (13).
    Compute empiricial estimate of (14): $\hat{\nabla}_\phi J_\pi$.
    Update ϕ according to $\hat{\nabla}_\phi J_\pi$ using ADAM.
  end for
  if epoch *mod* update_interval $= 0$ **then**
    Update target parameters: $\bar{\theta} \leftarrow \theta$, $\bar{\phi} \leftarrow \phi$.
  end if
end for

4. Related Work

Maximum entropy policies emerge as the solution when we cast optimal control as probabilistic inference. In the case of linear-quadratic systems, the mean of the maximum entropy policy is exactly the optimal deterministic policy (Todorov, 2008), which has been exploited to construct practical path planning methods based on iterative linearization and probabilistic inference techniques (Toussaint, 2009). In discrete state spaces, the maximum entropy policy can be obtained exactly. This has been explored in the context of linearly solvable MDPs (Todorov, 2007) and, in the case of inverse reinforcement learning, MaxEnt IRL (Ziebart et al., 2008). In continuous systems and continuous time, path integral control studies maximum entropy policies and maximum entropy planning (Kappen, 2005). In contrast to these prior methods, our work is focused on extending the maximum entropy policy search framework to high-dimensional continuous spaces and highly multimodal objectives, via expressive general-purpose energy functions represented by deep neural networks. A number of related methods have also used maximum entropy policy optimization as an intermediate step for optimizing policies under a standard expected reward objective (Pe



ters et al., 2010; Neumann, 2011; Rawlik et al., 2012; Fox et al., 2016). Among these, the work of Rawlik et al. (2012) resembles ours in that it also makes use of a temporal difference style update to a soft Q-function. However, unlike this prior work, we focus on general-purpose energy functions with approximate sampling, rather than analytically normalizable distributions. A recent work (Liu et al., 2017) also considers an entropy regularized objective, though the entropy is on policy parameters, not on sampled actions. Thus the resulting policy may not represent an arbitrarily complex multi-modal distribution with a single parameter. The form of our sampler resembles the stochastic networks proposed in recent work on hierarchical learning (Florensa et al., 2017). However this prior work uses a task-specific reward bonus system to encourage stochastic behavior, while our approach is derived from optimizing a general maximum entropy objective.

A closely related concept to maximum entropy policies is Boltzmann exploration, which uses the exponential of the standard Q-function as the probability of an action (Kaelbling et al., 1996). A number of prior works have also explored representing policies as energy-based models, with the Q-value obtained from an energy model such as a restricted Boltzmann machine (RBM) (Sallans & Hinton, 2004; Elfwing et al., 2010; Otsuka et al., 2010; Heess et al., 2012). Although these methods are closely related, they have not, to our knowledge, been extended to the case of deep network models, have not made extensive use of approximate inference techniques, and have not been demonstrated on the complex continuous tasks. More recently, O'Donoghue et al. (2016) drew a connection between Boltzmann exploration and entropy-regularized policy gradient, though in a theoretical framework that differs from maximum entropy policy search: unlike the full maximum entropy framework, the approach of O'Donoghue et al. (2016) only optimizes for maximizing entropy at the current time step, rather than planning for visiting future states where entropy will be further maximized. This prior method also does not demonstrate learning complex multi-modal policies in continuous action spaces.

Although we motivate our method as Q-learning, its structure resembles an actor-critic algorithm. It is particularly instructive to observe the connection between our approach and the deep deterministic policy gradient method (DDPG) (Lillicrap et al., 2015), which updates a Q-function critic according to (hard) Bellman updates, and then backpropagates the Q-value gradient into the actor, similarly to NFQCA (Hafner & Riedmiller, 2011). Our actor update differs only in the addition of the κ term. Indeed, without this term, our actor would estimate a maximum a posteriori (MAP) action, rather than capturing the entire EBM distribution. This suggests an intriguing connection between our method and DDPG: if we simply modify the DDPG critic updates to estimate soft Q-values, we recover the MAP variant of our method. Furthermore, this connection allows us to cast DDPG as simply an approximate Q-learning method, where the actor serves the role of an approximate maximizer. This helps explain the good performance of DDPG on off-policy data. We can also make a connection between our method and policy gradients. In Appendix B, we show that the policy gradient for a policy represented as an energy-based model closely corresponds to the update in soft Q-learning. Similar derivation is presented in a concurrent work (Schulman et al., 2017).

5. Experiments

Our experiments aim to answer the following questions: (1) Does our soft Q-learning method accurately capture a multi-modal policy distribution? (2) Can soft Q-learning with energy-based policies aid exploration for complex tasks that require tracking multiple modes? (3) Can a maximum entropy policy serve as a good initialization for fine-tuning on different tasks, when compared to pretraining with a standard deterministic objective? We compare our algorithm to DDPG (Lillicrap et al., 2015), which has been shown to achieve better sample efficiency on the continuous control problems that we consider than other recent techniques such as REINFORCE (Williams, 1992), TRPO (Schulman et al., 2015a), and A3C (Mnih et al., 2016). This comparison is particularly interesting since, as discussed in Section 4, DDPG closely corresponds to a deterministic maximum a posteriori variant of our method. The detailed experimental setup can be found in Appendix D. Videos of all experiments[2] and example source code[3] are available online.

5.1. Didactic Example: Multi-Goal Environment

In order to verify that amortized SVGD can correctly draw samples from energy-based policies of the form $\exp\left(Q^{\theta}_{\text{soft}}(s,a)\right)$, and that our complete algorithm can successful learn to represent multi-modal behavior, we designed a simple "multi-goal" environment, in which the agent is a 2D point mass trying to reach one of four symmetrically placed goals. The reward is defined as a mixture of Gaussians, with means placed at the goal positions. An optimal strategy is to go to an arbitrary goal, and the optimal maximum entropy policy should be able to choose each of the four goals at random. The final policy obtained with our method is illustrated in Figure 1. The Q-values indeed have complex shapes, being unimodal at $s = (-2, 0)$, convex at $s = (0, 0)$, and bimodal at $s = (2.5, 2.5)$. The stochastic policy samples actions closely following the energy landscape, hence learning diverse trajectories that lead to all four goals. In comparison, a policy trained with DDPG randomly commits to a single goal.

