ISL: Optimal Policy Learning With Optimal Exploration-Exploitation Trade-Off

Lucas Cassano[*]
Department of Electrical Engineering
University of California, Los Angeles
Los Angeles, CA 90095-1594
`cassanolucas@ucla.edu`

Ali H. Sayed
School of Engineering
École Polytechnique Fédérale de Lausanne
Lausanne, Switzerland CH-1015
`ali.sayed@epfl.ch`

Abstract

Traditionally, off-policy learning algorithms (such as Q-learning) and exploration schemes have been derived separately. Often times, the exploration-exploitation dilemma being addressed through heuristics. In this article we show that both the learning equations and the exploration-exploitation strategy can be derived in tandem as the solution to a unique and well-posed optimization problem whose minimization leads to the optimal value function. We present a new algorithm following this idea. The algorithm is of the gradient type (and therefore has good convergence properties even when used in conjunction with function approximators such as neural networks); it is off-policy; and it specifies both the update equations and the strategy to address the exploration-exploitation dilemma. To the best of our knowledge, this is the first algorithm that has these properties.

1 Introduction

Reinforcement learning (RL) is concerned with designing algorithms that seek to maximize long term cumulative rewards by interacting with an environment whose dynamics are unknown. Three main features are desirable for model-free algorithms to achieve this goal efficiently: (a) high sample efficiency, (b) guaranteed convergence (even when the algorithm is used in conjunction with expressive function approximators like neural networks) and (c) the ability to perform deep exploration. Recently, algorithms based on policy gradient have been introduced with guaranteed convergence and achieve state-of-the-art results in many tasks. The most notable cases being TRPO [1], PPO [2] and A3C [3]. These algorithms have two main drawbacks: they have poor sample efficiency due to the fact that they operate on-policy and they are not capable of performing deep exploration (i.e., they tend to perform poorly in environments with sparse rewards). Another group of algorithms is based on Q-learning (like DQN [4] and DDQN [5]). These algorithms have high sample efficiency but they also have two main drawbacks: they can diverge when used in conjunction with function approximators and they do not address the exploration-exploitation dilemma. Hence, heuristics are typically necessary to endow these algorithms with better exploration capabilities (for example, Bootstrap DQN [6]). More recently, a new family of algorithms has been introduced, which is based in the idea of learning policies that aim to maximize the long term cumulative rewards while also maximizing their own entropy. One notable algorithm within this group is SBEED [7], which has high sample efficiency and guaranteed convergence when used with function approximators. SBEED still has the deficiency that it does not address the exploration-exploitation dilemma and hence it does not perform efficient exploration. The contribution of this work is the introduction of a novel algorithm which, to the best of our knowledge, is the first algorithm that has the three aforementioned

[*]The author is also with the Institute of Electrical Engineering at EPFL. Alternative address `lucas.cassano@epfl.ch`.

properties (a)-(c). The main difference between our algorithm and previous work is that we use a novel cost function which makes the exploration-exploitation dilemma explicit and derives the learning rule and the exploratory strategy in tandem as the solution to a unique optimization problem.

1.1 Relation to prior work

Our paper is mostly related to recent work on maximum entropy algorithms. Some of the most prominent algorithms in this area are G-learning [8], soft Q-learning [9], PCL [10], SAC [11], Trust-PCL [12], and SBEED [7]. All these algorithms augment the traditional RL objective with a term that aims to maximize the entropy of the learned policy, which is weighted by a temperature parameter. The consequence of using this augmented objective is two-fold. First, it allows to derive convergent off-policy algorithms (even when used with function approximators). Second, it improves the exploration properties of the algorithms. However, using this augmented objective has two main drawbacks. In the first place, the policy to which these algorithms converge is biased away from the true optimal policy. This point can be handled by annealing the temperature parameter but this can slow down convergence. Furthermore, it is unclear what the optimal schedule is to perform such annealing and how it affects the conditioning of the optimization problem. In the second place, even though the exploration is improved, algorithms derived from this modified cost are not efficient at performing deep exploration. The reason for this is that a unique temperature parameter is used for all states. In order to perform deep exploration it is necessary to have a scheme that allows agents to learn policies which exploit more in states where the agent has high confidence in the optimal action and act in a more exploratory manner in unknown states. The main difference between our approach and these works is that we augment the traditional RL objective with a term that makes the exploration-exploitation trade-off explicit instead of the policy's entropy. Under our scheme, agents converge to the true optimal policy without the need for annealing of any parameters and, moreover, an exploration strategy is derived that is capable of performing deep exploration.

2 Preliminaries

We consider the problem of policy optimization within the traditional reinforcement learning framework. We model our setting as a Markov Decision Process (MDP), with an MDP defined by ($\mathcal{S}$,$\mathcal{A}$,$\mathcal{P}$,r), where $\mathcal{S}$ is a set of states of size $S = |\mathcal{S}|$, $\mathcal{A}$ is a set of actions of size $A = |\mathcal{A}|$, $\mathcal{P}(s'|s,a)$ specifies the probability of transitioning to state $s' \in \mathcal{S}$ from state $s \in \mathcal{S}$ having taken action $a \in \mathcal{A}$ and $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the average reward $r(s,a,s')$ when the agent transitions to state $s' \in \mathcal{S}$ from state $s \in \mathcal{S}$ having taken action $a \in \mathcal{A}$).

Assumption 1. *We assume* $r_{\min} \leq r(s,a,s') \leq r_{\max}$ *for all* (s,a,s').

In this work we consider the maximization of the discounted infinite reward as the objective of the RL agent:

$$\pi^\dagger(a|s) = \arg\max_\pi \mathbb{E}_{\mathcal{P},\pi}\left(\sum_{t=0}^{\infty} \gamma^t \boldsymbol{r}(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}) \Big| \boldsymbol{s}_0 = s\right) \tag{1}$$

where $\pi^\dagger(a|s)$ is the optimal action, $\gamma \in [0,1)$ is the discount factor and $\boldsymbol{s}_t$ and $\boldsymbol{a}_t$ are the state and action at time t, respectively. We clarify that in this work, random variables are always denoted in bold font. We recall that each policy π has an associated state value function $v^\pi(s)$ and state-action value function $q^\pi(s,a)$[2], and that value functions corresponding to the optimal policy are given by [16]:

$$q^\dagger(s,a) = r(s,a) + \gamma \mathbb{E}_{\mathcal{P}} \max_{a'} q^\dagger(\boldsymbol{s}', a'), \qquad \pi^\dagger(a|s) = \arg\max_\pi \sum_a \pi(a|s) q^\dagger(s,a) \tag{2}$$

where for convenience we defined $r(s,a) = \mathbb{E}_{\mathcal{P}} \boldsymbol{r}(s,a,\boldsymbol{s}')$.

