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 $P(r \,|\, a)$, 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

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

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

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

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 $\pi^{\mathcal{B}}_{q_\theta}$:

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

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

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

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

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

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:

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

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

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

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

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

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

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

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 $\sum_{t=0}^{\infty} \gamma^t (r_t - \tau \, \mathrm{KL}_t)$ 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: $\mathrm{KL}_t = D_{\mathrm{KL}} \left[\pi(\cdot \mid s_t) \parallel \overline{\pi}(\cdot \mid s_t)\right]$. We will sometimes use the notation $\mathrm{KL}(s) = D_{\mathrm{KL}} \left[\pi \parallel \overline{\pi}\right](s) = D_{\mathrm{KL}} \left[\pi(\cdot \mid s) \parallel \overline{\pi}(\cdot \mid s)\right]$. 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:

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

and we shall define the Q-function as

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

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:

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

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

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

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

$$\begin{aligned}
[(\mathcal{T}_{\pi^{\mathcal{B}}_{Q}})^n Q](s,a) - \tau \operatorname{KL}(s) &= \mathbb{E}\left[\sum_{t=0}^{n-1} \gamma^t (r_t - \tau \operatorname{KL}_t) + \gamma^n (Q(s_n, a_n) - \tau \operatorname{KL}_n) \,\middle|\, s_0 = s, a_0 = a \right] && (40)\\
&= \mathbb{E}\left[\sum_{t=0}^{n-1} \gamma^t (r_t - \tau \operatorname{KL}_t) + \gamma^n V_Q \,\middle|\, s_0 = s, a_0 = a \right]. && (41)
\end{aligned}$$

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

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

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

One can straightforwardly verify by comparing terms that it satisfies

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

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. These methods, which optimize the entropy-augmented, 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 $Q \leftarrow \mathcal{T}_{\pi^{\mathcal{B}}_{\underline{Q}},n} \underline{Q}$ (n-step). To perform this approximate minimization, the algorithms minimize the least squares loss

$$\begin{aligned}
L(Q) &= \mathbb{E}_{t,s_t,a_t}\left[\tfrac{1}{2}(Q(s_t,a_t) - y_t)^2 \right], \quad \text{where} && && (44)\\
y_t &= r_t + \gamma V_{\underline{Q}}(s_{t+1}) && \text{1-step } Q\text{-learning} && (45)\\
y_t &= \tau \operatorname{KL}_t + \sum_{d=0}^{n-1} \gamma^d (r_{t+d} - \tau \operatorname{KL}_{t+d}) + \gamma^n V_{\underline{Q}}(s_{t+n}) && n\text{-step } Q\text{-learning} && (46)\\
&= \tau \operatorname{KL}_t + V_{\underline{Q}}(s_t) + \sum_{d=0}^{n-1} \gamma^d \delta_{t+d} && &&\\
&\quad \text{where} \quad \delta_t = (r_t - \tau \operatorname{KL}_t) + \gamma V_{\underline{Q}}(s_{t+1}) - V_{\underline{Q}}(s_t) && && (47)
\end{aligned}$$

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 $[\mathcal{T}_{\pi^{\mathcal{B}}_{\underline{Q}},n} \underline{Q}](s_t,a_t)$ if actions $a_t, a_{t+1}, \dots, a_{t+d-1}$ are sampled using $\pi^{\mathcal{B}}_{\underline{Q}}$.

3.6 Policy Gradients

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

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

(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:

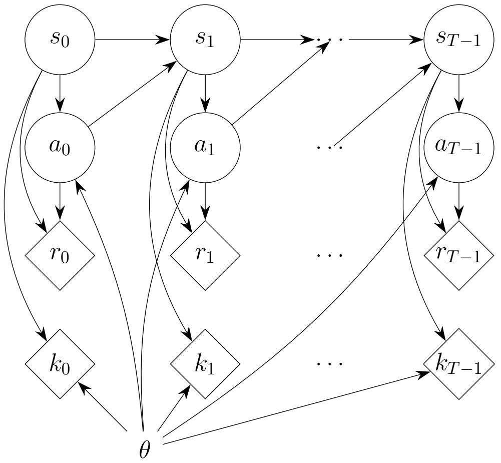
$$g_\gamma(\pi_\theta) = \mathbb{E}\left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \left(r_0 + \sum_{d=1}^{\infty} \gamma^d (r_{t+d} - \tau \,\mathrm{KL}_{t+d}) - \tau \nabla_\theta D_{\mathrm{KL}}\left[\pi_\theta \parallel \overline{\pi}\right](s_t) \right)\right] \tag{49}$$