[2] https://sites.google.com/view/softqlearning/home
[3] https://github.com/haarnoja/softqlearning

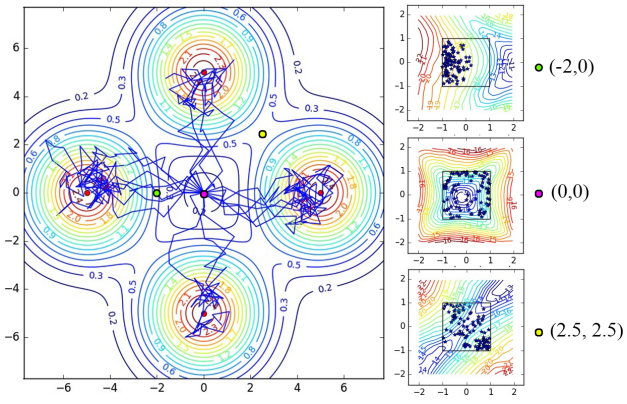
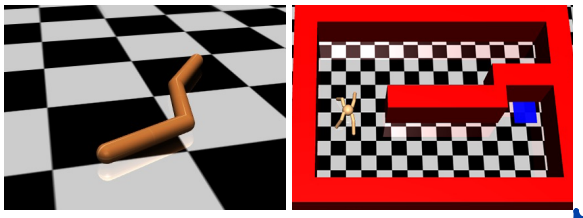


Figure 1. Illustration of the 2D multi-goal environment. Left: trajectories from a policy learned with our method (solid blue lines). The x and y axes correspond to 2D positions (states). The agent is initialized at the origin. The goals are depicted as red dots, and the level curves show the reward. Right: Q-values at three selected states, depicted by level curves (red: high values, blue: low values). The x and y axes correspond to 2D velocity (actions) bounded between -1 and 1. Actions sampled from the policy are shown as blue stars. Note that, in regions (e.g. $(2.5, 2.5)$) between the goals, the method chooses multimodal actions.

5.2. Learning Multi-Modal Policies for Exploration

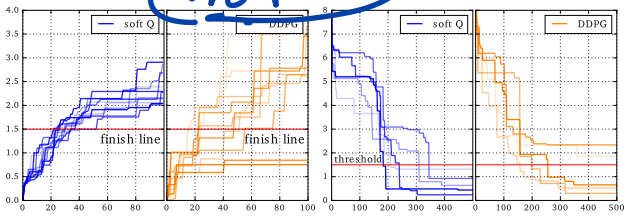
Though not all environments have a clear multi-modal reward landscape as in the "multi-goal" example, multi-modality is prevalent in a variety of tasks. For example, a chess player might try various strategies before settling on one that seems most effective, and an agent navigating a maze may need to try various paths before finding the exit. During the learning process, it is often best to keep trying multiple available options until the agent is confident that one of them is the best (similar to a bandit problem (Lai & Robbins, 1985)). However, deep RL algorithms for continuous control typically use unimodal action distributions, which are not well suited to capture such multi-modality. As a consequence, such algorithms may prematurely commit to one mode and converge to suboptimal behavior.

To evaluate how maximum entropy policies might aid exploration, we constructed simulated continuous control environments where tracking multiple modes is important for success. The first experiment uses a simulated swimming snake (see Figure 2), which receives a reward equal to its speed along the x-axis, either forward or backward. However, once the swimmer swims far enough forward, it crosses a "finish line" and receives a larger reward. Therefore, the best learning strategy is to explore in both directions until the bonus reward is discovered, and then commit to swimming forward. As illustrated in Figure 6 in Appendix D.3, our method is able to recover this strategy, keeping track of both modes until the finish line is discovered. All stochastic policies eventually commit to swimming forward. The deterministic DDPG method shown in the comparison commits to a mode prematurely, with only 80% of the policies converging on a forward motion, and 20% choosing the suboptimal backward mode.



(a) Swimming snake (b) Quadrupedal robot

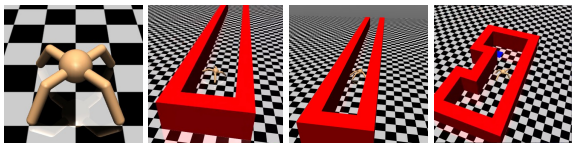
Figure 2. Simulated robots used in our experiments.



(a) Swimmer (higher is better) (b) Quadruped (lower is better)

Figure 3. Comparison of soft Q-learning and DDPG on the swimmer snake task and the quadrupedal robot maze task. (a) Shows the maximum traveled forward distance since the beginning of training for several runs of each algorithm; there is a large reward after crossing the finish line. (b) Shows our method was able to reach a low distance to the goal faster and more consistently. The different lines show the minimum distance to the goal since the beginning of training. For both domains, all runs of our method cross the threshold line, acquiring the more optimal strategy, while some runs of DDPG do not.

The second experiment studies a more complex task with a continuous range of equally good options prior to discovery of a sparse reward goal. In this task, a quadrupedal 3D robot (adapted from Schulman et al. (2015b)) needs to find a path through a maze to a target position (see Figure 2). The reward function is a Gaussian centered at the target. The agent may choose either the upper or lower passage, which appear identical at first, but the upper passage is blocked by a barrier. Similar to the swimmer experiment, the optimal strategy requires exploring both directions and choosing the better one. Figure 3(b) compares the performance of DDPG and our method. The curves show the minimum distance to the target achieved so far and the threshold equals the minimum possible distance if the robot chooses the upper passage. Therefore, successful exploration means reaching below the threshold. All policies trained with our method manage to succeed, while only 60% policies trained with DDPG converge to choosing the lower passage.

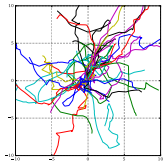


(a) (b) (c) (d)

Figure 4. Quadrupedal robot (a) was trained to walk in random directions in an empty pretraining environment (details in Figure 7, see Appendix D.3), and then finetuned on a variety of tasks, including a wide (b), narrow (c), and U-shaped hallway (d).

5.3. Accelerating Training on Complex Tasks with Pretrained Maximum Entropy Policies

A standard way to accelerate deep neural network training is task-specific initialization (Goodfellow et al., 2016), where a network trained for one task is used as initialization for another task. The first task might be something highly general, such as classifying a large image dataset, while the second task might be more specific, such as fine-grained classification with a small dataset. Pretraining has also been explored in the context of RL (Shelhamer et al., 2016). However, in RL, near-optimal policies are often near-deterministic, which makes them poor initializers for new tasks. In this section, we explore how our energy-based policies can be trained with fairly broad objectives to produce an initializer for more quickly learning more specific tasks.



We demonstrate this on a variant of the quadrupedal robot task. The pretraining phase involves learning to locomote in an arbitrary direction, with a reward that simply equals the speed of the center of mass. The resulting policy moves the agent quickly to an randomly chosen direction. An overhead plot of the center of mass traces is shown above to illustrate this. This pretraining is similar in some ways to recent work on modulated controllers (Heess et al., 2016) and hierarchical models (Florensa et al., 2017). However, in contrast to these prior works, we do not require any task-specific high-level goal generator or reward.