3 Algorithm derivation

Optimization problem (1) and relations (2) are useful to derive algorithms for planning problems (i.e., problems in which the reward function and transition kernel are known) but are unfit to derive

[2]In this paper we will refer to both $v^\pi(s)$ and $q^\pi(s,a)$ as value functions indistinctly.

RL algorithms because they obviate the fact that the agent relies on estimated quantities (which are subject to uncertainty). Hence, in this work, we modify (1) to reflect the fact that an RL agent is constrained by the uncertainty of its estimates. Intuitively, we change the goal of the agent to not just maximize the discounted cumulative rewards but also to collect information about the MDP in order to minimize the uncertainty of its estimated quantities. For this purpose, we assume that at any point in time the agent has some estimate of the optimal value function denoted by $\widehat{q}(s,a)$, which is subject to some error $\tilde{q}(s,a) = q^{\dagger}(s,a) - \widehat{q}(s,a)$. Following a Bayesian approach we model the unknown quantities $\tilde{q}(s,a)$ as random variables. More specifically, we assume $\tilde{\boldsymbol{q}}(s,a)$ follows a uniform probability distribution with zero mean $\tilde{\boldsymbol{q}}(s,a) \sim \mathrm{U}(0, \ell(s,a))$ such that:

$$\tilde{q}(s,a) \in [\widehat{q}(s,a) - \ell(s,a); \widehat{q}(s,a) + \ell(s,a)] \tag{3}$$

We will refer to the probability density function of $\tilde{\boldsymbol{q}}(s,a)$ as $d_{(s,a)}(\tilde{q})$. We assume zero mean uniform distributions for the following reasons:

- Zero mean: if the mean were different than zero, it could be used to improve the estimate $\widehat{q}(s,a)$ as $\widehat{q}(s,a) \leftarrow \widehat{q}(s,a) + \mathbb{E}\tilde{\boldsymbol{q}}(s,a)$ resulting in a new estimate whose corresponding error would be zero mean.
- Uniform distribution: under assumption 1, we know that for any infinitely discounted MDP, a symmetric bound for the error of the value function exists in the form $-\ell(s,a) < \tilde{\boldsymbol{q}}(s,a) < \ell(s,a)$[3]. Moreover, typically there is no prior information about the error distribution between these bounds and therefore a non-informative uniform distribution becomes appropriate.

We further define the state uncertainty $\boldsymbol{\delta}^{\pi}(s)$ (whose distribution is given by a mixture of the state-action error distributions) and the Maximum-Error-Entropy policy $\pi^{\bullet}$ (which at every state chooses the action whose corresponding $\tilde{\boldsymbol{q}}(s,a)$ has the greatest uncertainty):

$$\boldsymbol{\delta}^{\pi}(s) \sim \mathbb{E}_{\pi} d_{(s,\boldsymbol{a})}(\tilde{q}) \tag{4a}$$

$$\pi^{\bullet}(a|s) = \begin{cases} 1, \text{ if } & \ell(s,a) = \max_a \ell(s,a) \\ 0, \text{ else} \end{cases} \tag{4b}$$

3.1 Optimization problem

We now define the optimization problem for our RL agent to be:

$$\pi^{\star}(a|s) = \arg\max_{\pi} \mathbb{E}_{\mathcal{P},\pi}\left(\sum_{t=0}^{\infty} \gamma^t \left[\boldsymbol{r}(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}) - \kappa D_{KL}(\boldsymbol{\delta}^{\pi}(\boldsymbol{s}_t) || \boldsymbol{\delta}^{\bullet}(\boldsymbol{s}_t)) \right] \Big| \boldsymbol{s}_0 = s \right) \tag{5}$$

where D_{KL} is the Kullback-Leibler divergence and $\boldsymbol{\delta}^{\bullet}$ is the state uncertainty corresponding to $\pi^{\bullet}$. In this work we refer to $\pi^{\star}(a|s)$ as the *uncertainty constrained* optimal policy (or *uc*-optimal policy). Under this new objective, we redefine the value functions as:

$$v^{\pi}(s) = \mathbb{E}_{\mathcal{P},\pi}\left(\sum_{t=0}^{\infty} \gamma^t \left[\boldsymbol{r}(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}) - \kappa D_{KL}(\boldsymbol{\delta}^{\pi}(\boldsymbol{s}_t) || \boldsymbol{\delta}^{\bullet}(\boldsymbol{s}_t)) \right] \Big| \boldsymbol{s}_0 = s \right) \tag{6a}$$

$$= \mathbb{E}_{\mathcal{P},\pi}\big(\boldsymbol{r}(s, \boldsymbol{a}_t, \boldsymbol{s'}) + \gamma v^{\pi}(\boldsymbol{s'})\big) - \kappa D_{KL}(\boldsymbol{\delta}^{\pi}(s) || \boldsymbol{\delta}^{\bullet}(s)) \tag{6b}$$

$$q^{\pi}(s,a) = r(s,a) + \gamma \mathbb{E}_{\mathcal{P}} v^{\pi}(\boldsymbol{s}') \tag{6c}$$

Using (6) we can rewrite (5) as:

$$\pi^{\star}(a|s) = \arg\max_{\pi} \sum_a \pi(a|s)\widehat{q}(s,a) - \kappa D_{KL}(\boldsymbol{\delta}^{\pi}(s) || \boldsymbol{\delta}^{\bullet}(s)) \tag{7}$$

Note that the exploration-exploitation trade-off becomes explicit in our cost function. To maximize the first term of the summation, the agent has to exploit its knowledge of $\widehat{q}$, while to maximize the second term, the agent's policy needs to match $\pi^{\bullet}$, which is a policy that seeks to maximize the information gathered through exploration. Since the argument being maximized in (7) is differentiable with respect to $\pi(a|s)$ we can obtain a closed-form expression for $\pi^{\star}$. Before providing the closed form solution for $\pi^{\star}$ we introduce the following useful lemma and definitions.

[3]This is due to the fact that the value functions are lower and upper bounded by $r_{\min}/(1-\gamma)$ and $r_{\max}/(1-\gamma)$.

