Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (2008)

Maximum Entropy Inverse Reinforcement Learning

Brian D. Ziebart, Andrew Maas, J.Andrew Bagnell, and **Anind K. Dey**
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
bziebart@cs.cmu.edu, amaas@andrew.cmu.edu, dbagnell@ri.cmu.edu, anind@cs.cmu.edu

Abstract

Recent research has shown the benefit of framing problems of imitation learning as solutions to Markov Decision Problems. This approach reduces learning to the problem of recovering a utility function that makes the behavior induced by a near-optimal policy closely mimic demonstrated behavior. In this work, we develop a probabilistic approach based on the principle of maximum entropy. Our approach provides a well-defined, globally normalized distribution over decision sequences, while providing the same performance guarantees as existing methods.

We develop our technique in the context of modeling real-world navigation and driving behaviors where collected data is inherently noisy and imperfect. Our probabilistic approach enables modeling of route preferences as well as a powerful new approach to inferring destinations and routes based on partial trajectories.

Introduction

In problems of *imitation learning* the goal is to learn to predict the behavior and decisions an agent would choose– e.g., the motions a person would take to grasp an object or the route a driver would take to get from home to work. Capturing purposeful, sequential decision-making behavior can be quite difficult for general-purpose statistical machine learning algorithms; in such problems, algorithms must often reason about consequences of actions far into the future.

A powerful recent idea for approaching problems of imitation learning is to structure the space of learned policies to be solutions of search, planning, or, more generally, Markov Decision Problems (MDP). The key notion, intuitively, is that agents act to optimize an unknown reward function (assumed to be linear in the features) and that we must find reward weights that make their demonstrated behavior appear (near)-optimal. The imitation learning problem then is reduced to recovering a reward function that induces the demonstrated behavior with the search algorithm serving to "stitch-together" long, coherent sequences of decisions that optimize that reward function.

We take a thoroughly probabilistic approach to reasoning about uncertainty in imitation learning. Under the constraint of matching the reward value of demonstrated behavior, we employ the principle of *maximum entropy* to resolve the ambiguity in choosing a distribution over decisions. We provide efficient algorithms for learning and inference for deterministic MDPs. We rely on an additional simplifying assumption to make reasoning about non-deterministic MDPs tractable. The resulting distribution is a probabilistic model that normalizes globally over behaviors and can be understood as an extension to chain conditional random fields that incorporates the dynamics of the planning system and extends to the infinite horizon.

Our research effort is motivated by the problem of modeling real-world routing preferences of drivers. We apply our approach to route preference modeling using 100,000 miles of collected GPS data of taxi-cab driving, where the structure of the world (i.e., the road network) is known and the actions available (i.e., traversing a road segment) are characterized by road features (e.g., speed limit, number of lanes). In sharp contrast to many imitation learning techniques, our probabilistic model of purposeful behavior integrates seamlessly with other probabilistic methods including hidden variable techniques. This allows us to extend our route preferences with hidden goals to naturally infer both future routes and destinations based on partial trajectories.

A key concern is that demonstrated behavior is prone to noise and imperfect behavior. The maximum entropy approach provides a principled method of dealing with this uncertainty. We discuss several additional advantages in modeling behavior that this technique has over existing approaches to inverse reinforcement learning including margin methods (Ratliff, Bagnell, & Zinkevich 2006) and those that normalize locally over each state's available actions (Ramachandran & Amir 2007; Neu & Szepesvri 2007).

Background

In the imitation learning setting, an agent's behavior (i.e., its trajectory or path, ζ, of states s_i and actions a_i) in some planning space is observed by a learner trying to model or imitate the agent. The agent is assumed to be attempting to optimize some function that linearly maps the features of each state, $\mathbf{f}_{s_j} \in \Re^k$, to a state *reward value* representing the agent's utility for visiting that state. This function is parameterized by some *reward weights*, θ. The reward value of a trajectory is simply the sum of state rewards, or, equivalently, the reward weight applied to the path *feature*

counts, $\mathbf{f}_\zeta = \sum_{s_j \in \zeta} \mathbf{f}_{s_j}$, which are the sum of the state features along the path.

$$\text{reward}(\mathbf{f}_\zeta) = \theta^\top \mathbf{f}_\zeta = \sum_{s_j \in \zeta} \theta^\top \mathbf{f}_{s_j}$$

The agent demonstrates single trajectories, $\tilde{\zeta}_i$, and has an expected empirical feature count, $\tilde{\mathbf{f}} = \frac{1}{m} \sum_i \mathbf{f}_{\tilde{\zeta}_i}$, based on many ($m$) demonstrated trajectories.

Recovering the agent's exact reward weights is an ill-posed problem; many reward weights, including degeneracies (e.g., all zeroes), make demonstrated trajectories optimal. Ratliff, Bagnell, & Zinkevich (2006) cast this problem as one of *structured maximum margin prediction* (MMP). They consider a class of loss functions that directly measure disagreement between an agent and a learned policy, and then efficiently learn a reward function based on a convex relaxation of this loss using the structured margin method and requiring only oracle access to an MDP solver. However, this method suffers from some significant drawbacks when no single reward function makes demonstrated behavior both optimal and significantly better than any alternative behavior. This arises quite frequently when, for instance, the behavior demonstrated by the agent is imperfect, or the planning algorithm only captures a part of the relevant state-space and cannot perfectly describe the observed behavior.

Abbeel & Ng (2004) provide an alternate approach based on Inverse Reinforcement Learning (IRL) (Ng & Russell 2000). The authors propose a strategy of matching *feature expectations* (Equation 1) between an observed policy and a learner's behavior; they demonstrate that this matching is both necessary and sufficient to achieve the same performance as the agent if the agent were in fact solving an MDP with a reward function linear in those features.

$$\sum_{\text{Path } \zeta_i} P(\zeta_i) \mathbf{f}_{\zeta_i} = \tilde{\mathbf{f}} \tag{1}$$

Unfortunately, both the IRL concept and the matching of feature counts are ambiguous. Each policy can be optimal for many reward functions (e.g., all zeros) and many policies lead to the same feature counts. When sub-optimal behavior is demonstrated, mixtures of policies are required to match feature counts, and, similarly, many different mixtures of policies satisfy feature matching. No method is proposed to resolve the ambiguity.

Maximum Entropy IRL

We take a different approach to matching feature counts that allows us to deal with this ambiguity in a principled way, and results in a single stochastic policy. We employ the principle of maximum entropy (Jaynes 1957) to resolve ambiguities in choosing distributions. This principle leads us to the distribution over behaviors constrained to match feature expectations, while being no more committed to any particular path than this constraint requires.

Deterministic Path Distributions

Unlike previous work that reasons about policies, we consider a distribution over the entire class of possible behaviors. This corresponds to paths of (potentially) variable length (Figure 1b) for deterministic MDPs (Figure 1a).

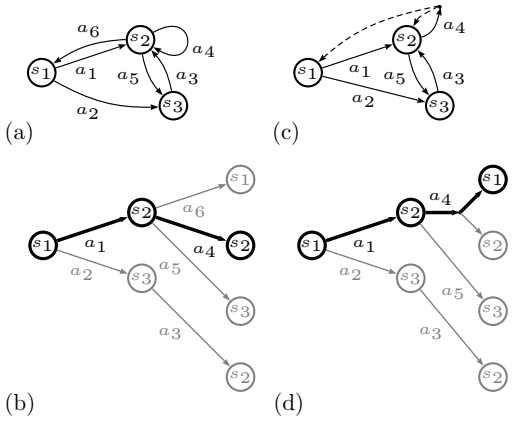


Figure 1: A deterministic MDP (a) and a single path from its path-space (b). A non-deterministic MDP (c) and a single path from its path-space (d).

Similar to distributions of policies, many different distributions of paths match feature counts when any demonstrated behavior is sub-optimal. Any one distribution from among this set may exhibit a preference for some of the paths over others that is not implied by the path features. We employ the principle of maximum entropy, which resolves this ambiguity by choosing the distribution that does not exhibit any additional preferences beyond matching feature expectations (Equation 1). The resulting distribution over paths for deterministic MDPs is parameterized by reward weights θ (Equation 2). Under this model, plans with equivalent rewards have equal probabilities, and plans with higher rewards are exponentially more preferred.

$$P(\zeta_i|\theta) = \frac{1}{Z(\theta)} e^{\theta^\top \mathbf{f}_{\zeta_i}} = \frac{1}{Z(\theta)} e^{\sum_{s_j \in \zeta_i} \theta^\top \mathbf{f}_{s_j}} \tag{2}$$

Given parameter weights, the *partition function*, $Z(\theta)$, always converges for finite horizon problems and infinite horizons problems with discounted reward weights. For infinite horizon problems with zero-reward absorbing states, the partition function can fail to converge even when the rewards of all states are negative. However, given demonstrated trajectories that are absorbed in a finite number of steps, the reward weights maximizing entropy must be convergent.

Non-Deterministic Path Distributions

In general MDPs, actions produce non-deterministic transitions between states (Figure 1c) according to the state transition distribution, T. Paths in these MDPs (Figure 1d) are now determined by the action choices of the agent and the random outcomes of the MDP. Our distribution over paths must take this randomness into account.

We use the maximum entropy distribution of paths conditioned on the transition distribution, T, and constrained to match feature expectations (Equation 1). Consider the space

of action outcomes, $\mathcal{T}$, and an outcome sample, o, specifying the next state for every action. The MDP is deterministic given o with the previous distribution (Equation 2) over paths compatible with o (i.e., the action outcomes of the path and o match). The indicator function, $I_{\zeta \in o}$ is 1 when ζ is compatible with o and 0 otherwise. Computing this distribution (Equation 3) is generally intractable. However, if we assume that transition randomness has a limited effect on behavior and that the partition function is constant for all $o \in \mathcal{T}$, then we obtain a tractable approximate distribution over paths (Equation 4).

$$P(\zeta|\theta, T) = \sum_{o \in \mathcal{T}} P_T(o) \frac{e^{\theta^\top \mathbf{f}_\zeta}}{Z(\theta, o)} I_{\zeta \in o} \quad (3)$$

$$\approx \frac{e^{\theta^\top \mathbf{f}_\zeta}}{Z(\theta, T)} \prod_{s_{t+1}, a_t, s_{t \in \zeta}} P_T(s_{t+1}|a_t, s_t) \quad (4)$$

Stochastic Policies

This distribution over paths provides a stochastic policy (i.e., a distribution over the available actions of each state) when the partition function of Equation 4 converges. The probability of an action is weighted by the expected exponentiated rewards of all paths that begin with that action.

$$P(\text{action } a|\theta, T) \propto \sum_{\zeta : a \in \zeta_{t=0}} P(\zeta|\theta, T) \quad (5)$$

Learning from Demonstrated Behavior

Maximizing the entropy of the distribution over paths subject to the feature constraints from observed data implies that we maximize the likelihood of the observed data under the maximum entropy (exponential family) distribution derived above (Jaynes 1957).

$$\theta^* = \operatorname*{argmax}_\theta L(\theta) = \operatorname*{argmax}_\theta \sum_{\text{examples}} \log P(\tilde{\zeta}|\theta, T)$$

This function is convex for deterministic MDPs and the optima can be obtained using gradient-based optimization methods. The gradient is the difference between expected empirical feature counts and the learner's expected feature counts, which can be expressed in terms of expected state visitation frequencies, D_{s_i}.

$$\nabla L(\theta) = \tilde{\mathbf{f}} - \sum_\zeta P(\zeta|\theta, T)\mathbf{f}_\zeta = \tilde{\mathbf{f}} - \sum_{s_i} D_{s_i}\mathbf{f}_{s_i} \quad (6)$$

At the maxima, the feature expectations match, guaranteeing that the learner performs equivalently to the agent's demonstrated behavior regardless of the actual reward weights the agent is attempting to optimize (Abbeel & Ng 2004).

In practice, we measure empirical, sample-based expectations of the feature values, and not the true values of the agent to be imitated. Assuming the magnitude of the features can be bounded, a standard union and Hoeffding bound argument can provide high-probability bounds on the error in feature expectations as a function of the number of samples– in particular, these bounds have only an $O(\log K)$ dependence on the number of features.[1] Dudík & Schapire (2006) show that the maximum entropy problem that results given bounded uncertainty in feature expectation is a *maximum a posteriori* problem exactly like the one described above, but with an l_1-regularizer added on (with the strength of regularization depending on the uncertainty in that feature expectation). In our experimental section we use the online exponentiated gradient descent algorithm, which is both very efficient and induces an l_1-type regularizing effect on the coefficients. [2]

Efficient State Frequency Calculations

Given the expected state frequencies, the gradient can easily be computed (Equation 6) for optimization. The most straight-forward approach for computing the expected state frequencies is based on enumerating each possible path. Unfortunately, the exponential growth of paths with the MDP's time horizon makes enumeration-based approaches computationally infeasible.

Algorithm 1 Expected Edge Frequency Calculation

Backward pass

1. Set $Z_{s_i,0} = 1$
2. Recursively compute for N iterations

$$Z_{a_{i,j}} = \sum_k P(s_k|s_i, a_{i,j}) e^{\text{reward}(s_i|\theta)} Z_{s_k}$$

$$Z_{s_i} = \sum_{a_{i,j}} Z_{a_{i,j}}$$

Local action probability computation

3. $P(a_{i,j}|s_i) = \dfrac{Z_{a_{i,j}}}{Z_{s_i}}$

Forward pass

4. Set $D_{s_i,t} = P(s_i = s_{\text{initial}})$
5. Recursively compute for $t = 1$ to N

$$D_{s_i,t+1} = \sum_{a_{i,j}} \sum_k D_{s_k,t} P(a_{i,j}|s_i) P(s_k|a_{i,j}, s_i)$$

Summing frequencies

6. $D_{s_i} = \sum_t D_{s_{i,t}}$

Instead, our algorithm computes the expected state occupancy frequencies efficiently using a technique similar to the

[1] In contrast, margin-based and locally normalizing models rely on techniques that scale linearly in the number of features.