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

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 $V \leftarrow \mathcal{T}_\pi^n V$, 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):

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

cancel out $V_\theta(s_t)$:

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

replace the other Q_θ by Equation (57):

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

expand out terms:

$$\begin{aligned}&= E_{t,s_t,a_t\sim\pi}\Big[\tau^2\nabla_\theta \log \pi_\theta(a_t \mid s_t) \log \tfrac{\pi_\theta(a_t \mid s_t)}{\overline{\pi}(a_t \mid s_t)} - \tau^2 \underbrace{\nabla_\theta \log \pi_\theta(a_t \mid s_t) D_{\mathrm{KL}}\left[\pi_\theta \parallel \overline{\pi}\right](s_t)}_{(*)} \\ &- \tau\nabla_\theta \log \pi_\theta(a_t \mid s_t)\Delta_t + \underbrace{\tau\nabla_\theta V_\theta(s_t) \log \tfrac{\pi_\theta(a_t \mid s_t)}{\overline{\pi}(a_t \mid s_t)} - \tau\nabla_\theta V_\theta(s_t) D_{\mathrm{KL}}\left[\pi_\theta \parallel \overline{\pi}\right](s_t)}_{(**)} - \nabla_\theta V_\theta(s_t)\Delta_t\Big]\Big|_{\pi=\pi_\theta}\end{aligned} \tag{63}$$

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

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

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

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

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:

$$\begin{aligned} g &\propto \mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\Delta_t - \tau\nabla_\theta \operatorname{KL}[\pi_\theta, \overline{\pi}](s_t)\right] && (69)\\ &= \mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\left(\Delta_t - \tau\left[\log\left(\tfrac{\pi_\theta(a_t \mid s_t)}{\overline{\pi}(a_t \mid s_t)}\right) - \operatorname{KL}[\pi_\theta, \overline{\pi}](s_t)\right]\right)\right] && (70) \end{aligned}$$

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

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

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.)

$$\begin{aligned} \tfrac{1}{2}(Q_\theta(s_t, a_t) - y_t)^2 &= \tfrac{1}{2}\left(\left(V_\theta(s_t, a_t) + \tau\log\left(\tfrac{\pi_\theta(a_t \mid s_t)}{\overline{\pi}(a_t \mid s_t)}\right)\right) - (V_{\theta_{\text{old}}}(s_t) + \tau\operatorname{KL}[\pi_{\theta_{\text{old}}}, \overline{\pi}](s_t) + \Delta_t)\right)^2 && (72)\\ &= \tfrac{1}{2}\left(\tau\log\left(\tfrac{\pi_\theta(a_t \mid s_t)}{\overline{\pi}(a_t \mid s_t)}\right) - (\Delta_t + \tau\operatorname{KL}[\pi_{\theta_{\text{old}}}, \overline{\pi}](s_t))\right)^2 && (73) \end{aligned}$$

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.

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

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

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

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

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

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

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.