Definition 1. ***Pareto dominated action***: *For a certain state s we say that an action a_j is Pareto dominated by action a_i if $\widehat{q}(s,a_j) \le \widehat{q}(s,a_i)$ and $\ell(s,a_j) < \ell(s,a_i)$.*

Lemma 1. *For all Pareto dominated actions it holds that $\pi^\star(a|s) = 0$.*

Proof. See appendix A. ■

The statement of Lemma 1 is intuitive since choosing a Pareto dominated action lowers the expected cumulative reward and the information gained, relative to choosing the action that dominates it. Also note that Lemma 1 implies that for all Pareto optimal actions it must be the case that if $q(a_i,s) < q(a_j,s)$ then $\ell(s,a_i) > \ell(s,a_j)$.

Definition 2. ***Mixed Pareto dominated action***: *For a certain state s we say that an action a_k is mixed Pareto dominated if there exists two actions a_i and a_j (which satisfy $\ell(s,a_i) > \ell(s,a_k) > \ell(s,a_j)$) such that:*

$$\widehat{q}(s,a_k) < \frac{(\ell(s,a_i) - \ell(s,a_k))\,\ell(s,a_j)\widehat{q}(s,a_j) + (\ell(s,a_k) - \ell(s,a_j))\,\ell(s,a_i)\widehat{q}(s,a_i)}{\ell(s,a_k)\,(\ell(s,a_i) - \ell(s,a_j))} \tag{8}$$

Definition 3. ***Pareto optimal action***: *We define an action a as Pareto optimal if it is not Pareto dominated or mixed Pareto dominated.*

We now introduce the state dependent set of actions $\mathcal{E}_s$ with cardinality $|\mathcal{E}_s|$, which is formed by all the Pareto optimal actions corresponding to state s. Furthermore, we introduce the ordering functions $\sigma_s(a) : [|\mathcal{E}_s|, 1] \to [1, A]$ which for every state provide an ordering amongst the Pareto optimal actions from lowest uncertainty to highest (i.e., $\ell(s,\sigma_s(i)) > \ell(s,\sigma_s(j)) \iff i > j$). For instance, $\sigma_s(1)$ provides the index of the action at state s, which has the lowest uncertainty amongst the actions contained in $\mathcal{E}_s$.

Theorem 1. *$\pi^\star(a|s)$ is given by:*

$$\pi^\star(a|s) = \begin{cases} \frac{\ell_{\sigma(j)}(p_j(s) - p_{j+1}(s))}{\sum_{j=1}^{|\mathcal{E}_s|}(\ell_{\sigma(j)} - \ell_{\sigma(j-1)})p_j(s)}, & \textit{if } a = \sigma_s(j), \textit{for some } j \in [1, |\mathcal{E}_s|] \\ 0, & \textit{otherwise} \end{cases} \tag{9a}$$

$$p_j(s) = \exp\left[\frac{\ell_{\sigma(j)}(s)\widehat{q}(s,\sigma(j)) - \ell_{\sigma(j-1)}(s)\widehat{q}(s,\sigma(j-1))}{\kappa\left(\ell_{\sigma(j)}(s) - \ell_{\sigma(j-1)}(s)\right)}\right] \tag{9b}$$

where to simplify notation we defined $\ell_{\sigma(j)}(s) = \ell(s,\sigma_s(j))$ and $\widehat{q}(s,\sigma(j)) = \widehat{q}(s,\sigma_s(j))$ and we also set $p_{|\mathcal{E}_s|+1}(s) = 0$, $l_0(s) = 0$.

Proof. See Appendix B. ■

Note that as expected, according to (9), the *uc*-optimal policy always assigns strictly positive probability to the actions that have the biggest uncertainty and biggest $\widehat{q}$ (in cases where one action a_k has both then $\pi^\star(a_k|s) = 1$).

Lemma 2. *The value function corresponding to policy $\pi^\star(a|s)$ is given by:*

$$v^\star(s) = \kappa \log\left[\sum_{j=1}^{|\mathcal{E}_s|} \frac{(\ell_{\sigma(j)}(s) - \ell_{\sigma(j-1)}(s))}{\ell_{\max}(s)} p_j(s)\right], \quad q^\star(s,a) = r(s,a) + \gamma \mathbb{E} v^\star(\boldsymbol{s'}) \tag{10}$$

where $\ell_{\max}(s) = \max_a \ell(s,a)$.

Proof. The proof follows by combining (10) with (6b) and (6c). ■

Remark 1. Note that the pair $[\pi^\star(a|s), v^\star(s)]$ satisfies two important conditions:

$$\lim_{\kappa \to 0^+} [\pi^\star(a|s), v^\star(s)] = [\pi^\dagger(a|s), v^\dagger(s)], \qquad \lim_{\ell_A \to \ell_{A-1}^+ \to \cdots \to \ell_1^+} [\pi^\star(a|s), v^\star(s)] = [\pi^\dagger(a|s), v^\dagger(s)] \tag{11}$$

The first condition is expected since when the relative entropy term is eliminated, (1) and (5) become equivalent. The second condition reflects the fact that when the uncertainty is equal for all actions the distributions $\boldsymbol{\delta}^\pi(s)$ and $\boldsymbol{\delta}^\bullet(s)$ become equal regardless of π and therefore $D_{KL}(\boldsymbol{\delta}^\pi(s)||\boldsymbol{\delta}^\bullet(s)) = 0$ and hence (1) and (5) become equivalent. The second condition of (11) is of fundamental importance because it guarantees that as learning progresses and the limits $\ell(s,a)$ diminish, policy $\pi^\star$ tends to the desired policy $\pi^\dagger$ (note that annealing of κ is not necessary for this convergence of $\pi^\star$ towards $\pi^\dagger$).