[2] For stochastic MDPs we can achieve better usage of finite data by removing the variance in sample feature expectations due to the uncertainty in the MDP. Space doesn't permit the full exposition of the incomplete (and non-convex) log-likelihood, but the intuitive expectation-maximization algorithm that results fits the maximum-entropy model using initial feature expectations and then improves those estimates by running the resulting policy in the MDP.

forward-backward algorithm for Conditional Random Fields or value iteration in Reinforcement Learning. The algorithm approximates the state frequencies for the infinite time horizon using a large fixed time horizon. It recursively "backs up" from each possible terminal state (Step 1) and computes the probability mass associated with each branch along the way (Step 2) by computing the partition function for Equation 4 at each action and state. These branching values yield local action probabilities (Step 3), from which state frequencies in each timestep can be computed (Steps 4 and 5) and summed for the total state frequency counts (Step 6).

Driver Route Modeling

Our research effort on maximum entropy approaches to IRL was motivated by applications of imitation learning of driver route choices. We are interested in recovering a utility function useful for *predicting driving behavior* as well as for *route recommendation*. To our knowledge, this is the largest-scale IRL problem investigated to date in terms of demonstrated data size.

Route Choice as an MDP

Road networks present a large planning space with known structure. We model this structure for the road network surrounding Pittsburgh, Pennsylvania, as a deterministic MDP with over 300,000 states (i.e., road segments) and 900,000 actions (i.e., transitions at intersections). We assume that drivers who are executing plans within the road network are attempting to reach some goal while efficiently optimizing some trade-off between time, safety, stress, fuel costs, maintenance costs, and other factors. We call this value a *cost* (i.e., a negative reward). We represent the destination within the MDP as an absorbing state where no additional costs are incurred. Different trips have different destinations and slightly different corresponding MDPs. We assume that the reward weight is independent of the goal state and therefore a single reward weight can be learned from many MDPs that differ only in goal state.

Collecting and Processing GPS Data

We collected GPS trace data from 25 Yellow Cab taxi drivers over a 12 week duration at all times of day. This yielded a dataset of over 100,000 miles of travel collected during over 3,000 hours of driving and covering a large area surrounding Pittsburgh. We employed a particle filter to fit the sparse GPS data to the road network and segmented the fitted traces into approximately 13,000 distinct trips using a time-based threshold to determine stopping locations. We discarded roughly 30% of the trips that were too short (fewer than 10 road segments), too cyclic, or too noisy, and split 20% of the remaining trips into a training set and the remaining 80% of the data into a testing set of 7403 examples.

Path Features

Our road network data includes a detailed set of characteristics that describe each road segment. For our experiments, we consider four different dimensions of characteristics: road type, speed, lanes, and transitions. A road segment is categorized in each of these dimensions (i.e., from interstate to local road, high speed to low speed, and one lane to many lanes) and transitions are categorized as straight, left, right, hard left, and hard right. A path is described by how many miles of each road segment categorization it contains and the number of each transition type. Each road segment's contribution to these 22 different counts is represented in the road segment's features.

IRL Models

We apply our Maximum Entropy IRL model (MaxEnt) to the task of learning taxi drivers' collective utility function for the different features describing paths in our road network. We maximize the probability of demonstrated paths within a smaller fixed class of reasonably good paths rather than the class of all possible paths below a fixed length. Our algorithm is efficient (polynomial time) for both classes, but this reduction provides a significant speed up (without introducing optimization non-convexity) and limits consideration of cycles in the road network.

We demonstrate our approach's effectiveness by comparing with two other IRL models. The first is Maximum Margin Planning (MMP) (Ratliff, Bagnell, & Zinkevich 2006), which is a model capable of predicting new paths, but incapable of density estimation (i.e., computing the probability of some demonstrated path). The second model is an action-based distribution model (Action) that has been employed for Bayesian IRL (Ramachandran & Amir 2007) and hybrid IRL (Neu & Szepesvri 2007). The choice of action in any particular state is assumed to be distributed according to the future expected reward of the best policy after taking the action, $Q^*(S, a)$. In our setting, this value is simply the optimal path cost to the goal after taking a particular action.

$$P(\text{action } a|s_i, \theta) \propto e^{Q^*(s_i,a)} \tag{7}$$

The difference between this action-based model and our model is best illustrated in the following example.

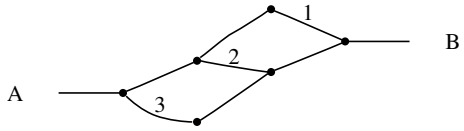


Figure 2: Example of probability distributions over paths.

There are three obvious paths from **A** to **B** in Figure 2. Assuming each path provides the same reward, in the maximum entropy model, each path will have equal probability. In the action-based model, path 3 will have 50% probability while paths 1 and 2 have 25% probability. The distribution will be different for the return trip from **B** to **A**.

More generally, paths in action-based distributions such as this one only compete for probability mass with other paths locally at the action level, and not against other paths that branched earlier. This problem is known as label bias in the Conditional Random Field literature (Lafferty, McCallum, & Pereira 2001). It has undesirable consequences

for IRL. For instance, the highest reward policy may not be the most probable policy in the model, and policies with the same expected reward can have different probabilities. Compared to our maximum entropy distribution over paths, this model gives higher probability mass to paths with a smaller branching factor and lower probability mass to those with a higher branching factor.

Comparative Evaluation

We now evaluate each model's ability to model paths in the withheld testing set after being trained on the training set given the path's origin and destination. We use three different metrics. The first compares the model's most likely path estimate with the actual demonstrated path and evaluates the amount of route distance shared. The second shows what percentage of the testing paths match at least 90% (distance) with the model's predicted path. The final metric measures the average log probability of paths in the training set under the given model. For path matching, we evaluate both the most likely path within the action-based model and the lowest cost path using the weights learned from the action-based model. We additionally evaluate a model based on expected travel times that weights the cost of a unit distance of road to be inversely proportional to the speed of the road, and predicts the fastest (i.e., lowest cost) route given these costs.

	Matching	90% Match	Log Prob
Time-based	72.38%	43.12%	N/A
Max Margin	75.29%	46.56%	N/A
Action	77.30%	50.37%	-7.91
Action (costs)	77.74%	50.75%	N/A
MaxEnt paths	**78.79%**	**52.98%**	**-6.85**

Table 1: Comparison of different models' abilities to match most likely path predictions to withheld paths (average percentage of distance matching and percentage of examples where at least 90% of the paths' distances match) and the probability of withheld paths (average log probability).

The results of this analysis are shown in Table 1. For each of these metrics, our maximum entropy model shows significant ($\alpha < .01$) improvements over the other models.

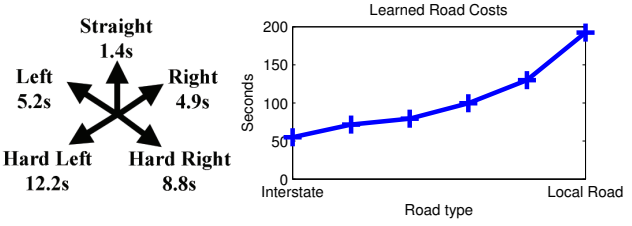


Figure 3: Learned costs of turns (left) and miles of different road types (right) normalized to seconds (with interstate driving fixed to 65 miles per hour).

The learned cost values using our MaxEnt model are shown in Figure 3. Additionally, we learn a fixed per edge cost of 1.4 seconds that helps to penalize paths composed of many short roads.

Applications

Beyond the route recommendation application described above, our approach opens up a range of possibilities for driver prediction. Route recommendation can be easily personalized based on passive observation of a single user. Further, by learning a probability distribution over driver preferences, destinations, and routes the MaxEntIRL model of driver behavior can go beyond route recommendation, to new queries like: "What is the probability the driver will take this street?" This enables a range of new applications, including, e.g., warning drivers about unanticipated traffic problems on their route *without* ever explicitly having to query the user about route or destination; optimizing battery and fuel consumption in a hybrid vehicle; and activating temperature controls at a home prior to the driver's arrival.

So far, we have not described situations where the driver's intended destination is unknown. Fortunately we can reason easily about intended destinations by applying Bayes' theorem to our model of route preference. Consider the case where we want the posterior probability of a set of destinations given a partially traveled path from A to B.

$$P(dest|\tilde{\zeta}_{A\to B}) \propto P(\tilde{\zeta}_{A\to B}|dest)P(dest)$$
$$\propto \frac{\sum_{\zeta_{B\to dest}} e^{\theta^\top \mathbf{f}_\zeta}}{\sum_{\zeta_{A\to dest}} e^{\theta^\top \mathbf{f}_\zeta}} P(dest)$$

These quantities can easily be computed using our inference algorithm (Algorithm 1).

Figure 4: Destination distribution (from 5 destinations) and remaining path distribution given partially traveled path. The partially traveled path is heading westward, which is a very inefficient (i.e., improbable) partial route to any of the eastern destinations (3, 4, 5). The posterior destination probability is split between destinations 1 and 2 primarily based on the prior distribution on destinations.

Figure 4 shows one particular destination prediction problem. We evaluate our model's ability to predict destinations for routes terminating in one of five locations around the city (Figure 4) based on the fraction of total route observed (Figure 5). We use a training set to form a prior over destinations and evaluate our model on a withheld test set. Incorporating additional contextual information into this prior distribution, like time of day, will be beneficial for predicting the destinations of most drivers.

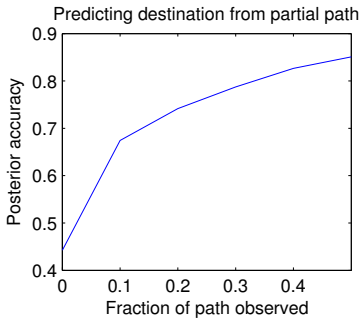


Figure 5: Posterior prediction accuracy over five destinations given partial path.

Related Work

In locally normalizing probabilistic IRL models, probability mass is assigned to each action based on some summary statistic. The value of the optimal policy has been employed (Neu & Szepesvri 2007; Ramachandran & Amir 2007). Beyond this probability mass assignment, paths prefixed with one action do not compete for probability mass with paths prefixed by other actions. The effect, known as label bias in the CRF literature (Lafferty, McCallum, & Pereira 2001), is that paths in portions of the state space with many branches will be biased towards lower probability mass while those with fewer branches will be biased towards higher probability mass. As a consequence, the highest reward behavior in an MDP may not be the most probable, and behaviors that match in expected reward need not match in probability. Our model avoids the label bias problem, giving equivalent probability to behaviors with equivalent expected reward, and larger probability to higher reward behavior. Further, we note that the models suggested lead to potentially difficult non-convex optimization problems with multiple minima.

Route preference modeling has been studied using a few different approaches. Liao *et al.* (2007) model transportation decisions using a directed graphical model. Local action distributions are learned from demonstrated behavior captured in GPS traces. While this model can represent the same distributions as our undirected model, it is much less efficient. Contextual information, like road closures, can influence action probability distribution throughout the entire road network. Consequentially, a different set of action distributions must be learned for every destination and possible context, leading to estimates based on very sparse amounts of data.

Krumm & Horvitz (2006) use route efficiency of partial routes to varying destinations to perform destination prediction of a driver. This same notion of efficiency is captured within our probabilistic model. The TRIP system (Letchner, Krumm, & Horvitz 2006) learns the time inefficiency values drivers are willing to accept for each of their traveled routes and discounts the costs of these previously traveled road segments for each user by the level of accepted inefficiency, implicitly capturing some of their preferences. Our IRL formalization of the problem can be viewed as an extension to this work that not only enables portions of desired routes to have lowered costs, but also increases the costs of undesirable routes. Additionally, approaching the problem in a parametric fashion allows our model to efficiently incorporate contextual information by learning drivers' preferences of those contexts, and to generalize to previously un-encountered road networks.

Conclusions and Future Work

We present a novel approach to inverse reinforcement and imitation learning that cleanly resolves ambiguities in previous approaches, provides a convex, computationally efficient procedure for optimization and maintains important performance guarantees. We applied our method to the problem of modeling route preferences, but we focused primarily on describing and evaluating the differences between our model and other imitation learning models using a small feature space. In future work, we plan to improve our model by incorporating contextual factors (e.g., time of day, weather) into our feature space, and inducing region-based or even specific road based features that can explain, e.g., the avoidance of a particular road only during rush hour, or a steep road during winter weather.

References

Abbeel, P., and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proc. ICML*, 1–8.

Dudík, M., and Schapire, R. E. 2006. Maximum entropy distribution estimation with generalized regularization. In *Proc. COLT*, 123–138.

Jaynes, E. T. 1957. Information theory and statistical mechanics. *Physical Review* 106:620–630.

Krumm, J., and Horvitz, E. 2006. Predestination: Inferring destinations from partial trajectories. In *Proc. Ubicomp*, 243–260.

Lafferty, J.; McCallum, A.; and Pereira, F. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*, 282–289.