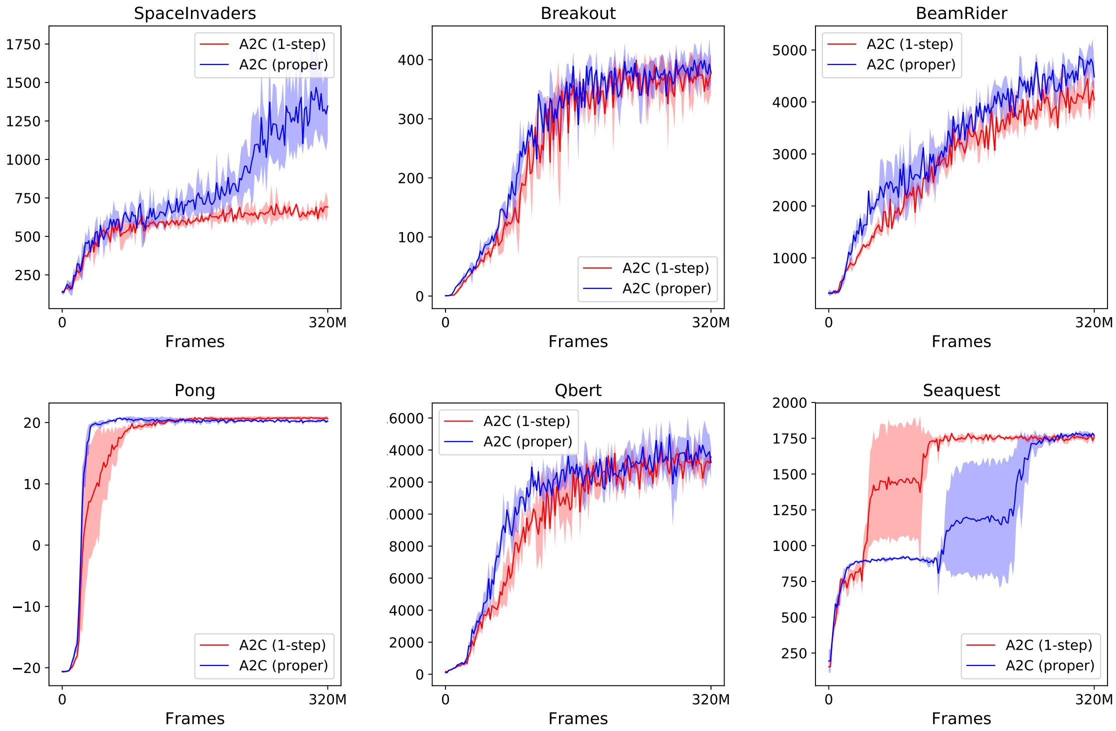


Figure 1: Atari performance with different RL objectives. EntRL is A2C modified to optimize for return augmented with entropy (instead of KL penalty). Solid lines are average evaluation return over 3 random seeds and shaded area is one standard deviation.