3.2 $\widehat{q}$ update equations

Using the relations from Theorem 1 and Lemma 2 we pose the following optimization problem:

$$\min_{\omega} \frac{1}{2}\mathbb{E}_{\psi}\Bigg[\underbrace{\widehat{q}(s,a;\omega) - r(s,a) - \gamma\kappa\mathbb{E}_{\mathcal{P}}\log\left(\sum_{j=1}^{|\mathcal{E}_s|}\frac{(\ell_{\sigma(j)}(\boldsymbol{s'}) - \ell_{\sigma(j-1)}(\boldsymbol{s'}))}{\ell_{\max}(\boldsymbol{s'})}p_j(s';\omega)\right)}_{=\delta_{\omega}(s,a)}\Bigg]^2 \tag{12}$$

where ψ is the distribution according to which state-action pairs are sampled, and $q(s,a;\omega)$ is a parametric approximation of $q(s,a)$ with parameters ω. Note that the gradient of $[\delta_{\omega}(s,a)]^2$ with respect to ω is a product of expectations. Therefore, in the general case where transitions are stochastic sample estimates of such gradient become biased. To bypass this issue, we use the *duality trick* [7, 17–21] as follows:

$$\frac{1}{2}\mathbb{E}_{\psi}\delta(s,a)^2 = \mathbb{E}_{\psi}\max_{\rho}\rho(s,a)\delta(s,a) - \frac{1}{2}\rho(s,a)^2 = \max_{\rho}\mathbb{E}_{\psi}\rho(s,a)\delta(s,a) - \frac{1}{2}\rho(s,a)^2 \tag{13}$$

Hence, minimization problem (12) becomes equivalent to the following primal-dual formulation:

$$\min_{\omega}\max_{\theta}\underbrace{\mathbb{E}_{\psi}\left[\eta\left(\rho(s,a;\theta)\delta(s,a) - \frac{1}{2}\rho(s,a;\theta)^2\right) + \frac{(1-\eta)}{2}\delta(s,a)^2\right]}_{\triangleq J_{\eta}(\omega,\theta)} \tag{14}$$

where $0 \leq \eta \leq 1$ and $\rho(s,a;\theta)$ is the parameterized version of $\rho(s,a)$ through parameters θ. The gradients of (14) are given by:

$$\nabla_{\theta}J_{\eta}(\omega,\theta) = \mathbb{E}\eta(\delta_{\omega}(\boldsymbol{s},\boldsymbol{a}) - \rho(\boldsymbol{s},\boldsymbol{a};\theta))\nabla_{\theta}\rho(\boldsymbol{s},\boldsymbol{a};\theta) \tag{15a}$$

$$\nabla_{\omega}J_{\eta}(\omega,\theta) = \mathbb{E}(\eta\rho(\boldsymbol{s},\boldsymbol{a};\theta) + (1-\eta)\delta_{\omega}(\boldsymbol{s},\boldsymbol{a}))\left(\nabla_{\omega}\widehat{q}(\boldsymbol{s},\boldsymbol{a};\omega) - \gamma\sum_{j=1}^{|\mathcal{E}_s|}\pi^{\star}(a_j|\boldsymbol{s'})\nabla_{\omega}\widehat{q}(\boldsymbol{s'},a_j;\omega)\right) \tag{15b}$$

The $0 \leq \eta \leq 1$ parameter allows control of a variance-bias trade-off for the estimate of the gradient. In the particular case where the transitions of the MDP are deterministic, the optimal choice is $\eta = 0$. However, while the entropy of the distribution over state transitions increases, higher values of η become preferable. Removing the expectation from expressions (15) we obtain sample estimates for the gradients (which are unbiased and have bounded variance). Note that these stochastic gradients can be used to define a learning algorithm, but such algorithm would be incomplete since we are still missing update equations for the uncertainty estimators $\ell(s,a)$.

3.3 Uncertainty Estimation

Recalling that $\widetilde{q}(s,a) = q^{\dagger}(s,a) - \widehat{q}(s,a)$ and the definition of $\delta(s,a)$ in (12) we can write:

$$\widetilde{q}(s,a) = r(s,a) + \gamma\mathbb{E}_{\boldsymbol{s'}}v^{\dagger}(\boldsymbol{s'}) - \widehat{q}(s,a) \overset{(a)}{=} -\delta(s,a) + \gamma\mathbb{E}_{\boldsymbol{s'}}\widetilde{v}(\boldsymbol{s'}) \tag{16}$$

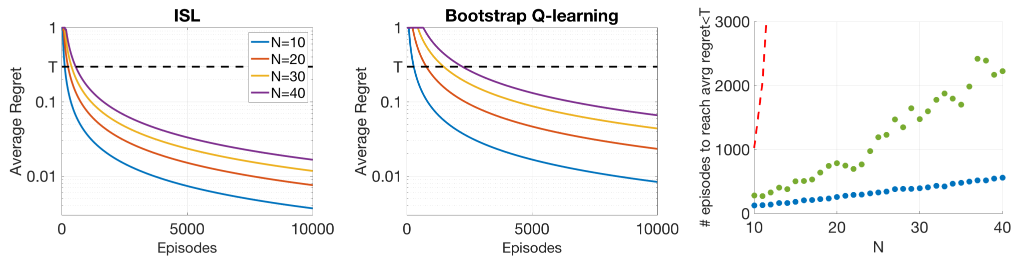
where in (a) we used $v^{\dagger}(s) = \widehat{v}(s) + \widetilde{v}(s)$, and further $\widehat{v}(s)$ is the estimate obtained using (10) and $\widehat{q}(s,a)$. Note that we don't have a closed form expression for $\widetilde{v}(s')$ (because $v^{\dagger}(s')$ is unknown), but we can bound it as follows:

$$v^{\dagger}(s) - \widehat{v}(s) = \max_{a} q^{\dagger}(s,a) - \widehat{v}(s) \overset{(b)}{\leq} \max_{a} q^{\dagger}(s,a) - \max_{a}\widehat{q}(s,a) \overset{(c)}{\leq} \max_{a}\ell(s,a) \tag{17}$$

where (b) follows from Jensen's inequality applied to (10) and in (c) we applied (3). Combining (16) and (17) we get:

$$|\widetilde{q}(s,a)| \leq |\delta(s,a)| + \gamma\mathbb{E}_{\boldsymbol{s'}}\max_{a}\ell(\boldsymbol{s'},a) \tag{18}$$

Therefore updating $\ell(s,a)$ as $\ell(s,a) = |\delta(s,a)| + \gamma\mathbb{E}_{\boldsymbol{s'}}\max_a \ell(\boldsymbol{s'},a)$ satisfies condition (3). However, note that in general $\delta(s,a) = \widehat{q}(s,a) - \mathbb{E}[\boldsymbol{r}(s,a,\boldsymbol{s'}) + \gamma\widehat{v}(\boldsymbol{s'})]$ is not known because it depends on the reward function and the transition probabilities $\mathcal{P}$, which are not known. Therefore, to obtain an update equation for $\ell(s,a)$ we need an estimate of $\delta(s,a)$. In the particular case where the MDP is



(a) (b) (c)

Figure 1: The legend for (b) is the same as (a). In (c) the red dashed line plots 2^N, the blue and green dots correspond to ISL and Bootstrap Q-learning, respectively.