Letchner, J.; Krumm, J.; and Horvitz, E. 2006. Trip router with individualized preferences (trip): Incorporating personalization into route planning. In *Proc. IAAI*, 1795–1800.

Liao, L.; Patterson, D. J.; Fox, D.; and Kautz, H. 2007. Learning and inferring transportation routines. *Artificial Intelligence* 171(5-6):311–331.

Neu, G., and Szepesvri, C. 2007. Apprenticeship learning using inverse reinforcement learning and gradient methods. In *Proc. UAI*, 295–302.

Ng, A. Y., and Russell, S. 2000. Algorithms for inverse reinforcement learning. In *Proc. ICML*, 663–670.

Ramachandran, D., and Amir, E. 2007. Bayesian inverse reinforcement learning. In *Proc. IJCAI*, 2586–2591.

Ratliff, N.; Bagnell, J. A.; and Zinkevich, M. 2006. Maximum margin planning. In *Proc. ICML*, 729–736.

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 34^{th} 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

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

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

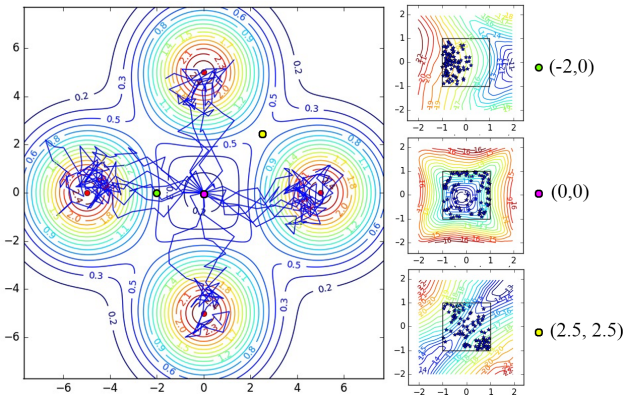


Figure 1. Illustration of the 2D multi-goal environment. Left: trajectories from a policy learned with our method (solid blue lines). The x and y axes correspond to 2D positions (states). The agent is initialized at the origin. The goals are depicted as red dots, and the level curves show the reward. 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.

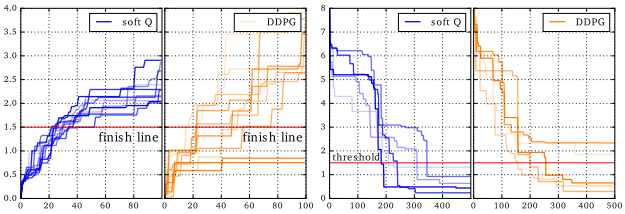
5.2. Learning Multi-Modal Policies for Exploration

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

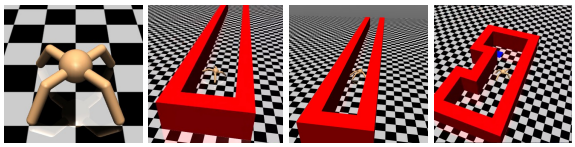
Figure 2. Simulated robots used in our experiments.



(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.

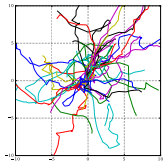


(a) (b) (c) (d)

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.

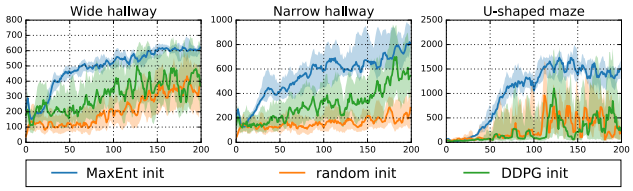


Figure 5. Performance in the downstream task with fine-tuning (MaxEnt) or training from scratch (DDPG). The x-axis shows the training iterations. The y-axis shows the average discounted return. Solid lines are average values over 10 random seeds. Shaded regions correspond to one standard deviation.

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).

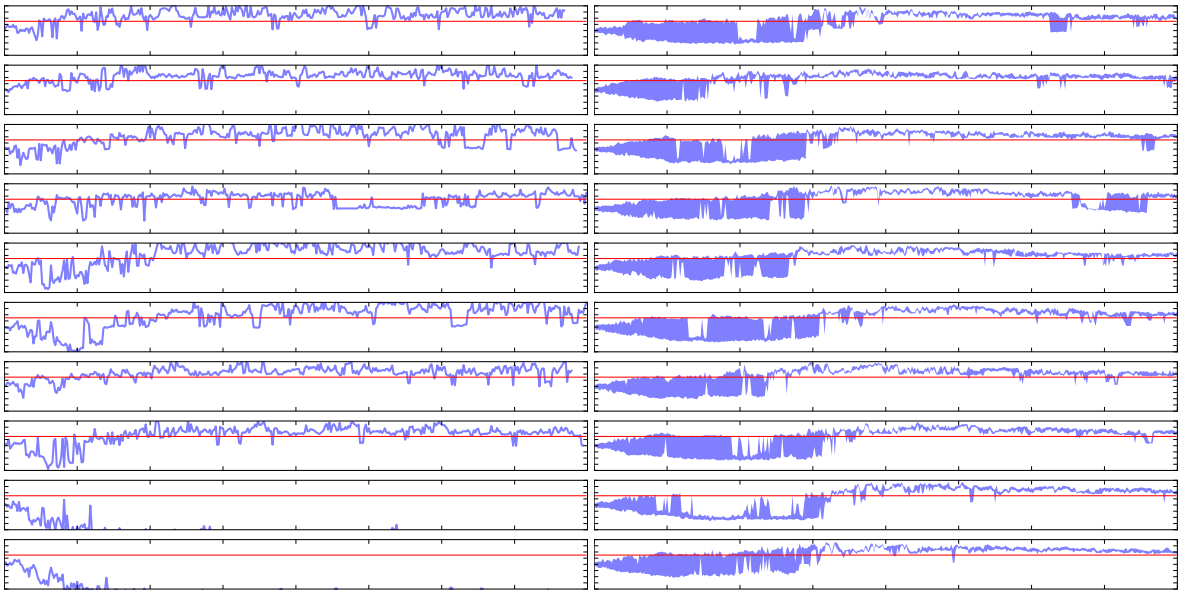
The entropy coefficient α is 10 for multi-goal environment, and 0.1 for the swimming snake, maze, hallway (pretraining) and U-shaped maze (pretraining) experiments.

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.

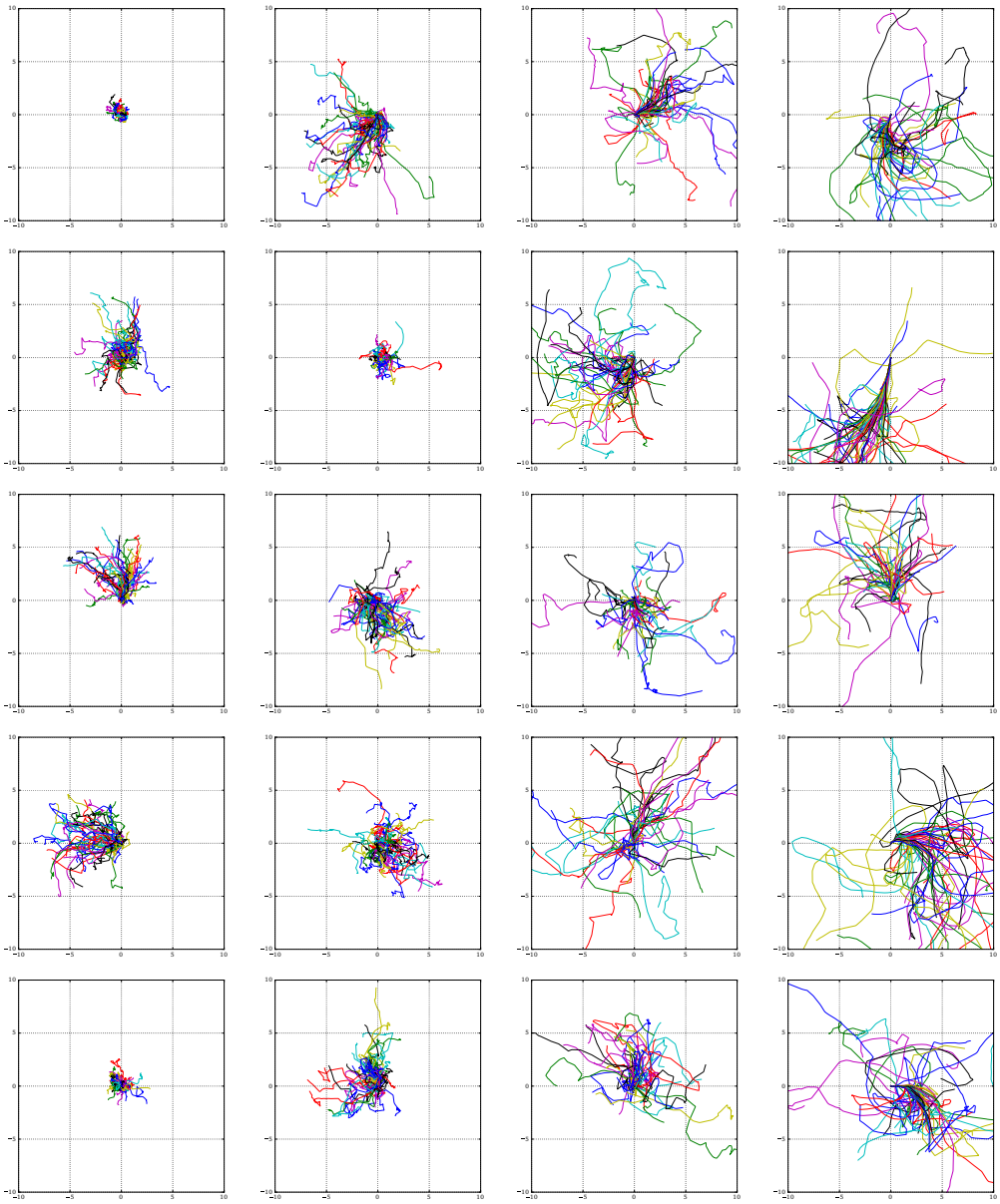


Figure 7. The plot shows trajectories of the quadrupedal robot during maximum entropy pretraining. The robot has diverse behavior and explores multiple directions. The four columns correspond to entropy coefficients $\alpha = 10, 1, 0.1, 0.01$ respectively. Different rows correspond to policies trained 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.

Equivalence Between Policy Gradients and Soft Q-Learning

John Schulman[1], Xi Chen[1,2], and Pieter Abbeel[1,2]

[1]OpenAI [2]UC Berkeley, EECS Dept.

{*joschu, peter, pieter*}*@openai.com*

Abstract

Two of the leading approaches for model-free reinforcement learning are policy gradient methods and Q-learning methods. Q-learning methods can be effective and sample-efficient when they work, however, it is not well-understood why they work, since empirically, the Q-values they estimate are very inaccurate. A partial explanation may be that Q-learning methods are secretly implementing policy gradient updates: we show that there is a precise equivalence between Q-learning and policy gradient methods in the setting of entropy-regularized reinforcement learning, that "soft" (entropy-regularized) Q-learning is exactly equivalent to a policy gradient method. We also point out a connection between Q-learning methods and natural policy gradient methods.

Experimentally, we explore the entropy-regularized versions of Q-learning and policy gradients, and we find them to perform as well as (or slightly better than) the standard variants on the Atari benchmark. We also show that the equivalence holds in practical settings by constructing a Q-learning method that closely matches the learning dynamics of A3C without using a target network or ϵ-greedy exploration schedule.

1 Introduction

Policy gradient methods (PG) and Q-learning (QL) methods perform updates that are qualitatively similar. In both cases, if the return following an action a_t is high, then that action is reinforced: in policy gradient methods, the probability $\pi(a_t \mid s_t)$ is increased; whereas in Q-learning methods, the Q-value $Q(s_t, a_t)$ is increased. The connection becomes closer when we add entropy regularization to these algorithms. With an entropy cost added to the returns, the optimal policy has the form $\pi(a \mid s) \propto \exp(Q(s, a))$; hence policy gradient methods solve for the optimal Q-function, up to an additive constant (Ziebart [2010]). O'Donoghue et al. [2016] also discuss the connection between the fixed points and updates of PG and QL methods, though the discussion of fixed points is restricted to the tabular setting, and the discussion comparing updates is informal and shows an approximate equivalence. Going beyond past work, this paper shows that under appropriate conditions, the gradient of the loss function used in n-step Q-learning is equal to the gradient of the loss used in an n-step policy gradient method, including a squared-error term on the value function. Altogether, the update matches what is typically done in "actor-critic" policy gradient methods such as A3C, which explains why Mnih et al. [2016] obtained qualitatively similar results from policy gradients and n-step Q-learning.

Section 2 uses the bandit setting to provide the reader with a simplified version of our main calculation. (The main calculation applies to the MDP setting.) Section 3 discusses the entropy-regularized formulation of RL, which is not original to this work, but is included for the reader's convenience. Section 4 shows that the soft Q-learning loss gradient can be interpreted as a policy gradient term plus a baseline-error-gradient term, corresponding to policy gradient instantiations such as A3C [Mnih et al., 2016]. Section 5 draws a connection between QL methods that use batch updates or replay-buffers, and natural policy gradient methods.

Some previous work on entropy regularized reinforcement learning (e.g., O'Donoghue et al. [2016], Nachum et al. [2017]) uses entropy bonuses, whereas we use a penalty on Kullback-Leibler (KL) divergence, which is a bit more general. However, in the text, we often refer to "entropy" terms; this refers to "relative entropy", i.e., the KL divergence.