Figure 4 also shows a variety of test environments that we used to finetune the running policy for a specific task. In the hallway environments, the agent receives the same reward, but the walls block sideways motion, so the optimal solution requires learning to run in a particular direction. Narrow hallways require choosing a more specific direction, but also allow the agent to use the walls to funnel itself. The U-shaped maze requires the agent to learn a curved trajectory in order to arrive at the target, with the reward given by a Gaussian bump at the target location.

As illustrated in Figure 7 in Appendix D.3, the pretrained policy explores the space extensively and in all directions. This gives a good initialization for the policy, allowing it to learn the behaviors in the test environments more quickly than training a policy with DDPG from a random initialization, as shown in Figure 5. We also evaluated an alternative pretraining method based on deterministic policies learned with DDPG. However, deterministic pretraining chooses an arbitrary but consistent direction in the training environment, providing a poor initialization for finetuning to a specific task, as shown in the results plots.

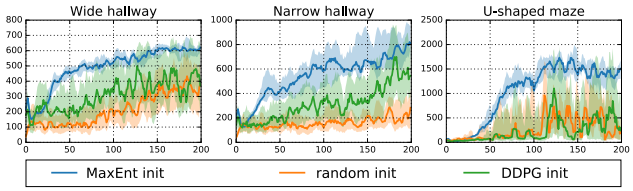


Figure 5. Performance in the downstream task with fine-tuning (MaxEnt) or training from scratch (DDPG). The x-axis shows the training iterations. The y-axis shows the average discounted return. Solid lines are average values over 10 random seeds. Shaded regions correspond to one standard deviation.

6. Discussion and Future Work

We presented a method for learning stochastic energy-based policies with approximate inference via Stein variational gradient descent (SVGD). Our approach can be viewed as a type of soft Q-learning method, with the additional contribution of using approximate inference to obtain complex multimodal policies. The sampling network trained as part of SVGD can also be viewed as tking the role of an actor in an actor-critic algorithm. Our experimental results show that our method can effectively capture complex multi-modal behavior on problems ranging from toy point mass tasks to complex torque control of simulated walking and swimming robots. The applications of training such stochastic policies include improved exploration in the case of multimodal objectives and compositionality via pretraining general-purpose stochastic policies that can then be efficiently finetuned into task-specific behaviors.

While our work explores some potential applications of energy-based policies with approximate inference, an exciting avenue for future work would be to further study their capability to represent complex behavioral repertoires and their potential for composability. In the context of linearly solvable MDPs, several prior works have shown that policies trained for different tasks can be composed to create new optimal policies (Da Silva et al., 2009; Todorov, 2009). While these prior works have only explored simple, tractable representations, our method could be used to extend these results to complex and highly multi-modal deep neural network models, making them suitable for composable control of complex high-dimensional systems, such as humanoid robots. This composability could be used in future work to create a huge variety of near-optimal skills from a collection of energy-based policy building blocks.

7. Acknowledgements

We thank Qiang Liu for insightful discussion of SVGD, and thank Vitchyr Pong and Shane Gu for help with implementing DDPG. Haoran Tang and Tuomas Haarnoja are supported by Berkeley Deep Drive.

References

Da Silva, M., Durand, F., and Popović, J. Linear Bellman combination for control of character animation. *ACM Trans. on Graphs*, 28(3):82, 2009.

Daniel, C., Neumann, G., and Peters, J. Hierarchical relative entropy policy search. In *AISTATS*, pp. 273–281, 2012.

Elfwing, S., Otsuka, M., Uchibe, E., and Doya, K. Free-energy based reinforcement learning for vision-based navigation with high-dimensional sensory inputs. In *Int. Conf. on Neural Information Processing*, pp. 215–222. Springer, 2010.

Florensa, C., Duan, Y., and P., Abbeel. Stochastic neural networks for hierarchical reinforcement learning. In *Int. Conf. on Learning Representations*, 2017.

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

Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. Deep learning. chapter 8.7.4. MIT Press, 2016. http://www.deeplearningbook.org.

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

Gu, S., Lillicrap, T., Sutskever, I., and Levine, S. Continuous deep Q-learning with model-based acceleration. In *Int. Conf. on Machine Learning*, pp. 2829–2838, 2016b.

Hafner, R. and Riedmiller, M. Reinforcement learning in feedback control. *Machine Learning*, 84(1-2):137–169, 2011.

Heess, N., Silver, D., and Teh, Y. W. Actor-critic reinforcement learning with energy-based policies. In *Workshop on Reinforcement Learning*, pp. 43. Citeseer, 2012.

Heess, N., Wayne, G., Tassa, Y., Lillicrap, T., Riedmiller, M., and Silver, D. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

Kakade, S. A natural policy gradient. *Advances in Neural Information Processing Systems*, 2:1531–1538, 2002.

Kappen, H. J. Path integrals and symmetry breaking for optimal control theory. *Journal of Statistical Mechanics: Theory And Experiment*, 2005(11):P11011, 2005.

Kim, T. and Bengio, Y. Deep directed generative models with energy-based probability estimation. *arXiv preprint arXiv:1606.03439*, 2016.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. 2015.

Lai, T. L. and Robbins, H. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.

Levine, S. and Abbeel, P. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems*, pp. 1071–1079, 2014.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

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

Liu, Q. and Wang, D. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances In Neural Information Processing Systems*, pp. 2370–2378, 2016.

Liu, Y., Ramachandran, P., Liu, Q., and Peng, J. Stein variational policy gradient. *arXiv preprint arXiv:1704.02399*, 2017.

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

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

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

Neumann, G. Variational inference for policy search in changing situations. In *Int. Conf. on Machine Learning*, pp. 817–824, 2011.

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

Otsuka, M., Yoshimoto, J., and Doya, K. Free-energy-based reinforcement learning in a partially observable environment. In *ESANN*, 2010.

Peters, J., Mülling, K., and Altun, Y. Relative entropy policy search. In *AAAI Conf. on Artificial Intelligence*, pp. 1607–1612, 2010.

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

Sallans, B. and Hinton, G. E. Reinforcement learning with factored states and actions. *Journal of Machine Learning Research*, 5(Aug):1063–1088, 2004.

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

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

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

Shelhamer, E., Mahmoudieh, P., Argus, M., and Darrell, T. Loss is its own reward: Self-supervision for reinforcement learning. *arXiv preprint arXiv:1612.07307*, 2016.

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

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

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

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

Todorov, E. Linearly-solvable Markov decision problems. In *Advances in Neural Information Processing Systems*, pp. 1369–1376. MIT Press, 2007.

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

Todorov, E. Compositionality of optimal control laws. In *Advances in Neural Information Processing Systems*, pp. 1856–1864, 2009.

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

Uhlenbeck, G. E. and Ornstein, L. S. On the theory of the brownian motion. *Physical review*, 36(5):823, 1930.

Wang, D. and Liu, Q. Learning to draw samples: With application to amortized mle for generative adversarial learning. *arXiv preprint arXiv:1611.01722*, 2016.