Algorithm 1 Information Seeking Learner (ISL)

```
Initialize: $\omega$, $\theta$ and $\nu$ randomly, and an empty replay buffer $\mathcal{D}$.
for environment transitions $t = 0, \ldots, T$ do
    Sample transitions $(s, a, r, s')$ by following policy (9) and store them in $\mathcal{D}$.
    $\nu \leftarrow \arg\min_\nu \mathbb{E}_{(s,a,s')\sim\mathcal{D}} J_U(\nu)$
    for iterations $i = 0, \ldots, I$ do
        Sample a minibatch from $\mathcal{D}$ and compute stochastic gradients $\widehat{\nabla}_\theta J_\eta(\omega, \theta)$ and $\widehat{\nabla}_\omega J_\eta(\omega, \theta)$
        according to (15).
        $\omega \leftarrow \omega - \mu_\omega \widehat{\nabla}_\omega J_\eta(\omega, \theta)$
        $\theta \leftarrow \theta + \mu_\theta \widehat{\nabla}_\theta J_\eta(\omega, \theta)$
    end for
end for
```

deterministic this is not an issue and $\delta(s, a)$ can be calculated with any sample (s, a, r, s') transition as $\delta(s, a) = \widehat{q}(s, a) - r(s, a, s') + \gamma\widehat{v}(s')$. However in the general case where the transitions and/or the rewards are stochastic, $\delta(s, a)$ cannot be estimated using a single transition. Nonetheless, note from (15a) that $\delta(s, a)$ is precisely the quantity that $\rho(s, a)$ estimates. Hence, similarly as we did in (14), we can estimate $|\delta(s, a)|$ as:

$$\eta|\delta_i(s, a)| + (1 - \eta)|\rho(s, a)| \tag{19}$$

where $\delta_i(s, a)$ is the estimate of $\delta(s, a)$ using some sample i. Note that we reuse the same parameter η from (14) which provides a bias-variance trade-off for the estimation of $|\delta(s, a)|$. Combining (18) and (19) we get the following update equation for $\ell(s, a)$:

$$\ell(s, a) \leftarrow \eta|\delta_t(s, a)| + (1 - \eta)|\rho(s, a)| + \gamma\mathbb{E}_{\boldsymbol{s}'} \max_a \ell(\boldsymbol{s}', a) \tag{20}$$

In the case where $\ell(s, a)$ is parameterized with parameters ν we can update such parameters through solving the following minimization problem:

$$\min_{\nu_{t+1}} \mathbb{E}_\psi \underbrace{2^{-1}[\ell(s, a; \nu_{t+1}) - \eta|\delta_t(s, a)| + (1 - \eta)|\rho(s, a)| + \gamma\mathbb{E}_{\boldsymbol{s}'}\ell_{\max}(\boldsymbol{s}'; \nu_t)]^2}_{=J_U(\nu_{t+1})} \tag{21}$$

Using (21) and the stochastic gradients obtained by removing the expectations in (15), we introduce a new algorithm which we refer to as *Information Seeking Learner (ISL)*, see Algorithm 1.

4 Experiments

In this section we test the capabilities of ISL to perform deep exploration. We compare the performance of a tabular implementation of ISL and Bootsrtap Q-learning in the Deep Sea game [22], which is useful benchmark to test the exploration capabilities of RL algorithms. Implementation details and results for the stochastic versions of Deep Sea can be found in Appendix C. Figures 1a and 1b show the average regret curves and figure 1c shows the amount episodes required for the regret to drop below the dotted lines indicated with T in figures 1a and 1b. As can be seen in the figures, the amount of episodes required to learn the optimal policy is $\mathcal{O}(N)$ which is optimal [22], which is shows that ISL's exploration mechanism is efficient.

References

[1] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in *International Conference on Machine Learning*, New York, USA, 2015, pp. 1889–1897.

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv:1707.06347*, August 2017.

[3] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in *Proc. International Conference on Machine Learning*, New York, USA, 2016, pp. 1928–1937.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, p. 529, 2015.

[5] H. Van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-learning,” in *Proc. AAAI Conference on Artificial Intelligence*, Arizona, USA, 2016.

[6] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, “Deep exploration via bootstrapped dqn,” in *Proc. Advances in neural information processing systems*, Barcelona, Spain, 2016, pp. 4026–4034.

[7] B. Dai, A. Shaw, L. Li, L. Xiao, N. He, Z. Liu, J. Chen, and L. Song, “SBEED: Convergent reinforcement learning with nonlinear function approximation,” in *Proc. International Conference on Machine Learning*, Stockholm, Sweden, 2018, pp. 1133–1142.

[8] R. Fox, A. Pakman, and N. Tishby, “Taming the noise in reinforcement learning via soft updates,” in *Proc. Conference on Uncertainty in Artificial Intelligence*, New York, USA, 2016, pp. 202–211.

[9] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, “Reinforcement learning with deep energy-based policies,” in *Proc. International Conference on Machine Learning-Volume 70*, Sydney, Australia, 2017, pp. 1352–1361.

[10] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans, “Bridging the gap between value and policy based reinforcement learning,” in *Proc. Advances in Neural Information Processing Systems*, Long Beach, USA, 2017, pp. 2775–2785.

[11] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *Proc. International Conference on Machine Learning*, Stockholm, Sweden, 2018, pp. 1856–1865.

[12] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans, “Trust-PCL: An off-policy trust region method for continuous control,” *arXiv:1707.01891*, February 2018.

[13] B. C. Stadie, S. Levine, and P. Abbeel, “Incentivizing exploration in reinforcement learning with deep predictive models,” *arXiv:1507.00814*, 2015.

[14] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros, “Large-scale study of curiosity-driven learning,” *arXiv:1808.04355*, 2018.

[15] I. Osband, J. Aslanides, and A. Cassirer, “Randomized prior functions for deep reinforcement learning,” in *Proc. Advances in Neural Information Processing Systems*, Montr’eal, Canada, 2018, pp. 8617–8629.

[16] M. L. Puterman, *Markov Decision Processes.: Discrete Stochastic Dynamic Programming*. Wiley, NY, 2014.

[17] L. Cassano, S. A. Alghunaim, and A. H. Sayed, “Team policy learning for multi-agent reinforcement learning,” in Proc. IEEE *International Conference on Acoustics, Speech and Signal* Processing *(ICASSP)*, Brighton, UK, May 2019, pp. 3062–3066.