2 Bandit Setting

Let's consider a bandit problem with a discrete or continuous action space: at each timestep the agent chooses an action a, and the reward r is sampled according to $P(r \mid a)$, where P is unknown to the agent. Let

$\bar{r}(a) = \mathbb{E}[r \mid a]$, and let π denote a policy, where $\pi(a)$ is the probability of action a. Then, the expected per-timestep reward of the policy π is $\mathbb{E}_{a\sim\pi}[r] = \sum_a \pi(a)\bar{r}(a)$ or $\int \mathrm{d}a\, \pi(a)\bar{r}(a)$. Let's suppose we are maximizing $\eta(\pi)$, an entropy-regularized version of this objective:

$$\eta(\pi) = \mathbb{E}_{a\sim\pi,r}[r] - \tau D_{\mathrm{KL}}[\pi \parallel \overline{\pi}] \tag{1}$$

where $\overline{\pi}$ is some "reference" policy, τ is a "temperature" parameter, and D_{KL} is the Kullback-Leibler divergence. Note that the temperature τ can be eliminated by rescaling the rewards. However, we will leave it so that our calculations are checkable through dimensional analysis, and to make the temperature-dependence more explicit.

First, let us calculate the policy π that maximizes η. We claim that $\eta(\pi)$ is maximized by $\pi^{\mathcal{B}}_{\bar{r}}$, defined as

$$\pi^{\mathcal{B}}_{\bar{r}}(a) = \overline{\pi}(a)\exp(\bar{r}(a)/\tau) / \underbrace{\mathbb{E}_{a'\sim\overline{\pi}}[\exp(\bar{r}(a')/\tau)]}_{\text{normalizing constant}}. \tag{2}$$

To derive this, consider the KL divergence between π and $\pi^{\mathcal{B}}_{\bar{r}}$:

$$\begin{aligned} D_{\mathrm{KL}}\left[\pi \,\middle\|\, \pi^{\mathcal{B}}_{\bar{r}}\right] &= \mathbb{E}_{a\sim\pi}\left[\log\pi(a) - \log\pi^{\mathcal{B}}_{\bar{r}}(a)\right] && (3)\\ &= \mathbb{E}_{a\sim\pi}\left[\log\pi(a) - \log\overline{\pi}(a) - \bar{r}(a)/\tau + \log\mathbb{E}_{a\sim\overline{\pi}}[\exp(\bar{r}(a)/\tau)]\right] && (4)\\ &= D_{\mathrm{KL}}[\pi \parallel \overline{\pi}] - \mathbb{E}_{a\sim\pi}[\bar{r}(a)/\tau] + \log\mathbb{E}_{a\sim\overline{\pi}}[\exp(\bar{r}(a)/\tau)] && (5)\end{aligned}$$

Rearranging and multiplying by τ,

$$\mathbb{E}_{a\sim\pi}[\bar{r}(a)] - \tau D_{\mathrm{KL}}[\pi \parallel \overline{\pi}] = \tau\log\mathbb{E}_{a\sim\overline{\pi}}[\exp(\bar{r}(a)/\tau)] - \tau D_{\mathrm{KL}}\left[\pi \,\middle\|\, \pi^{\mathcal{B}}_{\bar{r}}\right] \tag{6}$$

Clearly the left-hand side is maximized (with respect to π) when the KL term on the right-hand side is minimized (as the other term does not depend on π), and $D_{\mathrm{KL}}\left[\pi \,\middle\|\, \pi^{\mathcal{B}}_{\bar{r}}\right]$ is minimized at $\pi = \pi^{\mathcal{B}}_{\bar{r}}$.

The preceding calculation gives us the optimal policy when $\bar{r}$ is known, but in the entropy-regularized bandit problem, it is initially unknown, and the agent learns about it by sampling. There are two approaches for solving the entropy-regularized bandit problem:

1. A direct, policy-based approach, where we incrementally update the agent's policy π based on stochastic gradient ascent on η.
2. An indirect, value-based approach, where we learn an action-value function q_θ that estimates and approximates $\bar{r}$, and we define π based on our current estimate of q_θ.

For the policy-based approach, we can obtain unbiased estimates the gradient of η. For a parameterized policy π_θ, the gradient is given by

$$\nabla_\theta\eta(\pi_\theta) = \mathbb{E}_{a\sim\pi_\theta,r}[\nabla_\theta\log\pi_\theta(a)r - \tau\nabla_\theta D_{\mathrm{KL}}[\pi_\theta \parallel \overline{\pi}]]. \tag{7}$$

We can obtain an unbiased gradient estimate using a single sample (a, r).

In the indirect, value-based approach approach, it is natural to use a squared-error loss:

$$L_\pi(\theta) := \tfrac{1}{2}\mathbb{E}_{a\sim\pi,r}\left[(q_\theta(a) - r)^2\right] \tag{8}$$

Taking the gradient of this loss, with respect to the parameters of q_θ, we get

$$\nabla_\theta L_\pi(\theta) = \mathbb{E}_{a\sim\pi,r}[\nabla_\theta q_\theta(a)(q_\theta(a) - r)] \tag{9}$$

Soon, we will calculate the relationship between this loss gradient and the policy gradient from Equation (7).

In the indirect, value-based approach, a natural choice for policy π is the one that would be optimal if $q_\theta = \bar{r}$. Let's denote this policy, called the Boltzmann policy, by $\pi^{\mathcal{B}}_{q_\theta}$, where

$$\pi^{\mathcal{B}}_{q_\theta}(a) = \overline{\pi}(a)\exp(q_\theta(a)/\tau)/\mathbb{E}_{a'\sim\overline{\pi}}[\exp(q_\theta(a')/\tau)]. \tag{10}$$

It will be convenient to introduce a bit of notation for the normalizing factor; namely, we define the scalar

$$v_\theta = \tau \log \mathbb{E}_{a\sim\overline{\pi}} \left[\exp(q_\theta(a))/\tau\right] \tag{11}$$

Then the Boltzmann policy can be written as

$$\pi^{\mathcal{B}}_{q_\theta}(a) = \overline{\pi}(a) \exp((q_\theta(a) - v_\theta)/\tau). \tag{12}$$

Note that the term $\tau \log \mathbb{E}_{a\sim\overline{\pi}} \left[\exp(\bar{r}(a)/\tau)\right]$, appeared earlier in Equation (6)). Repeating the calculation from Equation (2) through Equation (6), but with q_θ instead of $\bar{r}$,

$$v_\theta = \mathbb{E}_{a\sim\pi^{\mathcal{B}}_{q_\theta}} \left[q_\theta(a)\right] - \tau D_{\mathrm{KL}} \left[\pi^{\mathcal{B}}_{q_\theta} \,\middle\|\, \overline{\pi}\right]. \tag{13}$$

Hence, v_θ is an estimate of $\eta(\pi^{\mathcal{B}}_{q_\theta})$, plugging in q_θ for $\bar{r}$.

Now we shall show the connection between the gradient of the squared-error loss (Equation (9)) and the policy gradient (Equation (7)). Rearranging Equation (12), we can write q_θ in terms of v_θ and the Boltzmann policy $\pi^{\mathcal{B}}_{q_\theta}$:

$$q_\theta(a) = v_\theta + \tau \log\left(\frac{\pi^{\mathcal{B}}_{q_\theta}(a)}{\overline{\pi}(a)}\right) \tag{14}$$

Let's substitute this expression for q_θ into the squared-error loss gradient (Equation (9)).

$$\nabla_\theta L_\pi(q_\theta) = \mathbb{E}_{a\sim\pi,r} \left[\nabla_\theta q_\theta(a)(q_\theta(a) - r)\right] \tag{15}$$

$$= \mathbb{E}_{a\sim\pi,r} \left[\nabla_\theta \left(v_\theta + \tau \log\left(\frac{\pi^{\mathcal{B}}_{q_\theta}(a)}{\overline{\pi}(a)}\right)\right)\left(v_\theta + \tau \log\left(\frac{\pi^{\mathcal{B}}_{q_\theta}(a)}{\overline{\pi}(a)}\right) - r\right)\right] \tag{16}$$

$$= \mathbb{E}_{a\sim\pi,r} \left[\tau \nabla_\theta \log \pi^{\mathcal{B}}_{q_\theta}(a)\left(v_\theta + \tau \log\left(\frac{\pi^{\mathcal{B}}_{q_\theta}(a)}{\overline{\pi}(a)}\right) - r\right) + \nabla_\theta v_\theta \left(v_\theta + \tau \log\left(\frac{\pi^{\mathcal{B}}_{q_\theta}(a)}{\overline{\pi}(a)}\right) - r\right)\right] \tag{17}$$

Note that we have not yet decided on a sampling distribution π. Henceforth, we'll assume actions were sampled by $\pi = \pi^{\mathcal{B}}_{q_\theta}$. Also, note the derivative of the KL-divergence:

$$\nabla_\theta D_{\mathrm{KL}} \left[\pi^{\mathcal{B}}_{q_\theta} \,\middle\|\, \overline{\pi}\right] = \nabla_\theta \int \mathrm{d}a \; \pi^{\mathcal{B}}_{q_\theta}(a) \log\left(\frac{\pi^{\mathcal{B}}_{q_\theta}(a)}{\overline{\pi}(a)}\right) \tag{18}$$

$$= \int \mathrm{d}a \; \nabla_\theta \pi^{\mathcal{B}}_{q_\theta}(a) \left(\log\left(\frac{\pi^{\mathcal{B}}_{q_\theta}(a)}{\overline{\pi}(a)}\right) + \pi^{\mathcal{B}}_{q_\theta}(a) \frac{1}{\pi^{\mathcal{B}}_{q_\theta}(a)}\right) \tag{19}$$

moving gradient inside and using identity $\int \mathrm{d}a \; \nabla_\theta \pi^{\mathcal{B}}_{q_\theta}(a) = 0$

$$= \int \mathrm{d}a \; \pi^{\mathcal{B}}_{q_\theta}(a) \nabla_\theta \log \pi^{\mathcal{B}}_{q_\theta}(a) \log\left(\frac{\pi^{\mathcal{B}}_{q_\theta}(a)}{\overline{\pi}(a)}\right) \tag{20}$$

$$= \mathbb{E}_{a\sim\pi^{\mathcal{B}}_{q_\theta}} \left[\nabla_\theta \log \pi^{\mathcal{B}}_{q_\theta}(a) \log\left(\frac{\pi^{\mathcal{B}}_{q_\theta}(a)}{\overline{\pi}(a)}\right)\right] \tag{21}$$

Continuing from Equation (17) but setting $\pi = \pi^{\mathcal{B}}_{q_\theta}$,

$$\nabla_\theta L_\pi(q_\theta)\big|_{\pi=\pi^{\mathcal{B}}_{q_\theta}} = \mathbb{E}_{a\sim\pi^{\mathcal{B}}_{q_\theta},r} \left[\tau \nabla_\theta \log \pi^{\mathcal{B}}_{q_\theta}(a)(v_\theta - r) + \tau^2 \nabla_\theta D_{\mathrm{KL}} \left[\pi^{\mathcal{B}}_{q_\theta} \,\middle\|\, \overline{\pi}\right]\right]$$
$$+ \nabla_\theta \mathbb{E}_{a\sim\pi^{\mathcal{B}}_{q_\theta},r} \left[v_\theta \left(v_\theta + \tau D_{\mathrm{KL}} \left[\pi^{\mathcal{B}}_{q_\theta} \,\middle\|\, \overline{\pi}\right] - r\right)\right] \tag{22}$$

$$= -\tau \underbrace{\nabla_\theta \mathbb{E}_{a\sim\pi^{\mathcal{B}}_{q_\theta},r} \left[r - \tau D_{\mathrm{KL}} \left[\pi^{\mathcal{B}}_{q_\theta} \,\middle\|\, \overline{\pi}\right]\right]}_{\text{policy gradient}} + \underbrace{\nabla_\theta \mathbb{E}_{a\sim\pi,r} \left[\tfrac{1}{2}(v_\theta - (r - \tau D_{\mathrm{KL}} \left[\pi \,\middle\|\, \overline{\pi}\right]))^2\right]}_{\text{value error gradient}}\Big|_{\pi=\pi^{\mathcal{B}}_{q_\theta}} \tag{23}$$

Hence, the gradient of the squared error for our action-value function can be broken into two parts: the first part is the policy gradient of the Boltzmann policy corresponding to q_θ, the second part arises from a squared error objective, where we are fitting v_θ to the entropy-augmented expected reward $\bar{r}(a) - \tau D_{\mathrm{KL}} \left[\pi^{\mathcal{B}}_{q_\theta} \,\middle\|\, \overline{\pi}\right]$.

Soon we will derive an equivalent interpretation of Q-function regression in the MDP setting, where we are approximating the state-value function $Q^{\pi,\gamma}$. However, we first need to introduce an entropy-regularized version of the reinforcement learning problem.