Williams, Ronald J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Zhao, J., Mathieu, M., and LeCun, Y. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.

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

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

Appendices

A. Policy Improvement Proofs

In this appendix, we present proofs for the theorems that allow us to show that soft Q-learning leads to policy improvement with respect to the maximum entropy objective. First, we define a slightly more nuanced version of the maximum entropy objective that allows us to incorporate a discount factor. This definition is complicated by the fact that, when using a discount factor for policy gradient methods, we typically do not discount the state distribution, only the rewards. In that sense, discounted policy gradients typically do not optimize the true discounted objective. Instead, they optimize average reward, with the discount serving to reduce variance, as discussed by Thomas (2014). However, for the purposes of the derivation, we can define the objective that *is* optimized under a discount factor as

$$\pi^*_{\text{MaxEnt}} = \arg\max_\pi \sum_t \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi}\left[\sum_{l=t}^{\infty}\gamma^{l-t}\mathbb{E}_{(\mathbf{s}_l,\mathbf{a}_l)}\left[r(\mathbf{s}_t,\mathbf{a}_t) + \alpha\mathcal{H}(\pi(\,\cdot\,|\mathbf{s}_t))|\mathbf{s}_t,\mathbf{a}_t\right]\right].$$

This objective corresponds to maximizing the discounted expected reward and entropy for future states originating from every state-action tuple $(\mathbf{s}_t, \mathbf{a}_t)$ weighted by its probability ρ_π under the current policy. Note that this objective still takes into account the entropy of the policy at future states, in contrast to greedy objectives such as Boltzmann exploration or the approach proposed by O'Donoghue et al. (2016).

We can now derive policy improvement results for soft Q-learning. We start with the definition of the soft Q-value Q^π_{soft} for any policy π as the expectation under π of the discounted sum of rewards and entropy :

$$Q^\pi_{\text{soft}}(\mathbf{s},\mathbf{a}) \triangleq r_0 + \mathbb{E}_{\tau\sim\pi,\mathbf{s}_0=\mathbf{s},\mathbf{a}_0=\mathbf{a}}\left[\sum_{t=1}^{\infty}\gamma^t(r_t + \mathcal{H}(\pi(\,\cdot\,|\mathbf{s}_t)))\right]. \tag{15}$$

Here, $\tau = (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \ldots)$ denotes the trajectory originating at $(\mathbf{s}, \mathbf{a})$. Notice that for convenience, we set the entropy parameter α to 1. The theory can be easily adapted by dividing rewards by α.

The discounted maximum entropy policy objective can now be defined as

$$J(\pi) \triangleq \sum_t \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi}\left[Q^\pi_{\text{soft}}(\mathbf{s}_t,\mathbf{a}_t) + \alpha\mathcal{H}(\pi(\,\cdot\,|\mathbf{s}_t))\right]. \tag{16}$$

A.1. The Maximum Entropy Policy

If the objective function is the expected discounted sum of rewards, the policy improvement theorem (Sutton & Barto, 1998) describes how policies can be improved monotonically. There is a similar theorem we can derive for the maximum entropy objective:

Theorem 4. *(Policy improvement theorem) Given a policy π, define a new policy $\tilde{\pi}$ as*

$$\tilde{\pi}(\,\cdot\,|\mathbf{s}) \propto \exp\left(Q^\pi_{soft}(\mathbf{s},\,\cdot\,)\right), \quad \forall \mathbf{s}. \tag{17}$$

Assume that throughout our computation, Q is bounded and $\int \exp(Q(\mathbf{s},\mathbf{a}))\,d\mathbf{a}$ is bounded for any $\mathbf{s}$ (for both π and $\tilde{\pi}$). Then $Q^{\tilde{\pi}}_{soft}(\mathbf{s},\mathbf{a}) \geq Q^\pi_{soft}(\mathbf{s},\mathbf{a})\ \forall \mathbf{s},\mathbf{a}$.

The proof relies on the following observation: if one greedily maximize the sum of entropy and value with one-step look-ahead, then one obtains $\tilde{\pi}$ from π:

$$\mathcal{H}(\pi(\,\cdot\,|\mathbf{s})) + \mathbb{E}_{\mathbf{a}\sim\pi}\left[Q^\pi_{\text{soft}}(\mathbf{s},\mathbf{a})\right] \leq \mathcal{H}(\tilde{\pi}(\,\cdot\,|\mathbf{s})) + \mathbb{E}_{\mathbf{a}\sim\tilde{\pi}}\left[Q^\pi_{\text{soft}}(\mathbf{s},\mathbf{a})\right]. \tag{18}$$

The proof is straight-forward by noticing that

$$\mathcal{H}(\pi(\,\cdot\,|\mathbf{s})) + \mathbb{E}_{\mathbf{a}\sim\pi}\left[Q^\pi_{\text{soft}}(\mathbf{s},\mathbf{a})\right] = -\mathrm{D}_{\mathrm{KL}}\left(\pi(\,\cdot\,|\mathbf{s})\parallel\tilde{\pi}(\,\cdot\,|\mathbf{s})\right) + \log\int\exp\left(Q^\pi_{\text{soft}}(\mathbf{s},\mathbf{a})\right)\,d\mathbf{a} \tag{19}$$