[18] S. V. Macua, J. Chen, S. Zazo, and A. H. Sayed, “Distributed policy evaluation under multiple behavior strategies,” *IEEE Transactions on Automatic Control*, vol. 60, no. 5, pp. 1260–1274, 2015.

[19] S. S. Du, J. Chen, L. Li, L. Xiao, and D. Zhou, “Stochastic variance reduction methods for policy evaluation,” in *Proc. International Conference on Machine Learning*, Sydney, Australia, 2017, pp. 1049–1058.

[20] L. Cassano, K. Yuan, and A. H. Sayed, “Distributed value-function learning with linear convergence rates,” in *Proc. of European Control Conference*, Napoli, Italy, 2019, pp. 505–511.

[21] L. Cassano, K. Yuan, and A. H. Sayed, “Multi-agent fully decentralized value function learning with linear convergence rates,” *arXiv:1810.07792*, October 2018.

[22] I. Osband, B. Van Roy, D. Russo, and Z. Wen, “Deep exploration via randomized value functions,” *arXiv:1703.07608*, March 2019.

[23] I. Osband, Y. Doron, M. Hessel, J. Aslanides, E. Sezener, A. Saraiva, K. McKinney, T. Lattimore, C. Szepezvari, S. Singh, B. Van Roy, R. Sutton, D. Silver, and H. Van Hasselt, “Behaviour suite for reinforcement learning,” *arXiv:1908.03568*, August 2019.

A Proof of Lemma 1

We start stating the following assumption to avoid having using function $\sigma_s(a)$ through the entire derivation.

Assumption 2. *Without loss of generality we assume that actions are numbered such that $\ell_{a_j}(s) > \ell_{a_i}(s) \iff j > i$. This implies that action a_A is the one whose bellman error has the biggest uncertainty, while action a_1 is the one with the lowest.*

We prove the lemma by contradiction. Assume that action a_j is Pareto dominated by action a_i, and further assume that there exists a uc-optimal policy $\pi^{\triangle}(a|s)$ for which $\pi^{\triangle}(a_j|s) = \delta > 0$. We construct policy $\pi^{\square}(a|s)$ and show that $v^{\square}(s) > v^{\triangle}(s)$ and therefore $\pi^{\triangle}(a|s)$ is not a uc-optimal policy.

$$\pi^{\square}(a|s) = \begin{cases} \pi^{\triangle}(a|s) & if \;\; a \neq a_j \wedge a \neq a_i \\ \pi^{\triangle}(a_i|s) + \pi^{\triangle}(a_j|s) & if \;\; a = a_i \\ 0 & if \;\; a = a_j \end{cases} \tag{22}$$

Note that since we assume a_i dominates a_j due to assumption 2 this implies that $i > j$. Without loss of generality we assume $i = j + 1$. We now proceed to show that $D_{KL}(\boldsymbol{\delta}^{\square}(s)||\boldsymbol{\delta}^{\bullet}(s)) < D_{KL}(\boldsymbol{\delta}^{\triangle}(s)||\boldsymbol{\delta}^{\bullet}(s))$.

$$D_{KL}(\boldsymbol{\delta}^{\pi}(s)||\boldsymbol{\delta}^{\bullet}(s)) = \int_b \sum_a \pi(a|s) d_{(s,a)}(\delta) \log\left(\frac{\sum_{a'} \pi(a'|s) d_{(s,a')}(b)}{\sum_{a'} \pi^{\bullet}(a'|s) d_{(s,a')}(b)}\right) db \tag{23}$$

$$\overset{(b)}{=} \sum_a \pi(a|s) \int_b d_{(s,a)}(\delta) \log\left(\sum_{a'} \pi(a'|s) d_{(s,a')}(b)\right) db + \log(\ell_A) \tag{24}$$

where in (b) we used the fact that $\pi^{\bullet}(a|s) = \mathbb{I}\left[\arg\max_a \ell(s,a)\right]$. Now using assumption 2 and the fact that all error densities $d_{(s,a')}$ are uniform we can write a closed form expression for the integral.

$$\int_b d_{(s,a)}(\delta) \log\left(\sum_{a'} \pi(a'|s) d_{(s,a')}(b)\right) db = \int_0^{\ell_j} \ell_j^{-1} \log\left(\sum_{a'} \pi(a'|s) d_{(s,a')}(b)\right) db \tag{25}$$

$$= \ell_j^{-1} \int_{\ell_{j-1}}^{\ell_j} \log\left(\sum_{a'} \pi(a'|s) d_{(s,a')}(b)\right) db + \ell_j^{-1} \int_0^{\ell_{j-1}} \log\left(\sum_{a'} \pi(a'|s) d_{(s,a')}(b)\right) db \tag{26}$$

$$= \frac{\ell_j - \ell_{j-1}}{\ell_j} \log\left(\sum_{b=0}^{A-j} \pi(A-b|s) \ell_{A-b}^{-1}\right) + \ell_j^{-1} \int_0^{\ell_{j-1}} \log\left(\sum_{a'} \pi(a'|s) d_{(s,a')}(b)\right) db \tag{27}$$

$$= \sum_{n=1}^{j} \frac{\ell_n - \ell_{n-1}}{\ell_j} \log\left(\sum_{b=n}^{A} \frac{\pi(b|s)}{\ell_b}\right) \tag{28}$$

Combining (24) and (28) we get:

$$D_{KL}(\boldsymbol{\delta}^{\pi}(s)||\boldsymbol{\delta}^{\bullet}(s)) = \sum_{k=1}^{A} \frac{\pi(a_k|s)}{\ell_k} \sum_{n=1}^{k} (\ell_n - \ell_{n-1}) \log\left(\sum_{b=n}^{A} \frac{\pi(b|s)}{\ell_b}\right) - \log(\ell_A) \tag{29}$$

$$= \sum_{n=1}^{A} (\ell_n - \ell_{n-1}) \left(\sum_{k=n}^{A} \frac{\pi(a_k|s)}{\ell_k}\right) \log\left(\sum_{b=n}^{A} \frac{\pi(b|s)}{\ell_b}\right) - \log(\ell_A) \tag{30}$$