3 Entropy-Regularized Reinforcement Learning

We shall consider an entropy-regularized version of the reinforcement learning problem, following various prior work (Ziebart [2010], Fox et al. [2015], Haarnoja et al. [2017], Nachum et al. [2017]). Specifically, let us define the *entropy-augmented return* to be $\sum_{t=0}^{\infty} \gamma^t (r_t - \tau \operatorname{KL}_t)$ where r_t is the reward, $\gamma \in [0, 1]$ is the discount factor, τ is a scalar temperature coefficient, and KL_t is the Kullback-Leibler divergence between the current policy π and a reference policy $\overline{\pi}$ at timestep t: $\operatorname{KL}_t = D_{\mathrm{KL}}\left[\pi(\cdot \mid s_t) \parallel \overline{\pi}(\cdot \mid s_t)\right]$. We will sometimes use the notation $\operatorname{KL}(s) = D_{\mathrm{KL}}\left[\pi \parallel \overline{\pi}\right](s) = D_{\mathrm{KL}}\left[\pi(\cdot \mid s) \parallel \overline{\pi}(\cdot \mid s)\right]$. To emulate the effect of a standard entropy bonus (up to a constant), one can define $\overline{\pi}$ to be the uniform distribution. The subsequent sections will generalize some of the concepts from reinforcement learning to the setting where we are maximizing the entropy-augmented discounted return.

3.1 Value Functions

We are obliged to alter our definitions of value functions to include the new KL penalty terms. We shall define the state-value function as the expected return:

$$V_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t (r_t - \tau \operatorname{KL}_t) \,\middle|\, s_0 = s\right] \tag{24}$$

and we shall define the Q-function as

$$Q_\pi(s, a) = \mathbb{E}\left[r_0 + \sum_{t=1}^{\infty} \gamma^t (r_t - \tau \operatorname{KL}_t) \,\middle|\, s_0 = s, a_0 = a\right] \tag{25}$$

Note that this Q-function does not include the first KL penalty term, which does not depend on the action a_0. This definition makes some later expressions simpler, and it leads to the following relationship between Q_π and V_π:

$$V_\pi(s) = \mathbb{E}_{a \sim \pi}\left[Q_\pi(s, a)\right] - \tau \operatorname{KL}(s), \tag{26}$$

which follows from matching terms in the sums in Equations (24) and (25).

3.2 Boltzmann Policy

In standard reinforcement learning, the "greedy policy" for Q is defined as $[\mathcal{G}Q](s) = \arg\max_a Q(s, a)$. With entropy regularization, we need to alter our notion of a greedy policy, as the optimal policy is stochastic. Since Q_π omits the first entropy term, it is natural to define the following stochastic policy, which is called the Boltzmann policy, and is analogous to the greedy policy:

$$\pi_Q^{\mathcal{B}}(\cdot \mid s) = \arg\max_\pi \{\mathbb{E}_{a \sim \pi}\left[Q(s, a)\right] - \tau D_{\mathrm{KL}}\left[\pi \parallel \overline{\pi}\right](s)\} \tag{27}$$

$$= \overline{\pi}(a \mid s) \exp(Q(s, a)/\tau) / \underbrace{\mathbb{E}_{a' \sim \overline{\pi}}\left[\exp(Q(s, a')/\tau)\right]}_{\text{normalizing constant}}. \tag{28}$$

where the second equation is analogous to Equation (2) from the bandit setting.

Also analogously to the bandit setting, it is natural to define V_Q (a function of Q) as

$$V_Q(s) = \tau \log \mathbb{E}_{a' \sim \overline{\pi}}\left[\exp(Q(s, a')/\tau)\right] \tag{29}$$

so that

$$\pi_Q^{\mathcal{B}}(a \mid s) = \overline{\pi}(a \mid s) \exp((Q(s, a) - V_Q(s))/\tau) \tag{30}$$

Under this definition, it also holds that

$$V_Q(s) = \mathbb{E}_{a \sim \pi_Q^{\mathcal{B}}(s)}\left[Q(s, a)\right] - \tau D_{\mathrm{KL}}\left[\pi_Q^{\mathcal{B}} \,\middle\|\, \overline{\pi}\right](s) \tag{31}$$

in analogy with Equation (13). Hence, $V_Q(s)$ can be interpreted as an estimate of the expected entropy-augmented return, under the Boltzmann policy $\pi_Q^{\mathcal{B}}$.

Another way to interpret the Boltzmann policy is as the exponentiated advantage function. Defining the advantage function as $A_Q(s,a) = Q(s,a) - V_Q(s)$, Equation (30) implies that $\frac{\pi_Q^{\mathcal{B}}(a \mid s)}{\overline{\pi}(a \mid s)} = \exp(A_Q(s,a)/\tau)$.

3.3 Fixed-Policy Backup Operators

The $\mathcal{T}_\pi$ operators (for Q and V) in standard reinforcement learning correspond to computing the expected return with a one-step lookahead: they take the expectation over one step of dynamics, and then fall back on the value function at the next timestep. We can easily generalize these operators to the entropy-regularized setting. We define

$$[\mathcal{T}_\pi V](s) = \mathbb{E}_{a\sim\pi,(r,s')\sim P(r,s' \mid s,a)}\left[r - \tau\,\mathrm{KL}(s) + \gamma V(s')\right] \tag{32}$$

$$[\mathcal{T}_\pi Q](s,a) = \mathbb{E}_{(r,s')\sim P(r,s' \mid s,a)}\left[r + \gamma(\mathbb{E}_{a'\sim\pi}\left[Q(s',a')\right] - \tau\,\mathrm{KL}(s'))\right]. \tag{33}$$

Repeatedly applying the $\mathcal{T}_\pi$ operator ($\mathcal{T}_\pi^n V = \underbrace{\mathcal{T}_\pi(\mathcal{T}_\pi(\ldots\mathcal{T}_\pi}_{n \text{ times}}(V))))$ corresponds to computing the expected return with a multi-step lookahead. That is, repeatedly expanding the definition of $\mathcal{T}_\pi$, we obtain

$$[\mathcal{T}_\pi^n V](s) = \mathbb{E}\left[\sum_{t=0}^{n-1}\gamma^t(r_t - \tau\,\mathrm{KL}_t) + \gamma^n V(s_n)\,\middle|\, s_0 = s\right] \tag{34}$$

$$[\mathcal{T}_\pi^n Q](s,a) - \tau\,\mathrm{KL}(s) = \mathbb{E}\left[\sum_{t=0}^{n-1}\gamma^t(r_t - \tau\,\mathrm{KL}_t) + \gamma^n(Q(s_n,a_n) - \tau\,\mathrm{KL}_n)\,\middle|\, s_0 = s, a_0 = a\right]. \tag{35}$$

As a sanity check, note that in both equations, the left-hand side and right-hand side correspond to estimates of the total discounted return $\sum_{t=0}^{\infty}\gamma^t(r_t - \tau\,\mathrm{KL}_t)$.

The right-hand side of these backup formulas can be rewritten using "Bellman error" terms δ_t. To rewrite the state-value (V) backup, define

$$\delta_t = (r_t - \tau\,\mathrm{KL}_t) + \gamma V(s_{t+1}) - V(s_t) \tag{36}$$

Then we have

$$[\mathcal{T}_\pi^n V](s) = \mathbb{E}\left[\sum_{t=0}^{n-1}\gamma^t\delta_t + \gamma^n V(s_n)\,\middle|\, s_0 = s\right]. \tag{37}$$

3.4 Boltzmann Backups

We can define another set of backup operators corresponding to the Boltzmann policy, $\pi(a|s) \propto \overline{\pi}(a|s)\exp(Q(s,a)/\tau)$. We define the following Boltzmann backup operator:

$$[\mathcal{T}Q](s,a) = \mathbb{E}_{(r,s')\sim P(r,s' \mid s,a)}\left[r + \gamma\underbrace{\mathbb{E}_{a'\sim\mathcal{G}Q}\left[Q(s,a)\right] - \tau D_{\mathrm{KL}}\left[\mathcal{G}Q \parallel \overline{\pi}\right](s')}_{(*)}\right] \tag{38}$$

$$= \mathbb{E}_{(r,s')\sim P(r,s' \mid s,a)}\left[r + \gamma\underbrace{\tau\log\mathbb{E}_{a'\sim\overline{\pi}}\left[\exp(Q(s',a')/\tau)\right]}_{(**)}\right] \tag{39}$$

where the simplification from $(*)$ to $(**)$ follows from the same calculation that we performed in the bandit setting (Equations (11) and (13)).

The n-step operator $\mathcal{T}_\pi^n$ for Q-functions also simplifies in the case that we are executing the Boltzmann policy. Starting with the equation for $\mathcal{T}_\pi^n Q$ (Equation (35)) and setting $\pi = \pi_Q^{\mathcal{B}}$, and then using Equation (31)

to rewrite the expected Q-function terms in terms of V_Q, we obtain

$$[(\mathcal{T}_{\pi_Q^{\mathcal{B}}})^n Q](s,a) - \tau\,\mathrm{KL}(s) = \mathbb{E}\left[\sum_{t=0}^{n-1} \gamma^t (r_t - \tau\,\mathrm{KL}_t) + \gamma^n (Q(s_n,a_n) - \tau\,\mathrm{KL}_n) \,\middle|\, s_0 = s, a_0 = a\right] \tag{40}$$

$$= \mathbb{E}\left[\sum_{t=0}^{n-1} \gamma^t (r_t - \tau\,\mathrm{KL}_t) + \gamma^n V_Q(s_n) \,\middle|\, s_0 = s, a_0 = a\right]. \tag{41}$$

From now on, let's denote this n-step backup operator by $\mathcal{T}_{\pi^{\mathcal{B}},n}$. (Note $T_{\pi_Q^{\mathcal{B}},n} \neq \mathcal{T}^n Q$, even though $\mathcal{T}_{\pi_Q^{\mathcal{B}},1} Q = \mathcal{T} Q$, because $\mathcal{T}_{\pi_Q^{\mathcal{B}}}$ depends on Q.)

One can similarly define the TD(λ) version of this backup operator

$$[\mathcal{T}_{\pi_Q^{\mathcal{B}},\lambda} Q] = (1-\lambda)(1 + \lambda \mathcal{T}_{\pi_Q^{\mathcal{B}}} + (\lambda \mathcal{T}_{\pi_Q^{\mathcal{B}}})^2 + \dots)\mathcal{T}_{\pi_Q^{\mathcal{B}}} Q. \tag{42}$$

One can straightforwardly verify by comparing terms that it satisfies

$$[\mathcal{T}_{\pi_Q^{\mathcal{B}},\lambda} Q](s,a) = Q(s,a) + \mathbb{E}\left[\sum_{t=0}^{\infty} (\gamma\lambda)^t \delta_t \,\middle|\, s_0 = s, a_0 = a\right],$$
$$\text{where} \quad \delta_t = (r_t - \tau\,\mathrm{KL}_t) + \gamma V_Q(s_{t+1}) - V_Q(s_t). \tag{43}$$

3.5 Soft Q-Learning

The Boltzmann backup operators defined in the preceding section can be used to define practical variants of Q-learning that can be used with nonlinear function approximation. These methods, which optimize the entropy-augmented return, will be called soft Q-learning. Following Mnih et al. [2015], modern implementations of Q-learning, and n-step Q-learning (see Mnih et al. [2016]) update the Q-function incrementally to compute the backup against a fixed target Q-function, which we'll call $\underline{Q}$. In the interval between each target network update, the algorithm is approximately performing the backup operation $Q \leftarrow \mathcal{T}\underline{Q}$ (1-step) or $Q \leftarrow \mathcal{T}_{\pi_{\underline{Q}}^{\mathcal{B}},n}\underline{Q}$ (n-step). To perform this approximate minimization, the algorithms minimize the least squares loss

$$L(Q) = \mathbb{E}_{t,s_t,a_t}\left[\tfrac{1}{2}(Q(s_t,a_t) - y_t)^2\right], \quad \text{where} \tag{44}$$

$$y_t = r_t + \gamma V_{\underline{Q}}(s_{t+1}) \qquad \text{1-step } Q\text{-learning} \tag{45}$$

$$y_t = \tau\,\mathrm{KL}_t + \sum_{d=0}^{n-1} \gamma^d (r_{t+d} - \tau\,\mathrm{KL}_{t+d}) + \gamma^n V_{\underline{Q}}(s_{t+n}) \qquad n\text{-step } Q\text{-learning} \tag{46}$$

$$= \tau\,\mathrm{KL}_t + V_{\underline{Q}}(s_t) + \sum_{d=0}^{n-1} \gamma^d \delta_{t+d}$$
$$\text{where} \quad \delta_t = (r_t - \tau\,\mathrm{KL}_t) + \gamma V_{\underline{Q}}(s_{t+1}) - V_{\underline{Q}}(s_t) \tag{47}$$

In one-step Q-learning (Equation (45)), y_t is an unbiased estimator of $[\mathcal{T}Q](s_t,a_t)$, regardless of what behavior policy was used to collect the data. In n-step Q-learning (Equation (46)), for $n > 1$, y_t is only an unbiased estimator of $[\mathcal{T}_{\pi_{\underline{Q}}^{\mathcal{B}},n}\underline{Q}](s_t,a_t)$ if actions $a_t, a_{t+1}, \dots, a_{t+d-1}$ are sampled using $\pi_{\underline{Q}}^{\mathcal{B}}$.

3.6 Policy Gradients

Entropy regularization is often used in policy gradient algorithms, with gradient estimators of the form

$$\mathbb{E}_{t,s_t,a_t}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \sum_{t' \geq t} r_{t'} - \tau \nabla_\theta D_{\mathrm{KL}}\left[\pi_\theta \parallel \overline{\pi}\right](s_t)\right] \tag{48}$$

(Williams [1992], Mnih et al. [2016]).

However, these are not proper estimators of the entropy-augmented return $\sum_t (r_t - \tau \,\mathrm{KL}_t)$, since they don't account for how actions affect entropy at future timesteps. Intuitively, one can think of the KL terms as a cost for "mental effort". Equation (48) only accounts for the instantaneous effect of actions on mental effort, not delayed effects.