Then we can show that

$$
\begin{aligned}
Q^{\pi}_{\text{soft}}(\mathbf{s},\mathbf{a}) &= \mathbb{E}_{\mathbf{s}_1}\left[r_0 + \gamma(\mathcal{H}(\pi(\,\cdot\,|\mathbf{s}_1)) + \mathbb{E}_{\mathbf{a}_1\sim\pi}\left[Q^{\pi}_{\text{soft}}(\mathbf{s}_1,\mathbf{a}_1)\right])\right] \\
&\leq \mathbb{E}_{\mathbf{s}_1}\left[r_0 + \gamma(\mathcal{H}(\tilde{\pi}(\,\cdot\,|\mathbf{s}_1)) + \mathbb{E}_{\mathbf{a}_1\sim\tilde{\pi}}\left[Q^{\pi}_{\text{soft}}(\mathbf{s}_1,\mathbf{a}_1)\right])\right] \\
&= \mathbb{E}_{\mathbf{s}_1}\left[r_0 + \gamma(\mathcal{H}(\tilde{\pi}(\,\cdot\,|\mathbf{s}_1)) + r_1)\right] + \gamma^2\,\mathbb{E}_{\mathbf{s}_2}\left[\mathcal{H}(\pi(\,\cdot\,|\mathbf{s}_2)) + \mathbb{E}_{\mathbf{a}_2\sim\pi}\left[Q^{\pi}_{\text{soft}}(\mathbf{s}_2,\mathbf{a}_2)\right]\right] \\
&\leq \mathbb{E}_{\mathbf{s}_1}\left[r_0 + \gamma(\mathcal{H}(\tilde{\pi}(\,\cdot\,|\mathbf{s}_1)) + r_1\right] + \gamma^2\,\mathbb{E}_{\mathbf{s}_2}\left[\mathcal{H}(\tilde{\pi}(\,\cdot\,|\mathbf{s}_2)) + \mathbb{E}_{\mathbf{a}_2\sim\tilde{\pi}}\left[Q^{\pi}_{\text{soft}}(\mathbf{s}_2,\mathbf{a}_2)\right]\right] \\
&= \mathbb{E}_{\mathbf{s}_1\,\mathbf{a}_2\sim\tilde{\pi},\mathbf{s}_2}\left[r_0 + \gamma(\mathcal{H}(\tilde{\pi}(\,\cdot\,|\mathbf{s}_1)) + r_1) + \gamma^2(\mathcal{H}(\tilde{\pi}(\,\cdot\,|\mathbf{s}_2)) + r_2)\right] + \gamma^3\,\mathbb{E}_{\mathbf{s}_3}\left[\mathcal{H}(\tilde{\pi}(\,\cdot\,|\mathbf{s}_3)) + \mathbb{E}_{\mathbf{a}_3\sim\tilde{\pi}}\left[Q^{\pi}_{\text{soft}}(\mathbf{s}_3,\mathbf{a}_3)\right]\right] \\
&\;\;\vdots \\
&\leq \mathbb{E}_{\tau\sim\tilde{\pi}}\left[r_0 + \sum_{t=1}^{\infty}\gamma^t(\mathcal{H}(\tilde{\pi}(\,\cdot\,|\mathbf{s}_t)) + r_t)\right] \\
&= Q^{\tilde{\pi}}_{\text{soft}}(\mathbf{s},\mathbf{a}).
\end{aligned}
\tag{20}
$$

With Theorem 4, we start from an arbitrary policy π_0 and define the *policy iteration* as

$$
\pi_{i+1}(\,\cdot\,|\mathbf{s}) \propto \exp\left(Q^{\pi_i}_{\text{soft}}(\mathbf{s},\,\cdot\,)\right). \tag{21}
$$

Then $Q^{\pi_i}_{\text{soft}}(\mathbf{s},\mathbf{a})$ improves monotonically. Under certain regularity conditions, π_i converges to π_∞. Obviously, we have $\pi_\infty(\mathbf{a}|\mathbf{s}) \propto_{\mathbf{a}} \exp\left(Q^{\pi_\infty}(\mathbf{s},\mathbf{a})\right)$. Since any non-optimal policy can be improved this way, the optimal policy must satisfy this energy-based form. Therefore we have proven Theorem 1.

A.2. Soft Bellman Equation and Soft Value Iteration

Recall the definition of the soft value function:

$$
V^{\pi}_{\text{soft}}(\mathbf{s}) \triangleq \log\int \exp\left(Q^{\pi}_{\text{soft}}(\mathbf{s},\mathbf{a})\right)\,d\mathbf{a}. \tag{22}
$$

Suppose $\pi(\mathbf{a}|\mathbf{s}) = \exp\left(Q^{\pi}_{\text{soft}}(\mathbf{s},\mathbf{a}) - V^{\pi}_{\text{soft}}(\mathbf{s})\right)$. Then we can show that

$$
\begin{aligned}
Q^{\pi}_{\text{soft}}(\mathbf{s},\mathbf{a}) &= r(\mathbf{s},\mathbf{a}) + \gamma\,\mathbb{E}_{\mathbf{s}'\sim p_{\mathbf{s}}}\left[\mathcal{H}(\pi(\,\cdot\,|\mathbf{s}')) + \mathbb{E}_{\mathbf{a}'\sim\pi(\,\cdot\,|\mathbf{s}')}\left[Q^{\pi}_{\text{soft}}(\mathbf{s}',\mathbf{a}')\right]\right] \\
&= r(\mathbf{s},\mathbf{a}) + \gamma\,\mathbb{E}_{\mathbf{s}'\sim p_{\mathbf{s}}}\left[V^{\pi}_{\text{soft}}(\mathbf{s}')\right].
\end{aligned}
\tag{23}
$$

This completes the proof of Theorem 2.

Finally, we show that the soft value iteration operator $\mathcal{T}$, defined as

$$
\mathcal{T}Q(\mathbf{s},\mathbf{a}) \triangleq r(\mathbf{s},\mathbf{a}) + \gamma\,\mathbb{E}_{\mathbf{s}'\sim p_{\mathbf{s}}}\left[\log\int \exp Q(\mathbf{s}',\mathbf{a}')\,d\mathbf{a}'\right], \tag{24}
$$

is a contraction. Then Theorem 3 follows immediately.

The following proof has also been presented by Fox et al. (2016). Define a norm on Q-values as $\|Q_1 - Q_2\| \triangleq \max_{\mathbf{s},\mathbf{a}}|Q_1(\mathbf{s},\mathbf{a}) - Q_2(\mathbf{s},\mathbf{a})|$. Suppose $\varepsilon = \|Q_1 - Q_2\|$. Then

$$
\begin{aligned}
\log\int \exp(Q_1(\mathbf{s}',\mathbf{a}'))\,d\mathbf{a}' &\leq \log\int \exp(Q_2(\mathbf{s}',\mathbf{a}') + \varepsilon)\,d\mathbf{a}' \\
&= \log\left(\exp(\varepsilon)\int \exp Q_2(\mathbf{s}',\mathbf{a}')\,d\mathbf{a}'\right) \\
&= \varepsilon + \log\int \exp Q_2(\mathbf{a}',\mathbf{a}')\,d\mathbf{a}'.
\end{aligned}
\tag{25}
$$

Similarly, $\log\int \exp Q_1(\mathbf{s}',\mathbf{a}')\,d\mathbf{a}' \geq -\varepsilon + \log\int \exp Q_2(\mathbf{s}',\mathbf{a}')\,d\mathbf{a}'$. Therefore $\|\mathcal{T}Q_1 - \mathcal{T}Q_2\| \leq \gamma\varepsilon = \gamma\|Q_1 - Q_2\|$. So $\mathcal{T}$ is a contraction. As a consequence, only one Q-value satisfies the soft Bellman equation, and thus the optimal policy presented in Theorem 1 is unique.