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

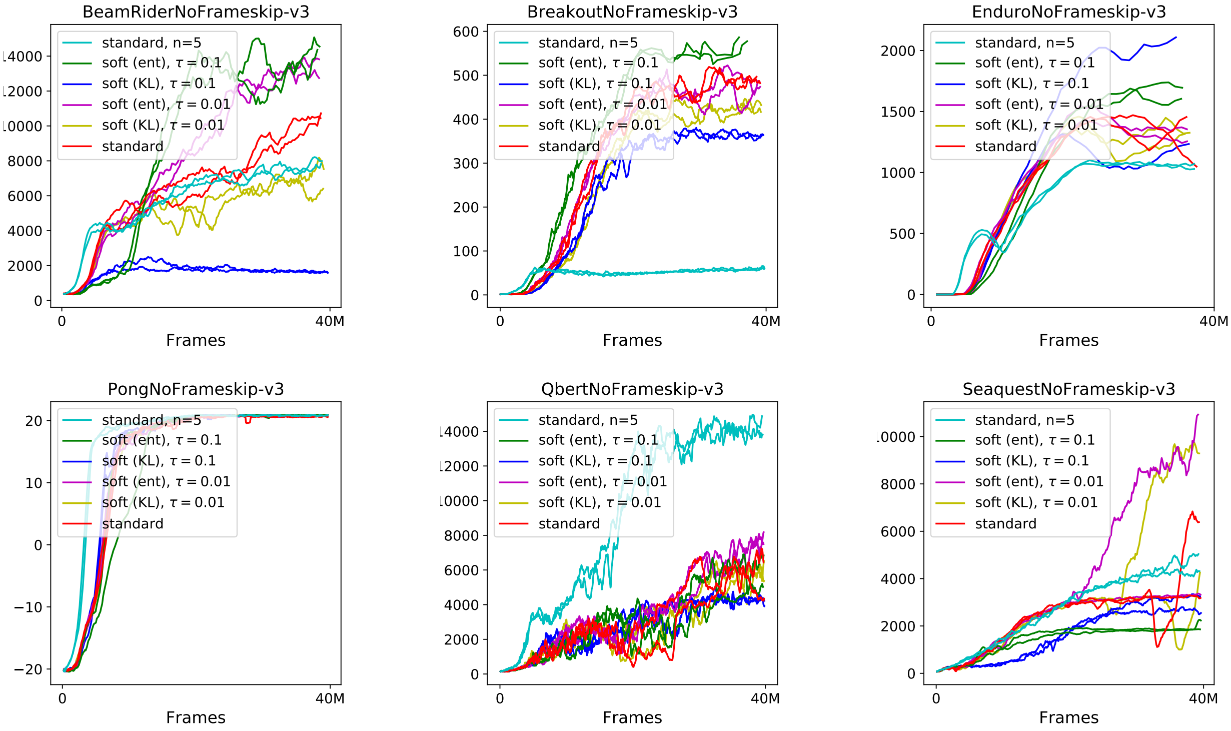


Figure 2: Different variants of soft Q-learning and standard Q-learning, applied to Atari games. Note that 4 frames = 1 timestep.

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:

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

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

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

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.

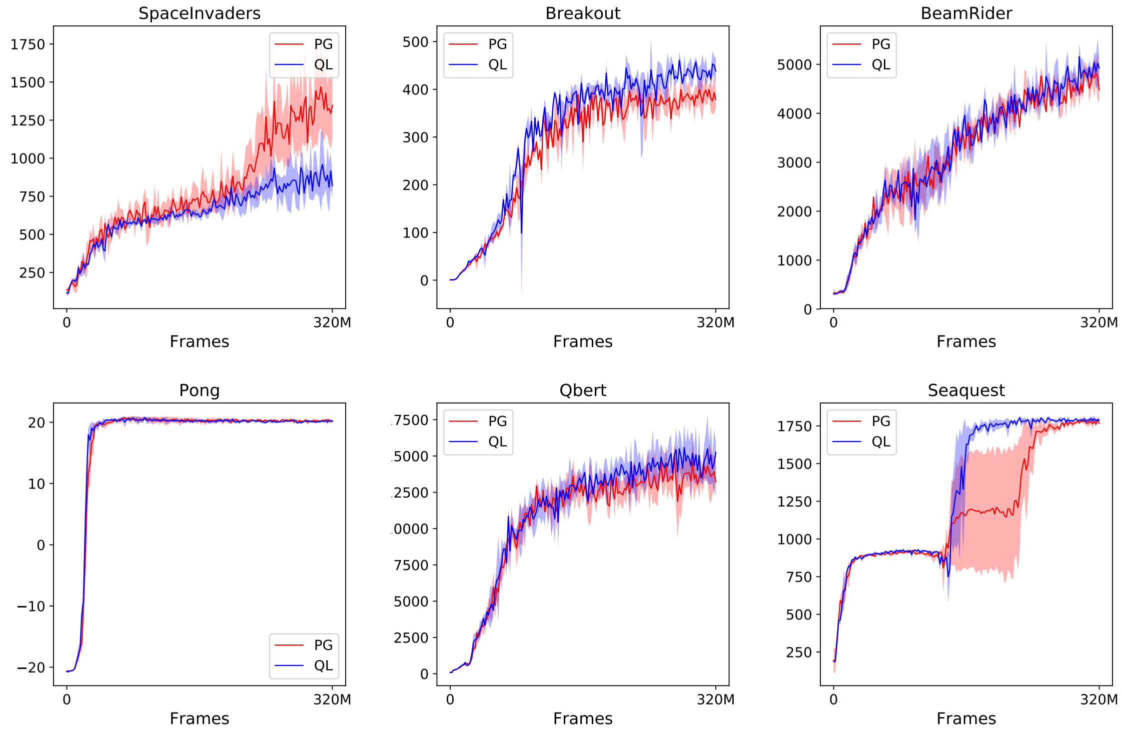


Figure 3: Atari performance with policy gradient vs Q-learning update rules. Solid lines are average evaluation return over 3 random seeds and shaded area is one standard deviation.

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.