To compute proper gradient estimators, we need to include the entropy terms in the return. We will define the discounted policy gradient in the following two equivalent ways—first, in terms of the empirical return; second, in terms of the value functions V_π and Q_π:

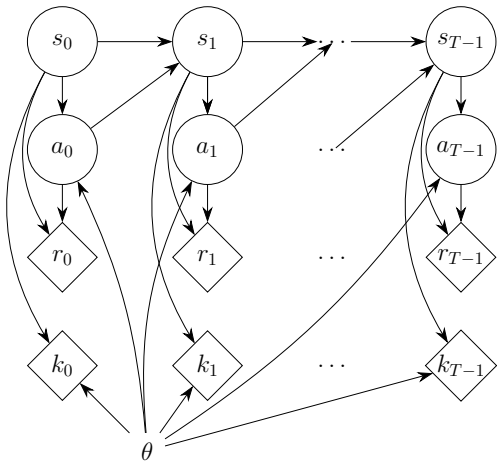
$$g_\gamma(\pi_\theta) = \mathbb{E}\left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\left(r_0 + \sum_{d=1}^{\infty} \gamma^d (r_{t+d} - \tau \,\mathrm{KL}_{t+d}) - \tau \nabla_\theta D_{\mathrm{KL}}\left[\pi_\theta \parallel \overline{\pi}\right](s_t)\right)\right] \tag{49}$$

$$= \mathbb{E}\left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t)(Q_\pi(s_t, a_t) - V_\pi(s_t)) - \tau \nabla_\theta D_{\mathrm{KL}}\left[\pi_\theta \parallel \overline{\pi}\right](s_t)\right] \tag{50}$$

In the special case of a finite-horizon problem—i.e., $r_t = \mathrm{KL}_t = 0$ for all $t \geq T$—the undiscounted ($\gamma = 1$) return is finite, and it is meaningful to compute its gradient. In this case, $g_1(\pi_\theta)$ equals the undiscounted policy gradient:

$$g_1(\pi) = \nabla_\theta \mathbb{E}\left[\sum_{t=0}^{T-1} (r_t - \tau \,\mathrm{KL}_t)\right] \tag{51}$$

This result is obtained directly by considering the stochastic computation graph for the loss (Schulman et al. [2015a]), shown in the figure on the right. The edges from θ to the KL loss terms lead to the $\nabla_\theta D_{\mathrm{KL}}\left[\pi_\theta \parallel \overline{\pi}\right](s_t)$ terms in the gradient; the edges to the stochastic actions a_t lead to the $\nabla_\theta \log \pi_\theta(a_t \mid s_t) \sum_{t=d}^{T-1} (r_{t+d} - \tau \,\mathrm{KL}_{t+d})$ terms in the gradient.



Since $g_1(\pi_\theta)$ computes the gradient of the entropy-regularized return, one interpretation of $g_\gamma(\pi_\theta)$ is that it is an approximation of the undiscounted policy gradient $g_1(\pi_\theta)$, but that it allows for lower-variance gradient estimators by ignoring some long-term dependencies. A different interpretation of $g_\gamma(\pi)$ is that it gives a gradient flow such that $\pi^* = \pi^{\mathcal{B}}_{Q_*}$ is the (possibly unique) fixed point.

As in the standard MDP setting, one can define approximations to g_γ that use a value function to truncate the returns for variance reduction. These approximations can take the form of n-step methods (Mnih et al. [2016]) or TD(λ)-like methods (Schulman et al. [2015b]), though we will focus on n-step returns here. Based on the definition of g_γ above, the natural choice of variance-reduced estimator is

$$\mathbb{E}_{t, s_t, a_t}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \sum_{d=0}^{n-1} \gamma^d \delta_{t+d}\right] \tag{52}$$

where δ_t was defined in Equation (36).

The state-value function V we use in the above formulas should approximate the entropy augmented return $\sum_{t=0}^{\infty} \gamma^t (r_t - \tau \,\mathrm{KL}_t)$. We can fit V iteratively by approximating the n-step backup $V \leftarrow \mathcal{T}^n_\pi V$, by minimizing a squared-error loss

$$L(V) = \mathbb{E}_{t, s_t}\left[\tfrac{1}{2}(V(s_t) - y_t)^2\right], \tag{53}$$

$$\text{where} \quad y_t = \sum_{d=0}^{n-1} \gamma^d r_{t+d} + \gamma^d V(s_{t+d}) = V(s_t) + \sum_{d=0}^{n-1} \gamma^d \delta_{t+d}. \tag{54}$$

4 Soft Q-learning Gradient Equals Policy Gradient

This section shows that the gradient of the squared-error loss from soft Q-learning (Section 3.5) equals the policy gradient (in the family of policy gradients described in Section 3.6) plus the gradient of a squared-error term for fitting the value function. We will not make any assumption about the parameterization of the Q-function, but we define V_θ and π_θ as the following functions of the parameterized Q-function Q_θ:

$$V_\theta(s) := \tau \log \mathbb{E}_a \left[\exp(Q_\theta(s,a)/\tau)\right] \tag{55}$$

$$\pi_\theta(a \mid s) := \overline{\pi}(a \mid s) \exp((Q_\theta(s,a) - V_\theta(s))/\tau) \tag{56}$$

Here, π_θ is the Boltzmann policy for Q_θ, and V_θ is the normalizing factor we described above. From these definitions, it follows that the Q-function can be written as

$$Q_\theta(s,a) = V_\theta(s) + \tau \log \tfrac{\pi_\theta(a \mid s)}{\overline{\pi}(a \mid s)} \tag{57}$$

We will substitute this expression into the squared-error loss function. First, for convenience, let us define $\Delta_t = \sum_{d=0}^{n-1} \gamma^d \delta_{t+d}$.

Now, let's consider the gradient of the n-step soft Q-learning objective:

$$\nabla_\theta \mathbb{E}_{t,s_t,a_t\sim\pi}\left[\tfrac{1}{2}\|Q_\theta(s_t,a_t) - y_t\|^2\right]\Big|_{\pi=\pi_\theta} \tag{58}$$

swap gradient and expectation, treating state-action distribution as fixed:

$$= \mathbb{E}_{t,s_t,a_t\sim\pi}\left[\nabla_\theta Q_\theta(s_t,a_t)(Q_\theta(s_t,a_t) - y_t)\right]\Big|_{\pi=\pi_\theta} \tag{59}$$

replace Q_θ using Equation (57), and replace Q-value backup y_t by Equation (46):

$$= \mathbb{E}_{t,s_t,a_t\sim\pi}\Big[\nabla_\theta Q_\theta(s_t,a_t)(\tau \log \tfrac{\pi(a_t \mid s_t)}{\overline{\pi}(a_t \mid s_t)} + V_\theta(s_t) - (V_\theta(s_t) + \tau D_{\mathrm{KL}}\left[\pi_\theta \parallel \overline{\pi}\right](s_t) + \Delta_t))\Big]\Big|_{\pi=\pi_\theta} \tag{60}$$

cancel out $V_\theta(s_t)$:

$$= \mathbb{E}_{t,s_t,a_t\sim\pi}\Big[\nabla_\theta Q_\theta(s_t,a_t)(\tau \log \tfrac{\pi(a_t \mid s_t)}{\overline{\pi}(a_t \mid s_t)} - \tau D_{\mathrm{KL}}\left[\pi_\theta \parallel \overline{\pi}\right](s_t) - \Delta_t)\Big]\Big|_{\pi=\pi_\theta} \tag{61}$$

replace the other Q_θ by Equation (57):

$$= E_{t,s_t,a_t\sim\pi}\Big[(\tau\nabla_\theta \log \pi_\theta(a_t \mid s_t) + \nabla_\theta V_\theta(s_t)) \cdot (\tau \log \tfrac{\pi(a_t \mid s_t)}{\overline{\pi}(a_t \mid s_t)} - \tau D_{\mathrm{KL}}\left[\pi_\theta \parallel \overline{\pi}\right](s_t) - \Delta_t)\Big]\Big|_{\pi=\pi_\theta} \tag{62}$$

expand out terms:

$$= E_{t,s_t,a_t\sim\pi}\Big[\tau^2\nabla_\theta \log \pi_\theta(a_t \mid s_t) \log \tfrac{\pi_\theta(a_t \mid s_t)}{\overline{\pi}(a_t \mid s_t)} - \tau^2 \underbrace{\nabla_\theta \log \pi_\theta(a_t \mid s_t) D_{\mathrm{KL}}\left[\pi_\theta \parallel \overline{\pi}\right](s_t)}_{(*)}$$

$$- \tau\nabla_\theta \log \pi_\theta(a_t \mid s_t)\Delta_t + \underbrace{\tau\nabla_\theta V_\theta(s_t) \log \tfrac{\pi_\theta(a_t \mid s_t)}{\overline{\pi}(a_t \mid s_t)} - \tau\nabla_\theta V_\theta(s_t) D_{\mathrm{KL}}\left[\pi_\theta \parallel \overline{\pi}\right](s_t)}_{(**)} - \nabla_\theta V_\theta(s_t)\Delta_t\Big]\Big|_{\pi=\pi_\theta} \tag{63}$$

$(*)$ vanishes because $\mathbb{E}_{a\sim\pi_\theta(\cdot \mid s_t)}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot const\right] = 0$

$(**)$ vanishes because $\mathbb{E}_{a\sim\pi_\theta(\cdot \mid s_t)}\left[\tfrac{\pi_\theta(a_t \mid s_t)}{\overline{\pi}(a_t \mid s_t)}\right] = D_{\mathrm{KL}}\left[\pi_\theta \parallel \overline{\pi}\right](s_t)$

$$= E_{t,s_t,a_t\sim\pi}\Big[\tau^2\nabla_\theta D_{\mathrm{KL}}\left[\pi_\theta \parallel \overline{\pi}\right](s_t) + 0 - \tau\nabla_\theta \log \pi_\theta(a_t \mid s_t)\Delta_t + 0 - \nabla_\theta V_\theta(s_t)\Delta_t\Big]\Big|_{\pi=\pi_\theta} \tag{64}$$

rearrange terms:

$$= \mathbb{E}_{t,s_t,a_t\sim\pi}\Big[\underbrace{-\tau\nabla_\theta \log \pi_\theta(a_t \mid s_t)\Delta_t + \tau^2\nabla_\theta D_{\mathrm{KL}}\left[\pi_\theta \parallel \overline{\pi}\right]](s_t)}_{\text{policy grad}} + \underbrace{\nabla_\theta \tfrac{1}{2}\left\|V_\theta(s_t) - \hat{V}_t\right\|^2}_{\text{value function grad}}\Big]\Big|_{\pi=\pi_\theta} \tag{65}$$

Note that the equivalent policy gradient method multiplies the policy gradient by a factor of τ, relative to the value function error. Effectively, the value function error has a coefficient of τ^{-1}, which is larger than what is typically used in practice (Mnih et al. [2016]). We will analyze this choice of coefficient in the experiments.

5 Soft Q-learning and Natural Policy Gradients

The previous section gave a first-order view on the equivalence between policy gradients and soft Q-learning; this section gives a second-order, coordinate-free view. As previous work has pointed out, the natural gradient is the solution to a regression problem; here we will explore the relation between that problem and the nonlinear regression in soft Q-learning.

The natural gradient is defined as $F^{-1}g$, where F is the average Fisher information matrix, $F = \mathbb{E}_{s,a\sim\pi}\left[(\nabla_\theta \log \pi_\theta(a \mid s))^T(\nabla_\theta \log \pi_\theta(a \mid s))\right]$, and g is the policy gradient estimate $g \propto \mathbb{E}\left[\nabla_\theta \log \pi_\theta(a \mid s)\Delta\right]$, where Δ is an estimate of the advantage function. As pointed out by Kakade [2002], the natural gradient step can be computed as the solution to a least squares problem. Given timesteps $t = 1, 2, \ldots, T$, define $\boldsymbol{\psi}_t = \nabla_\theta \log \pi_\theta(a_t \mid s_t)$. Define $\boldsymbol{\Psi}$ as the matrix whose t^{th} row is $\boldsymbol{\psi}_t$, let $\boldsymbol{\Delta}$ denote the vector whose t^{th} element is the advantage estimate Δ_t, and let ϵ denote a scalar stepsize parameter. Consider the least squares problem

$$\min_{\mathbf{w}} \tfrac{1}{2}\|\boldsymbol{\Psi}\mathbf{w} - \epsilon\boldsymbol{\Delta}\|^2 \tag{66}$$

The least-squares solution is $\mathbf{w} = \epsilon(\boldsymbol{\Psi}^T\boldsymbol{\Psi})^{-1}\boldsymbol{\Psi}^T\boldsymbol{\Delta}$. Note that $\mathbb{E}\left[\boldsymbol{\Psi}^T\boldsymbol{\Psi}\right]$ is the Fisher information matrix F, and $\mathbb{E}\left[\boldsymbol{\Psi}^T\boldsymbol{\Delta}\right]$ is the policy gradient g, so $\mathbf{w}$ is the estimated natural gradient.