B. Connection between Policy Gradient and Q-Learning

We show that entropy-regularized policy gradient can be viewed as performing soft Q-learning on the maximum-entropy objective. First, suppose that we parametrize a stochastic policy as

$$\pi^\phi(\mathbf{a}_t|\mathbf{s}_t) \triangleq \exp\left(\mathcal{E}^\phi(\mathbf{s}_t, \mathbf{a}_t) - \bar{\mathcal{E}}^\phi(\mathbf{s}_t)\right), \tag{26}$$

where $\mathcal{E}^\phi(\mathbf{s}_t, \mathbf{a}_t)$ is an energy function with parameters ϕ, and $\bar{\mathcal{E}}^\phi(\mathbf{s}_t) = \log \int_{\mathcal{A}} \exp \mathcal{E}^\phi(\mathbf{s}_t, \mathbf{a}_t) d\mathbf{a}_t$ is the corresponding partition function. This is the most general class of policies, as we can trivially transform any given distribution p into exponential form by defining the energy as $\log p$. We can write an entropy-regularized policy gradient as follows:

$$\nabla_\phi J_{\mathrm{PG}}(\phi) = \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_{\pi^\phi}}\left[\nabla_\phi \log \pi^\phi(\mathbf{a}_t|\mathbf{s}_t)\left(\hat{Q}_{\pi^\phi}(\mathbf{s}_t, \mathbf{a}_t) + b^\phi(\mathbf{s}_t)\right)\right] + \nabla_\phi \mathbb{E}_{\mathbf{s}_t\sim\rho_{\pi^\phi}}\left[\mathcal{H}(\pi^\phi(\,\cdot\,|\mathbf{s}_t))\right], \tag{27}$$

where $\rho_{\pi^\phi}(\mathbf{s}_t, \mathbf{a}_t)$ is the distribution induced by the policy, $\hat{Q}_{\pi^\phi}(\mathbf{s}_t, \mathbf{a}_t)$ is an empirical estimate of the Q-value of the policy, and $b^\phi(\mathbf{s}_t)$ is a state-dependent baseline that we get to choose. The gradient of the entropy term is given by

$$\begin{aligned}
\nabla_\phi \mathcal{H}(\pi^\phi) &= -\nabla_\phi \mathbb{E}_{\mathbf{s}_t\sim\rho_{\pi^\phi}}\left[\mathbb{E}_{\mathbf{a}_t\sim\pi^\phi(\mathbf{a}_t|\mathbf{s}_t)}\left[\log \pi^\phi(\mathbf{a}_t|\mathbf{s}_t)\right]\right] \\
&= -\mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_{\pi^\phi}}\left[\nabla_\phi \log \pi^\phi(\mathbf{a}_t|\mathbf{s}_t) \log \pi^\phi(\mathbf{a}_t|\mathbf{s}_t) + \nabla_\phi \log \pi^\phi(\mathbf{a}_t|\mathbf{s}_t)\right] \\
&= -\mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_{\pi^\phi}}\left[\nabla_\phi \log \pi^\phi(\mathbf{a}_t|\mathbf{s}_t)\left(1 + \log \pi^\phi(\mathbf{a}_t|\mathbf{s}_t)\right)\right],
\end{aligned} \tag{28}$$

and after substituting this back into (27), noting (26), and choosing $b^\phi(\mathbf{s}_t) = \bar{\mathcal{E}}^\phi(\mathbf{s}_t) + 1$, we arrive at a simple form for the policy gradient:

$$= \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_{\pi^\phi}}\left[\left(\nabla_\phi \mathcal{E}^\phi(\mathbf{s}_t, \mathbf{a}_t) - \nabla_\phi \bar{\mathcal{E}}^\phi(\mathbf{s}_t)\right)\left(\hat{Q}_{\pi^\phi}(\mathbf{s}_t, \mathbf{a}_t) - \mathcal{E}^\phi(\mathbf{s}_t, \mathbf{a}_t)\right)\right]. \tag{29}$$

To show that (29) indeed correponds to soft Q-learning update, we consider the Bellman error

$$J_Q(\theta) = \mathbb{E}_{\mathbf{s}_t\sim q_{\mathbf{s}_t}, \mathbf{a}_t\sim q_{\mathbf{a}_t}}\left[\frac{1}{2}\left(\hat{Q}^\theta_{\mathrm{soft}}(\mathbf{s}_t, \mathbf{a}_t) - Q^\theta_{\mathrm{soft}}(\mathbf{s}_t, \mathbf{a}_t)\right)^2\right], \tag{30}$$

where $\hat{Q}^\theta_{\mathrm{soft}}$ is an empirical estimate of the soft Q-function. There are several valid alternatives for this estimate, but in order to show a connection to policy gradient, we choose a specific form

$$\hat{Q}^\theta_{\mathrm{soft}}(\mathbf{s}_t, \mathbf{a}_t) = \hat{A}^{\bar{\theta}}_{\mathrm{soft}}(\mathbf{s}_t, \mathbf{a}_t) + V^\theta_{\mathrm{soft}}(\mathbf{s}_t), \tag{31}$$

where $\hat{A}^{\bar{\theta}}_{\mathrm{soft}}$ is an empirical soft advantage function that is assumed not to contribute the gradient computation. With this choice, the gradient of the Bellman error becomes

$$\begin{aligned}
\nabla_\theta J_Q(\theta) &= \mathbb{E}_{\mathbf{s}_t\sim q_{\mathbf{s}_t}, \mathbf{a}_t\sim q_{\mathbf{a}_t}}\left[\left(\nabla_\theta Q^\theta_{\mathrm{soft}}(\mathbf{s}_t, \mathbf{a}_t) - \nabla_\theta V^\theta_{\mathrm{soft}}(\mathbf{s}_t)\right)\left(\hat{A}^{\bar{\theta}}_{\mathrm{soft}}(\mathbf{s}_t, \mathbf{a}_t) + V^\theta_{\mathrm{soft}}(\mathbf{s}_t, \mathbf{a}_t) - Q^\theta_{\mathrm{soft}}(\mathbf{s}_t, \mathbf{a}_t)\right)\right] \\
&= \mathbb{E}_{\mathbf{s}_t\sim q_{\mathbf{s}_t}, \mathbf{a}_t\sim q_{\mathbf{a}_t}}\left[\left(\nabla_\theta Q^\theta_{\mathrm{soft}}(\mathbf{s}_t, \mathbf{a}_t) - \nabla_\theta V^\theta_{\mathrm{soft}}(\mathbf{s}_t)\right)\left(\hat{Q}^\theta_{\mathrm{soft}}(\mathbf{s}_t, \mathbf{a}_t) - Q^\theta_{\mathrm{soft}}(\mathbf{s}_t, \mathbf{a}_t)\right)\right].
\end{aligned} \tag{32}$$

Now, if we choose $\mathcal{E}^\phi(\mathbf{s}_t, \mathbf{a}_t) \triangleq Q^\theta_{\mathrm{soft}}(\mathbf{s}_t, \mathbf{a}_t)$ and $q_{\mathbf{s}_t}(\mathbf{s}_t) q_{\mathbf{a}_t}(\mathbf{a}_t) \triangleq \rho_{\pi^\phi}(\mathbf{s}_t, \mathbf{a}_t)$, we recover the policy gradient in (29). Note that the choice of using an empirical estimate of the soft advantage rather than soft Q-value makes the target independent of the soft value, and at convergence, Q^θ_{soft} approximates the soft Q-value up to an additive constant. The resulting policy is still correct, since the Boltzmann distribution in (26) is independent of constant shift in the energy function.