Combining (22) and (30) we can write:

$$D_{KL}(\boldsymbol{\delta}^{\square}(s)||\boldsymbol{\delta}^{\bullet}(s)) - \log(\ell_A) = \sum_{n=1}^{A} (\ell_n - \ell_{n-1}) \left(\sum_{k=n}^{A} \frac{\pi^{\square}(a_k|s)}{\ell_k}\right) \log\left(\sum_{b=n}^{A} \frac{\pi^{\square}(b|s)}{\ell_b}\right) \tag{31}$$

$$= \sum_{n=i+1}^{A} (\ell_n - \ell_{n-1}) \left(\sum_{k=n}^{A} \frac{\pi^{\triangle}(a_k|s)}{\ell_k}\right) \log\left(\sum_{b=n}^{A} \frac{\pi^{\triangle}(b|s)}{\ell_b}\right)$$

$$+ (\ell_i - \ell_j) \left(\sum_{k=i}^{A} \frac{\pi^{\triangle}(a_k|s)}{\ell_k} + \frac{\pi^{\triangle}(a_j|s)}{\ell_i} \right) \log \left(\sum_{b=i}^{A} \frac{\pi^{\triangle}(b|s)}{\ell_b} + \frac{\pi^{\triangle}(a_j|s)}{\ell_i} \right)$$

$$+ \sum_{n=1}^{j} (\ell_n - \ell_{n-1}) \left(\sum_{\substack{k=n \\ k \neq j}}^{A} \frac{\pi^{\triangle}(a_k|s)}{\ell_k} + \frac{\pi^{\triangle}(a_j|s)}{\ell_i} \right) \log \left(\sum_{\substack{b=n \\ b \neq j}}^{A} \frac{\pi^{\triangle}(b|s)}{\ell_b} + \frac{\pi^{\triangle}(a_j|s)}{\ell_i} \right) \tag{32}$$

For $D_{KL}(\boldsymbol{\delta}^{\triangle}(s)||\boldsymbol{\delta}^{\bullet}(s))$ we get:

$$D_{KL}(\boldsymbol{\delta}^{\triangle}(s)||\boldsymbol{\delta}^{\bullet}(s)) - \log(\ell_A) = \sum_{n=i+1}^{A} (\ell_n - \ell_{n-1}) \left(\sum_{k=n}^{A} \frac{\pi^{\triangle}(a_k|s)}{\ell_k} \right) \log \left(\sum_{b=n}^{A} \frac{\pi^{\triangle}(b|s)}{\ell_b} \right)$$

$$+ (\ell_i - \ell_j) \left(\sum_{k=i}^{A} \frac{\pi^{\triangle}(a_k|s)}{\ell_k} \right) \log \left(\sum_{b=i}^{A} \frac{\pi^{\triangle}(b|s)}{\ell_b} \right)$$

$$+ \sum_{n=1}^{j} (\ell_n - \ell_{n-1}) \left(\sum_{\substack{k=n \\ k \neq j}}^{A} \frac{\pi^{\triangle}(a_k|s)}{\ell_k} + \frac{\pi^{\triangle}(a_j|s)}{\ell_j} \right) \log \left(\sum_{\substack{b=n \\ b \neq j}}^{A} \frac{\pi^{\triangle}(b|s)}{\ell_b} + \frac{\pi^{\triangle}(a_j|s)}{\ell_j} \right) \tag{33}$$

For the purposes of simplifying equations (and only for the remainder of this subsection) we define:

$$x_n \triangleq \sum_{\substack{k=n \\ k \neq j}}^{A} \frac{\pi^{\triangle}(a_k|s)}{\ell_k} \tag{34}$$

$$\pi^{\triangle}(a_j|s) \triangleq \pi_j \tag{35}$$

Combining (32) and (33) we get:

$$\exp\Big(D_{KL}(\boldsymbol{\delta}^{\triangle}(s)||\boldsymbol{\delta}^{\bullet}(s)) - D_{KL}(\boldsymbol{\delta}^{\square}(s)||\boldsymbol{\delta}^{\bullet}(s)) \Big) = \left(\frac{x_i}{x_i + \frac{\pi_j}{\ell_i}} \right)^{(\ell_i - \ell_j) x_i}$$

$$\cdot \prod_{n=1}^{j} \left[\frac{\left(x_n + \frac{\pi_j}{\ell_j} \right)^{\left(x_n + \frac{\pi_j}{\ell_j} \right)}}{\left(x_n + \frac{\pi_j}{\ell_i} \right)^{\left(x_n + \frac{\pi_j}{\ell_i} \right)}} \right]^{(\ell_n - \ell_{n-1})} \left(x_i + \frac{\pi_j}{\ell_i} \right)^{-\left(\frac{(\ell_i - \ell_j)}{\ell_i} \pi_j \right)} \tag{36}$$

$$= \left(1 + \frac{\pi_j}{x_i \ell_i} \right)^{-(\ell_i - \ell_j) x_i} \left(x_i + \frac{\pi_j}{\ell_i} \right)^{-\left(\frac{(\ell_i - \ell_j)}{\ell_i} \pi_j \right)} \prod_{n=1}^{j} \left[\frac{\left(1 + \frac{\pi_j}{x_n \ell_j} \right)^{\left(x_n + \frac{\pi_j}{\ell_j} \right)}}{\left(1 + \frac{\pi_j}{x_n \ell_i} \right)^{\left(x_n + \frac{\pi_j}{\ell_i} \right)}} x_n^{\left(\frac{\ell_i - \ell_j}{\ell_j \ell_i} \pi_j \right)} \right]^{\ell_n - \ell_{n-1}} \tag{37}$$

$$= \left(1 + \frac{\pi_j}{x_i \ell_i} \right)^{-(\ell_i - \ell_j) x_i} \left(1 + \frac{\pi_j}{x_i \ell_i} \right)^{-\left(\frac{(\ell_i - \ell_j)}{\ell_i} \pi_j \right)} \left[\frac{\left(1 + \frac{\pi_j}{x_i \ell_j} \right)^{\left(x_i + \frac{\pi_j}{\ell_j} \right)}}{\left(1 + \frac{\pi_j}{x_i \ell_i} \right)^{\left(x_i + \frac{\pi_j}{\ell_i} \right)}} \right]^{\ell_j - \ell_{j-1}} x_i^{-\ell_{j-1} \left(\frac{\ell_i - \ell_j}{\ell_j \ell_i} \pi_j \right)}$$

$$\cdot \prod_{n=1}^{j-1} \left[\frac{\left(1 + \frac{\pi_j}{x_n \ell_j} \right)^{\left(x_n + \frac{\pi_j}{\ell_j} \right)}}{\left(1 + \frac{\pi_j}{x_n \ell_i} \right)^{\left(x_n + \frac{\pi_j}{\ell_i} \right)}} x_n^{\left(\frac{\ell_i - \ell_j}{\ell_j \ell_i} \pi_j \right)} \right]^{\ell_n - \ell_{n-1}} \tag{38}$$