Now let us interpret the least-squares problem in Equation (66). $\boldsymbol{\Psi}\mathbf{w}$ is the vector whose t^{th} row is $\nabla_\theta \log \pi_\theta(a \mid s) \cdot \mathbf{w}$. According to the definition of the gradient, if we perform a parameter update with $\theta - \theta_{\text{old}} = \epsilon\mathbf{w}$, the change in $\log \pi_\theta(a \mid s)$ is as follows, to first order in ϵ:

$$\log \pi_\theta(a \mid s) - \log \pi_{\theta_{\text{old}}}(a \mid s) \approx \nabla_\theta \log \pi_\theta(a \mid s) \cdot \epsilon\mathbf{w} = \epsilon\boldsymbol{\psi} \cdot \mathbf{w} \tag{67}$$

Thus, we can interpret the least squares problem (Equation (66)) as solving

$$\min_\theta \sum_{t=1}^{T} \tfrac{1}{2}(\log \pi_\theta(a_t \mid s_t) - \log \pi_{\theta_{\text{old}}}(a_t \mid s_t) - \epsilon\Delta_t)^2 \tag{68}$$

That is, we are adjusting each log-probility $\log \pi_{\theta_{\text{old}}}(a_t \mid s_t)$ by the advantage function Δ_t, scaled by ϵ.

In entropy-regularized reinforcement learning, we have an additional term for the gradient of the KL-divergence:

$$g \propto \mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\Delta_t - \tau\nabla_\theta \operatorname{KL}[\pi_\theta, \overline{\pi}](s_t)\right] \tag{69}$$

$$= \mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\Big(\Delta_t - \tau\Big[\log\Big(\tfrac{\pi_\theta(a_t \mid s_t)}{\overline{\pi}(a_t \mid s_t)}\Big) - \operatorname{KL}[\pi_\theta, \overline{\pi}](s_t)\Big]\Big)\right] \tag{70}$$

where the second line used the formula for the KL-divergence (Equation (21)) and the identity that $\mathbb{E}_{a_t\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot \text{const}\right] = 0$ (where the KL term is the constant.) In this case, the corresponding least squares problem (to compute $F^{-1}g$) is

$$\min_\theta \sum_{t=1}^{T} \tfrac{1}{2}\Big(\log \pi_\theta(a_t \mid s_t) - \log \pi_{\theta_{\text{old}}}(a_t \mid s_t) - \epsilon\Big(\Delta_t - \tau\Big[\log\Big(\tfrac{\pi_\theta(a_t \mid s_t)}{\overline{\pi}(a_t \mid s_t)}\Big) - \operatorname{KL}[\pi_{\theta_{\text{old}}}, \overline{\pi}](s_t)\Big]\Big)\Big)^2. \tag{71}$$

Now let's consider Q-learning. Let's assume that the value function is unchanged by optimization, so $V_\theta = V_{\theta_{\text{old}}}$. (Otherwise, the equivalence will not hold, since the value function will try to explain the measured advantage Δ, shrinking the advantage update.)

$$\tfrac{1}{2}(Q_\theta(s_t, a_t) - y_t)^2 = \tfrac{1}{2}\Big(\Big(V_\theta(s_t, a_t) + \tau\log\Big(\tfrac{\pi_\theta(a_t \mid s_t)}{\overline{\pi}(a_t \mid s_t)}\Big)\Big) - (V_{\theta_{\text{old}}}(s_t) + \tau\operatorname{KL}[\pi_{\theta_{\text{old}}}, \overline{\pi}](s_t) + \Delta_t)\Big)^2 \tag{72}$$

$$= \tfrac{1}{2}\Big(\tau\log\Big(\tfrac{\pi_\theta(a_t \mid s_t)}{\overline{\pi}(a_t \mid s_t)}\Big) - (\Delta_t + \tau\operatorname{KL}[\pi_{\theta_{\text{old}}}, \overline{\pi}](s_t))\Big)^2 \tag{73}$$

Evidently, we are regressing $\log \pi_\theta(a_t \mid s_t)$ towards $\log \pi_{\theta_{\text{old}}}(a_t \mid s_t) + \Delta_t/\tau + \operatorname{KL}[\pi_{\theta_{\text{old}}}, \overline{\pi}](s_t)$. This loss is *not* equivalent to the natural policy gradient loss that we obtained above.

We can recover the natural policy gradient by instead solving a *damped* version of the Q-function regression problem. Define $\hat{Q}_t^\epsilon = (1-\epsilon)Q_{\theta_{\text{old}}}(s_t,a_t) + \epsilon\hat{Q}_t$, i.e., we are interpolating between the old value and the backed-up value.

$$\hat{Q}_t^\epsilon = (1-\epsilon)Q_{\theta_{\text{old}}}(s_t,a_t) + \epsilon\hat{Q}_t = Q_{\theta_{\text{old}}}(s_t,a_t) + \epsilon(\hat{Q}_t - Q_{\theta_{\text{old}}}(s_t,a_t)) \tag{74}$$

$$\hat{Q}_t - Q_{\theta_{\text{old}}}(s_t,a_t) = (V_\theta(s_t) + \tau\,\mathrm{KL}[\pi_{\theta_{\text{old}}},\overline{\pi}](s_t) + \Delta_t) - \left(V_{\theta_{\text{old}}}(s_t) + \tau\log\left(\frac{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}{\overline{\pi}(a_t \mid s_t)}\right)\right) \tag{75}$$

$$= \Delta_t + \tau\left[\mathrm{KL}[\pi_{\theta_{\text{old}}},\overline{\pi}](s_t) - \log\left(\frac{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}{\overline{\pi}(a_t \mid s_t)}\right)\right] \tag{76}$$

$$Q_\theta(s_t,a_t) - \hat{Q}_t^\epsilon = Q_\theta(s_t,a_t) - \left(Q_{\theta_{\text{old}}}(s_t,a_t) + \epsilon\left(\hat{Q}_t - Q_{\theta_{\text{old}}}(s_t,a_t)\right)\right) \tag{77}$$

$$= V_\theta(s_t) + \log\left(\frac{\pi_\theta(a_t \mid s_t)}{\overline{\pi}(a_t \mid s_t)}\right) - \left\{V_{\theta_{\text{old}}}(s_t) + \log\left(\frac{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}{\overline{\pi}(a_t \mid s_t)}\right) + \epsilon\left(\Delta + \tau\left[\mathrm{KL}[\pi_{\theta_{\text{old}}},\overline{\pi}](s_t) - \log\left(\frac{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}{\overline{\pi}(a_t \mid s_t)}\right)\right]\right)\right\}$$

$$= \log\pi_\theta(a_t \mid s_t) - \log\pi_{\theta_{\text{old}}}(a_t \mid s_t) - \epsilon\left(\Delta_t - \tau\left[\log\left(\frac{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}{\overline{\pi}(a_t \mid s_t)}\right) - \mathrm{KL}[\pi_{\theta_{\text{old}}},\overline{\pi}](s_t)\right]\right) \tag{78}$$

which exactly matches the expression in the least squares problem in Equation (71), corresponding to entropy-regularized natural policy gradient. Hence, the "damped" Q-learning update corresponds to a natural gradient step.

6 Experiments

To complement our theoretical analyses, we designed experiments to study the following questions:

1. Though one-step entropy bonuses are used in PG methods for neural network policies (Williams [1992], Mnih et al. [2016]), how do the entropy-regularized RL versions of policy gradients and Q-learning described in Section 3 perform on challenging RL benchmark problems? How does the "proper" entropy-regularized policy gradient method (with entropy in the returns) compare to the naive one (with one-step entropy bonus)? (Section 6.1)
2. How do the entropy-regularized versions of Q-learning (with logsumexp) compare to the standard DQN of Mnih et al. [2015]? (Section 6.2)
3. The equivalence between PG and soft Q-learning is established *in expectation*, however, the actual gradient estimators are slightly different due to sampling. Furthermore, soft Q-learning is equivalent to PG with a particular penalty coefficient on the value function error. Does the equivalence hold under practical conditions? (Section 6.3)

6.1 A2C on Atari: Naive vs Proper Entropy Bonuses

Here we investigated whether there is an empirical effect of including entropy terms when computing returns, as described in Section 3. In this section, we compare the naive and proper policy gradient estimators:

$$\text{naive / 1-step:} \qquad \nabla\log\pi_\theta(a_t \mid s_t)\left(\sum_{d=0}^{n-1}\gamma^d r_{t+d} - V(s_t)\right) - \tau\nabla_\theta D_{\mathrm{KL}}\left[\pi_\theta \parallel \overline{\pi}\right](s_t) \tag{79}$$

$$\text{proper:} \qquad \nabla\log\pi_\theta(a_t \mid s_t)\left(\sum_{d=0}^{n-1}\gamma^d(r_{t+d} - \tau D_{\mathrm{KL}}\left[\pi_\theta \parallel \overline{\pi}\right](s_{t+d})) - V(s_t)\right) - \tau\nabla_\theta D_{\mathrm{KL}}\left[\pi_\theta \parallel \overline{\pi}\right](s_t) \tag{80}$$

In the experiments on Atari, we take $\overline{\pi}$ to be the uniform distribution, which gives a standard entropy bonus up to a constant.

We start with a well-tuned (synchronous, deterministic) version of A3C (Mnih et al. [2016]), henceforth called A2C (advantage actor critic), to optimize the entropy-regularized return. We use the parameter $\tau = 0.01$ and train for 320 million frames. We did not tune any hyperparameters for the "proper" algorithm—we used the same hyperparameters that had been tuned for the "naive" algorithm.

As shown in Figure 1, the "proper" version yields performance that is the same or possibly greater than the "naive" version. Hence, besides being attractive theoretically, the entropy-regularized formulation could lead to practical performance gains.

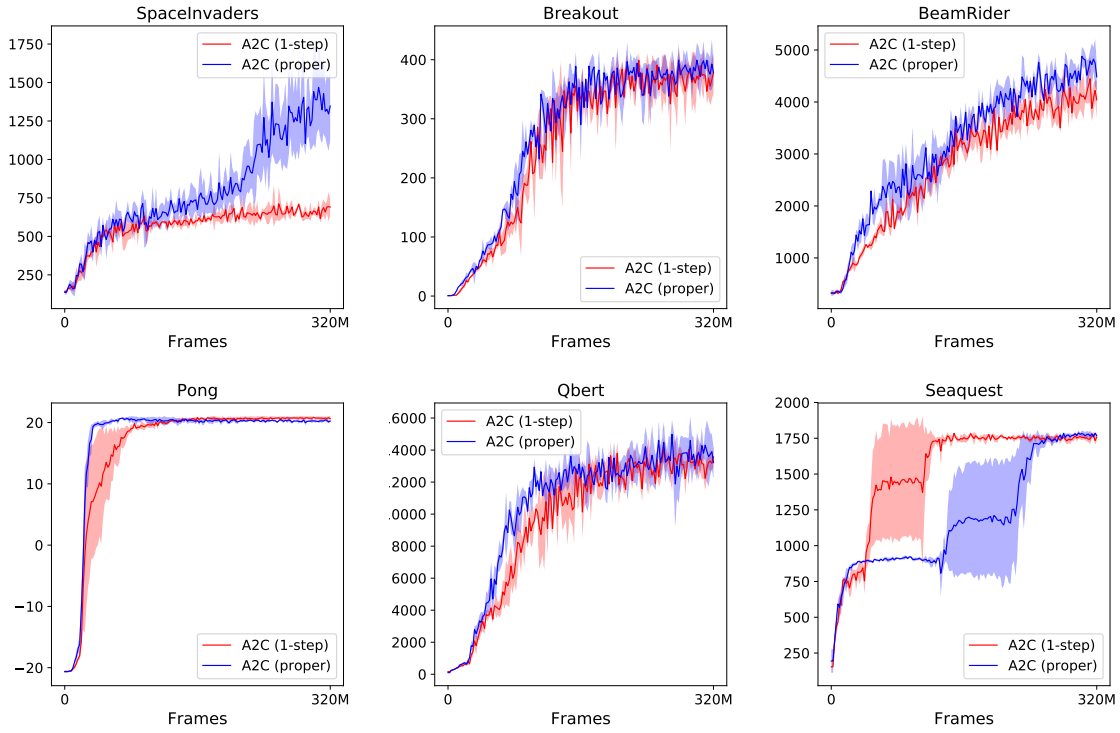


Figure 1: Atari performance with different RL objectives. EntRL is A2C modified to optimize for return augmented with entropy (instead of KL penalty). Solid lines are average evaluation return over 3 random seeds and shaded area is one standard deviation.

6.2 DQN on Atari: Standard vs Soft

Here we investigated whether soft Q-learning (which optimizes the entropy-augmented return) performs differently from standard "hard" Q-learning on Atari. We made a one-line change to a DQN implementation:

$$y_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a') \qquad \text{Standard} \tag{81}$$

$$y_t = r_t + \gamma \log \sum_{a'} \exp(Q(s_{t+1}, a')/\tau) - \log|\mathcal{A}| \qquad \text{"Soft": KL penalty} \tag{82}$$

$$y_t = r_t + \gamma \log \sum_{a'} \exp(Q(s_{t+1}, a')/\tau) \qquad \text{"Soft": Entropy bonus} \tag{83}$$

The difference between the entropy bonus and KL penalty (against uniform) is simply a constant, however, this constant made a big difference in the experiments, since a positive constant added to the reward encourages longer episodes. Note that we use the same epsilon-greedy exploration in all conditions; the only difference is the backup equation used for computing y_t and defining the loss function.

The results of two runs on each game are shown in Figure 2. The entropy-bonus version with $\tau = 0.1$ seems to perform a bit better than standard DQN, however, the KL-bonus version performs worse, so the benefit may be due to the effect of adding a small constant to the reward. We have also shown the results for 5-step Q-learning, where the algorithm is otherwise the same. The performance is better on Pong and Q-bert but worse on other games—this is the same pattern of performance found with n-step policy gradients. (E.g., see the A2C results in the preceding section.)