C. Implementation

C.1. Computing the Policy Update

Here we explain in full detail how the policy update direction $\hat{\nabla}_\phi J_\pi$ in Algorithm 1 is computed. We reuse the indices i, j in this section with a different meaning than in the body of the paper for the sake of providing a cleaner presentation.

Expectations appear in amortized SVGD in two places. First, SVGD approximates the optimal descent direction $\phi(\,\cdot\,)$ in Equation (13) with an empirical average over the samples $\mathbf{a}^{(i)}_t = f^\phi(\xi^{(i)})$. Similarly, SVGD approximates the expectation

in Equation (14) with samples $\tilde{\mathbf{a}}_t^{(j)} = f^\phi(\tilde{\xi}^{(j)})$, which can be the same or different from $\mathbf{a}_t^{(i)}$. Substituting (13) into (14) and taking the gradient gives the empirical estimate

$$\hat{\nabla}_\phi J_\pi(\phi; \mathbf{s}_t) = \frac{1}{KM} \sum_{j=1}^{K} \sum_{i=1}^{M} \left(\kappa(\mathbf{a}_t^{(i)}, \tilde{\mathbf{a}}_t^{(j)}) \nabla_{\mathbf{a}'} Q_{\text{soft}}(\mathbf{s}_t, \mathbf{a}') \big|_{\mathbf{a}'=\mathbf{a}_t^{(i)}} + \nabla_{\mathbf{a}'} \kappa(\mathbf{a}', \tilde{\mathbf{a}}_t^{(j)}) \big|_{\mathbf{a}'=\mathbf{a}_t^{(i)}} \right) \nabla_\phi f^\phi(\tilde{\xi}^{(j)}; \mathbf{s}_t).$$

Finally, the update direction $\hat{\nabla}_\phi J_\pi$ is the average of $\hat{\nabla}_\phi J_\pi(\phi; \mathbf{s}_t)$, where $\mathbf{s}_t$ is drawn from a mini-batch.

C.2. Computing the Density of Sampled Actions

Equation (10) states that the soft value can be computed by sampling from a distribution $q_{\mathbf{a}'}$ and that $q_{\mathbf{a}'}(\,\cdot\,) \propto \exp\left(\frac{1}{\alpha} Q^\phi_{\text{soft}}(\mathbf{s}, \,\cdot\,)\right)$ is optimal. A direct solution is to obtain actions from the sampling network: $\mathbf{a}' = f^\phi(\xi'; \mathbf{s})$. If the samples ξ' and actions $\mathbf{a}'$ have the same dimension, and if the jacobian matrix $\frac{\partial \mathbf{a}'}{\partial \xi'}$ is non-singular, then the probability density is

$$q_{\mathbf{a}'}(\mathbf{a}') = p_\xi(\xi') \frac{1}{\left| \det\left(\frac{\partial \mathbf{a}'}{\partial \xi'} \right) \right|}. \tag{33}$$

In practice, the Jacobian is usually singular at the beginning of training, when the sampler f^ϕ is not fully trained. A simple solution is to begin with uniform action sampling and then switch to f^ϕ later, which is reasonable, since an untrained sampler is unlikely to produce better samples for estimating the partition function anyway.

D. Experiments

D.1. Hyperparameters

Throughout all experiments, we use the following parameters for both DDPG and soft Q-learning. The Q-values are updated using ADAM with learning rate 0.001. The DDPG policy and soft Q-learning sampling network use ADAM with a learning rate of 0.0001. The algorithm uses a replay pool of size one million. Training does not start until the replay pool has at least 10,000 samples. Every mini-batch has size 64. Each training iteration consists of 10000 time steps, and both the Q-values and policy / sampling network are trained at every time step. All experiments are run for 500 epochs, except that the multi-goal task uses 100 epochs and the fine-tuning tasks are trained for 200 epochs. Both the Q-value and policy / sampling network are neural networks comprised of two hidden layers, with 200 hidden units at each layer and ReLU nonlinearity. Both DDPG and soft Q-learning use additional OU Noise (Uhlenbeck & Ornstein, 1930; Lillicrap et al., 2015) to improve exploration. The parameters are $\theta = 0.15$ and $\sigma = 0.3$. In addition, we found that updating the target parameters too frequently can destabilize training. Therefore we freeze target parameters for every 1000 time steps (except for the swimming snake experiment, which freezes for 5000 epochs), and then copy the current network parameters to the target networks directly ($\tau = 1$).

Soft Q-learning uses $K = M = 32$ action samples (see Appendix C.1) to compute the policy update, except that the multi-goal experiment uses $K = M = 100$. The number of additional action samples to compute the soft value is $K_V = 50$. The kernel κ is a radial basis function, written as $\kappa(\mathbf{a}, \mathbf{a}') = \exp(-\frac{1}{h}\|\mathbf{a} - \mathbf{a}'\|_2^2)$, where $h = \frac{d}{2\log(M+1)}$, with d equal to the median of pairwise distance of sampled actions $\mathbf{a}_t^{(i)}$. Note that the step size h changes dynamically depending on the state $\mathbf{s}$, as suggested in (Liu & Wang, 2016).

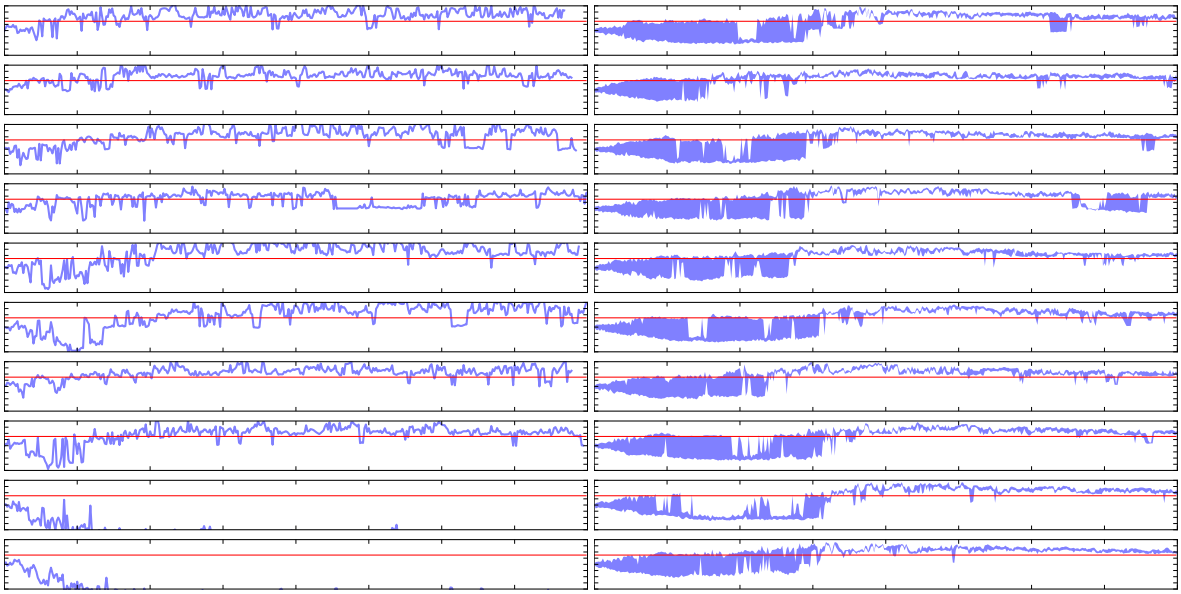
The entropy coefficient α is 10 for multi-goal environment, and 0.1 for the swimming snake, maze, hallway (pretraining) and U-shaped maze (pretraining) experiments.

All fine-tuning tasks anneal the entropy coefficient α quickly in order to improve performance, since the goal during fine-tuning is to recover a near-deterministic policy on the fine-tuning task. In particular, α is annealed log-linearly to 0.001 within 20 epochs of fine-tuning. Moreover, the samples ξ are fixed to a set $\{\xi_i\}_{i=1}^{K_\xi}$ and K_ξ is reduced linearly to 1 within 20 epochs.

D.2. Task description

All tasks have a horizon of $T = 500$, except the multi-goal task, which uses $T = 20$. We add an additional termination condition to the quadrupedal 3D robot to discourage it from flipping over.

D.3. Additional Results



(a) DDPG (b) Soft Q-learning

Figure 6. Forward swimming distance achieved by each policy. Each row is a policy with a unique random seed. x: training iteration, y: distance (positive: forward, negative: backward). Red line: the "finish line." The blue shaded region is bounded by the maximum and minimum distance (which are equal for DDPG). The plot shows that our method is able to explore equally well in both directions before it commits to the better one.

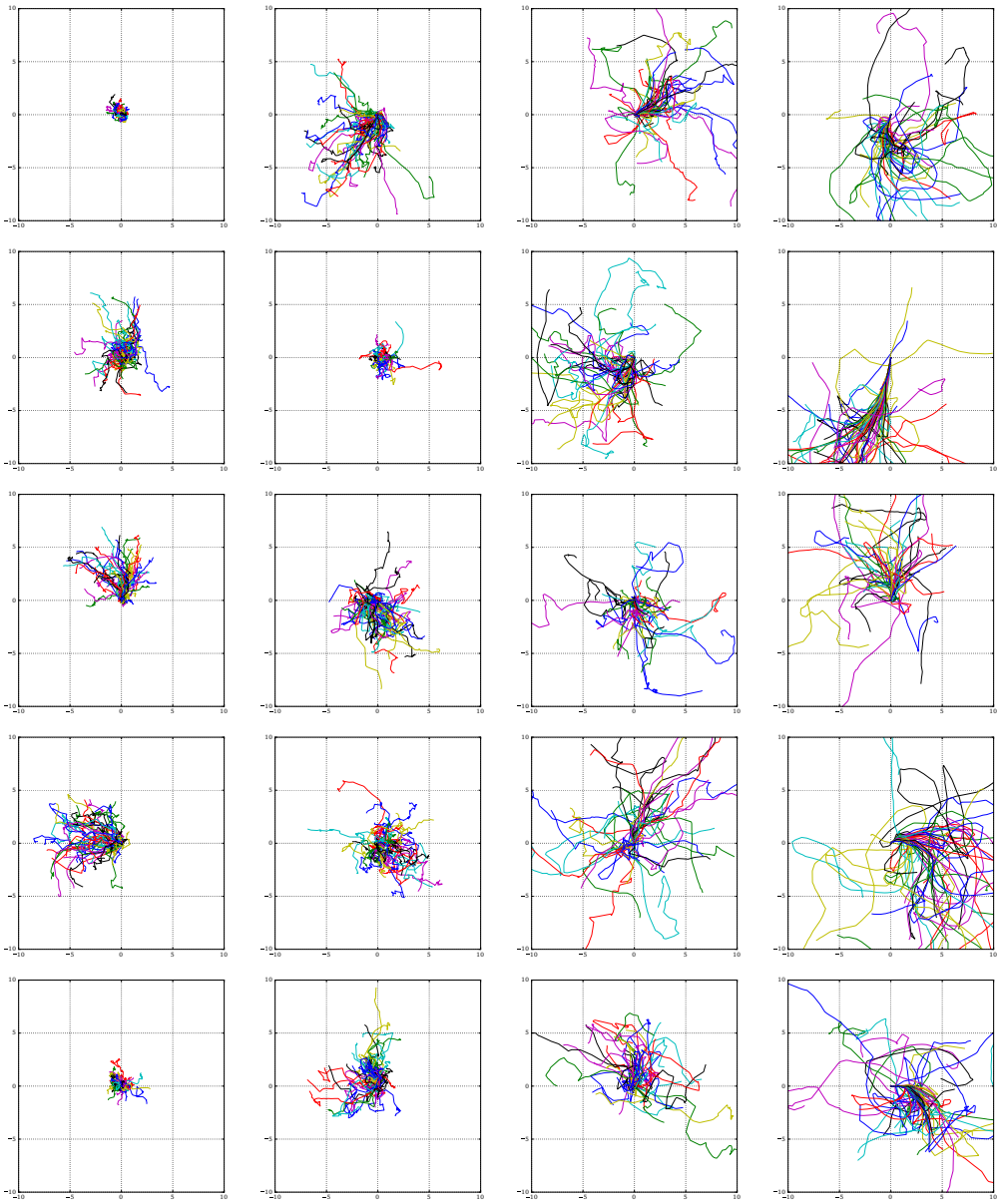


Figure 7. The plot shows trajectories of the quadrupedal robot during maximum entropy pretraining. The robot has diverse behavior and explores multiple directions. The four columns correspond to entropy coefficients $\alpha = 10, 1, 0.1, 0.01$ respectively. Different rows correspond to policies trained with different random seeds. The x and y axes show the x and y coordinates of the center-of-mass. As α decreases, the training process focuses more on high rewards, therefore exploring the training ground more extensively. However, low α also tends to produce less diverse behavior. Therefore the trajectories are more concentrated in the fourth column.