6.3 Entropy Regularized PG vs Online Q-Learning on Atari

Next we investigate if the equivalence between soft Q-learning and PG is relevant in practice—we showed above that the gradients are the same in expectation, but their variance might be different, causing different

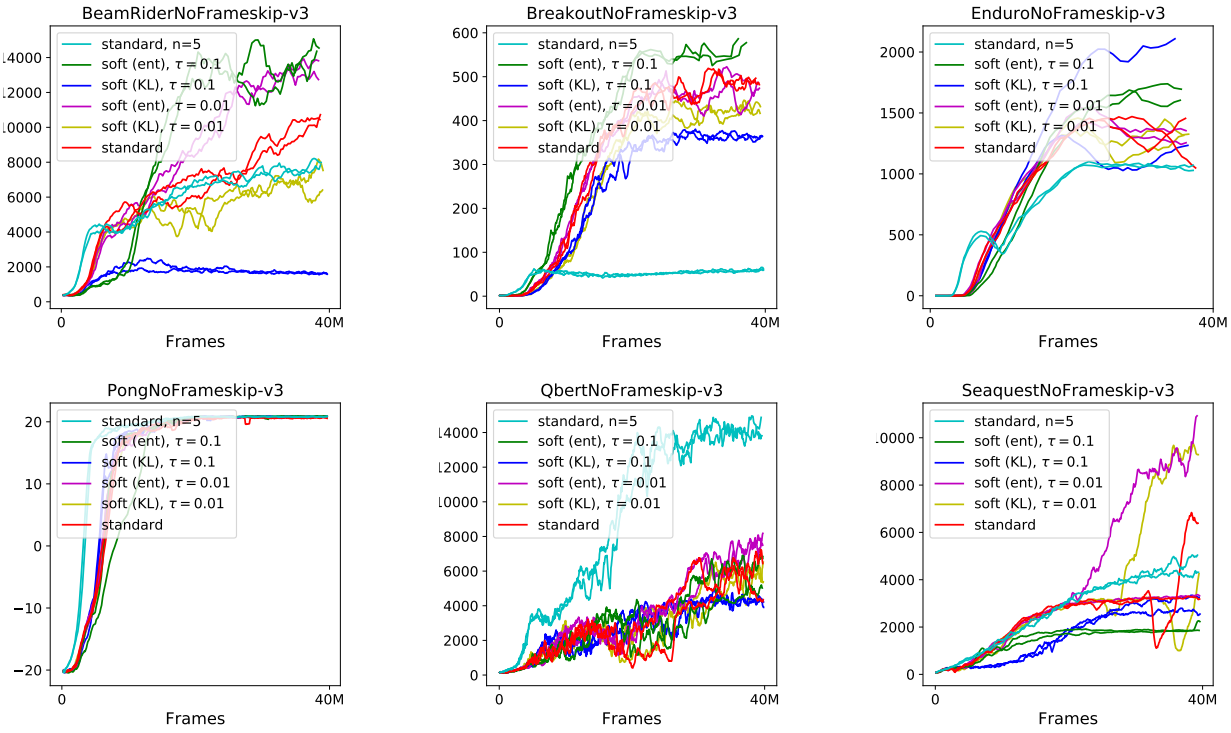


Figure 2: Different variants of soft Q-learning and standard Q-learning, applied to Atari games. Note that 4 frames = 1 timestep.

learning dynamics. For these experiments, we modified the gradient update rule used in A2C while making no changes to any algorithmic component, i.e. parallel rollouts, updating parameters every 5 steps, etc. The Q-function was represented as: $Q_\theta(s,a) = V_\theta(s) + \tau \log \pi_\theta(a \mid s)$, which can be seen as a form of dueling architecture with $\tau \log \pi_\theta(a \mid s)$ being the "advantage stream" (Wang et al. [2015]). V_θ, π_θ are parametrized as the same neural network as A2C, where convolutional layers and the first fully connected layer are shared. $\pi_\theta(a \mid s)$ is used as behavior policy.

A2C can be seen as optimizing a combination of a policy surrogate loss and a value function loss, weighted by hyperparameter c:

$$L_{\text{policy}} = -\log \pi_\theta(a_t \mid s_t)\Delta_t + \tau D_{\text{KL}}\left[\pi_\theta \parallel \overline{\pi}\right](s_t) \tag{84}$$

$$L_{\text{value}} = \frac{1}{2}\left\| V_\theta(s_t) - \hat{V}_t \right\|^2 \tag{85}$$

$$L_{\text{a2c}} = L_{\text{policy}} + cL_{\text{value}} \tag{86}$$

In normal A2C, we have found $c = 0.5$ to be a robust setting that works across multiple environments. On the other hand, our theory suggests that if we use this Q-function parametrization, soft Q-learning has the same expected gradient as entropy-regularized A2C with a specific weighting $c = \frac{1}{\tau}$. Hence, for the usual entropy bonus coefficient setting $\tau = 0.01$, soft Q-learning is implicitly weighting value function loss a lot more than usual A2C setup ($c = 100$ versus $c = 0.5$). We have found that such emphasis on value function ($c = 100$) results in unstable learning for both soft Q-learning and entropy-regularized A2C. Therefore, to make Q-learning exactly match known good hyperparameters used in A2C, we scale gradients that go into advantage stream by $\frac{1}{\gamma}$ and scale gradients that go into value function stream by $c = 0.5$.

With the same default A2C hyperparameters, learning curves of PG and QL are almost identical in most games (Figure 3), which indicates that the learning dynamics of both update rules are essentially the same even when the gradients are approximated with a small number of samples. Notably, the Q-learning method here demonstrates stable learning without the use of target network or ϵ schedule.

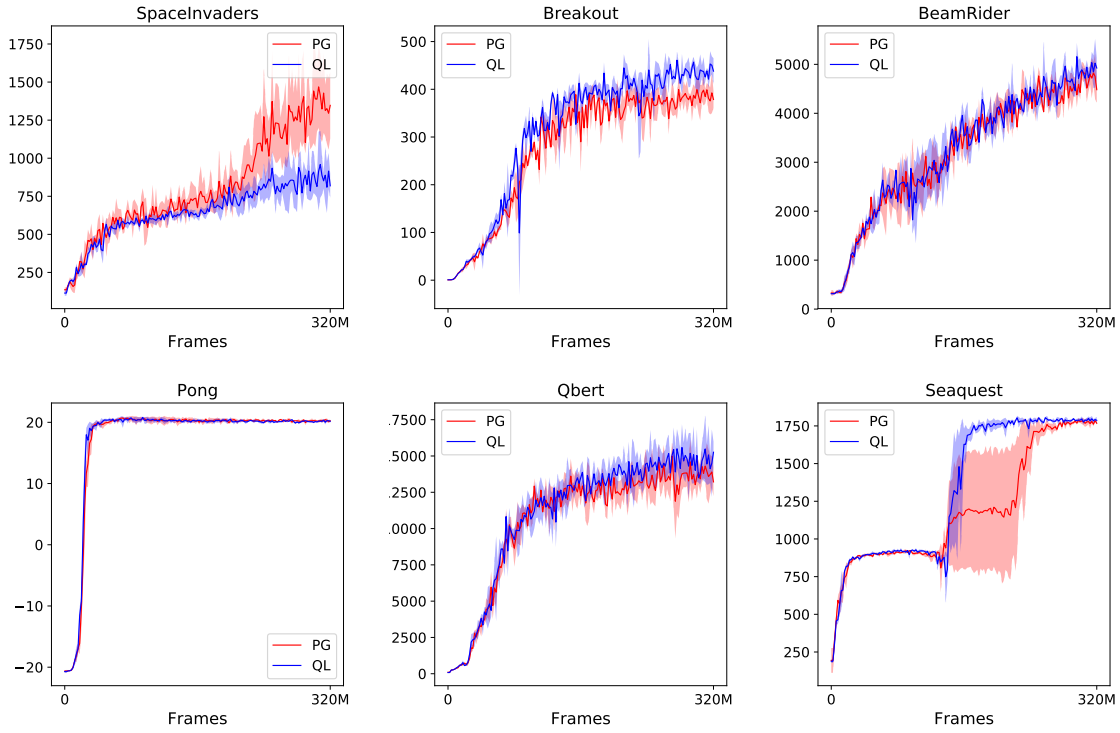


Figure 3: Atari performance with policy gradient vs Q-learning update rules. Solid lines are average evaluation return over 3 random seeds and shaded area is one standard deviation.

7 Related Work

Three recent papers have drawn the connection between policy-based methods and value-based methods, which becomes close with entropy regularization.

- O'Donoghue et al. [2016] begin with a similar motivation as the current paper: that a possible explanation for Q-learning and SARSA is that their updates are similar to policy gradient updates. They decompose the Q-function into a policy part and a value part, inspired by dueling Q-networks (Wang et al. [2015]):

$$Q(s,a) = V(s) + \tau(\log \pi(a \mid s) + \tau S[\pi(\cdot \mid s)]) \tag{87}$$

  This form is chosen so that the term multiplying τ has expectation zero under π, which is a property that the true advantage function satisfies: $\mathbb{E}_\pi [A_\pi] = 0$. Note that our work omits that S term, because it is most natural to define the Q-function to not include the first entropy term. The authors show that taking the gradient of the Bellman error of the above Q-function leads to a result similar to the policy gradient. They then propose an algorithm called PGQ that mixes together the updates from different prior algorithms.

- Nachum et al. [2017] also discuss the entropy-regularized reinforcement learning setting, and develop an off-policy method that applies in this setting. Their argument (modified to use our notation and KL penalty instead of entropy bonus) is as follows. The advantage function $A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$ lets us define a multi-step consistency equation, which holds even if the actions were sampled from a different (suboptimal) policy. In the setting of deterministic dynamics, $Q_\pi(s_t, a_t) = r_t + \gamma V_\pi(s_{t+1})$, hence

$$\sum_{t=0}^{n-1} \gamma^t A_\pi(s_t, a_t) = \sum_{t=0}^{n-1} \gamma^t (r_t + \gamma V_\pi(s_{t+1}) - V_\pi(s_t)) = \sum_{t=0}^{n-1} \gamma^t r_t + \gamma^n V_\pi(s_n) - V_\pi(s_0) \tag{88}$$

If π is the optimal policy (for the discounted, entropy-augmented return), then it is the Boltzmann policy for Q_π, thus

$$\tau(\log \pi(a \mid s) - \log \overline{\pi}(a \mid s)) = A_{Q_\pi}(s, a) \tag{89}$$

This expression for the advantage can be substituted into Equation (88), giving the consistency equation

$$\sum_{t=0}^{n-1} \gamma^t \tau(\log \pi(s_t, a_t) - \log \overline{\pi}(s_t, a_t)) = \sum_{t=0}^{n-1} \gamma^t r_t + \gamma^n V_\pi(s_n) - V_\pi(s_0), \tag{90}$$

which holds when π is optimal. The authors define a squared error objective formed from by taking LHS - RHS in Equation (90), and jointly minimize it with respect to the parameters of π and V. The resulting algorithm is a kind of Bellman residual minimization—it optimizes with respect to the future target values, rather than treating them as fixed Scherrer [2010].

- Haarnoja et al. [2017] work in the same setting of soft Q-learning as the current paper, and they are concerned with tasks with high-dimensional action spaces, where we would like to learn stochastic policies that are multi-modal, and we would like to use Q-functions for which there is no closed-form way of sampling from the Boltzmann distribution $\pi(a \mid s) \propto \overline{\pi}(a \mid s) \exp(Q(s, a)/\tau)$. Hence, they use a method called Stein Variational Gradient Descent to derive a procedure that jointly updates the Q-function and a policy π, which approximately samples from the Boltzmann distribution—this resembles variational inference, where one makes use of an approximate posterior distribution.

8 Conclusion

We study the connection between two of the leading families of RL algorithms used with deep neural networks. In a framework of entropy-regularized RL we show that soft Q-learning is equivalent to a policy gradient method (with value function fitting) in terms of expected gradients (first-order view). In addition, we also analyze how a damped Q-learning method can be interpreted as implementing natural policy gradient (second-order view). Empirically, we show that the entropy regularized formulation considered in our theoretical analysis works in practice on the Atari RL benchmark, and that the equivalence holds in a practically relevant regime.

9 Acknowledgements

We would like to thank Matthieu Geist for pointing out an error in the first version of this manuscript, Chao Gao for pointing out several errors in the second version, and colleagues at OpenAI for insightful discussions.

References

Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. *arXiv preprint arXiv:1512.08562*, 2015.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*, 2017.

Sham Kakade. A natural policy gradient. *Advances in neural information processing systems*, 2:1531–1538, 2002.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783*, 2016.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. *arXiv preprint arXiv:1702.08892*, 2017.

Brendan O'Donoghue, Remi Munos, Koray Kavukcuoglu, and Volodymyr Mnih. Pgq: Combining policy gradient and q-learning. *arXiv preprint arXiv:1611.01626*, 2016.

Bruno Scherrer. Should one compute the temporal difference fix point or minimize the bellman residual? the unified oblique projection view. *arXiv preprint arXiv:1011.4362*, 2010.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems*, pages 3528–3536, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

Ziyu Wang, Nando de Freitas, and Marc Lanctot. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

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

Brian D Ziebart. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. 2010.