$$= \left(1 + \frac{\pi_j}{x_i \ell_i} \right)^{-(\ell_i - \ell_j) \left(x_i + \frac{\pi_j}{\ell_i} \right)} \left[\frac{\left(1 + \frac{\pi_j}{x_i \ell_j} \right)^{\left(x_i + \frac{\pi_j}{\ell_j} \right)}}{\left(1 + \frac{\pi_j}{x_i \ell_i} \right)^{\left(x_i + \frac{\pi_j}{\ell_i} \right)}} \right]^{\ell_j} \left[\frac{\left(1 + \frac{\pi_j}{x_i \ell_j} \right)^{\left(x_i + \frac{\pi_j}{\ell_j} \right)}}{\left(1 + \frac{\pi_j}{x_i \ell_i} \right)^{\left(x_i + \frac{\pi_j}{\ell_i} \right)}} \right]^{-\ell_{j-1}} x_i^{-\ell_{j-1} \left(\frac{\ell_i - \ell_j}{\ell_j \ell_i} \pi_j \right)}$$

$$\cdot \prod_{n=1}^{j-1} \left[\frac{\left(1+\frac{\pi_j}{x_n \ell_j}\right)^{\left(x_n+\frac{\pi_j}{\ell_j}\right)}}{\left(1+\frac{\pi_j}{x_n \ell_i}\right)^{\left(x_n+\frac{\pi_j}{\ell_i}\right)}} x_n^{\left(\frac{\ell_i-\ell_j}{\ell_j \ell_i}\pi_j\right)} \right]^{\ell_n-\ell_{n-1}} \tag{39}$$

$$= \frac{\left(1+\frac{\pi_j}{x_i \ell_j}\right)^{(\ell_j x_i+\pi_j)}}{\left(1+\frac{\pi_j}{x_i \ell_i}\right)^{(\ell_i x_i+\pi_j)}} \left[\frac{\left(1+\frac{\pi_j}{x_j \ell_j}\right)^{\left(x_j+\frac{\pi_j}{\ell_j}\right)}}{\left(1+\frac{\pi_j}{x_j \ell_i}\right)^{\left(x_j+\frac{\pi_j}{\ell_i}\right)}} \right]^{-\ell_{j-1}} x_j^{-\ell_{j-1}\left(\frac{\ell_i-\ell_j}{\ell_j \ell_i}\pi_j\right)}$$

$$\cdot \prod_{n=1}^{j-1} \left[\frac{\left(1+\frac{\pi_j}{x_n \ell_j}\right)^{\left(x_n+\frac{\pi_j}{\ell_j}\right)}}{\left(1+\frac{\pi_j}{x_n \ell_i}\right)^{\left(x_n+\frac{\pi_j}{\ell_i}\right)}} x_n^{\left(\frac{\ell_i-\ell_j}{\ell_j \ell_i}\pi_j\right)} \right]^{\ell_n-\ell_{n-1}} \tag{40}$$

$$= \frac{\left(1+\frac{\pi_j}{x_i \ell_j}\right)^{(\ell_j x_i+\pi_j)}}{\left(1+\frac{\pi_j}{x_i \ell_i}\right)^{(\ell_i x_i+\pi_j)}}$$

$$\cdot \prod_{n=1}^{j-1} \left[\frac{\left(1+\frac{\pi_j}{x_{n+1} \ell_i}\right)^{\left(x_{n+1}+\frac{\pi_j}{\ell_i}\right)}}{\left(1+\frac{\pi_j}{x_n \ell_i}\right)^{\left(x_n+\frac{\pi_j}{\ell_i}\right)}} \frac{\left(1+\frac{\pi_j}{x_n \ell_j}\right)^{\left(x_n+\frac{\pi_j}{\ell_j}\right)}}{\left(1+\frac{\pi_j}{x_{n+1} \ell_j}\right)^{\left(x_{n+1}+\frac{\pi_j}{\ell_j}\right)}} \left(\frac{x_n}{x_{n+1}}\right)^{\left(\frac{\ell_i-\ell_j}{\ell_j \ell_i}\pi_j\right)} \right]^{\ell_n} \tag{41}$$

$$= \frac{\left(1+\frac{\pi_j}{x_i \ell_j}\right)^{(\ell_j x_i+\pi_j)}}{\left(1+\frac{\pi_j}{x_i \ell_i}\right)^{(\ell_i x_i+\pi_j)}} \prod_{n=1}^{j-1} \left[\frac{\left(x_{n+1}+\frac{\pi_j}{\ell_i}\right)^{\left(x_{n+1}+\frac{\pi_j}{\ell_i}\right)}}{\left(x_n+\frac{\pi_j}{\ell_i}\right)^{\left(x_{n+1}+\frac{\pi_n}{\ell_n}+\frac{\pi_j}{\ell_i}\right)}} \frac{\left(x_n+\frac{\pi_j}{\ell_j}\right)^{\left(x_{n+1}+\frac{\pi_n}{\ell_n}+\frac{\pi_j}{\ell_j}\right)}}{\left(x_{n+1}+\frac{\pi_j}{\ell_j}\right)^{\left(x_{n+1}+\frac{\pi_j}{\ell_j}\right)}} \right]^{\ell_n} \tag{42}$$

$$= \frac{\left(1+\frac{\pi_j}{x_i \ell_j}\right)^{(\ell_j x_i+\pi_j)}}{\left(1+\frac{\pi_j}{x_i \ell_i}\right)^{(\ell_i x_i+\pi_j)}}$$

$$\cdot \prod_{n=1}^{j-1} \left[\left(\frac{1+\frac{\pi_j}{x_n \ell_j}}{1+\frac{\pi_j}{x_n \ell_i}}\right)^{\frac{\pi_n}{\ell_n}} \frac{\left(x_{n+1}+\frac{\pi_j}{\ell_i}\right)^{\left(x_{n+1}+\frac{\pi_j}{\ell_i}\right)}}{\left(x_{n+1}+\frac{\pi_n}{\ell_n}+\frac{\pi_j}{\ell_i}\right)^{\left(x_{n+1}+\frac{\pi_j}{\ell_i}\right)}} \frac{\left(x_{n+1}+\frac{\pi_n}{\ell_n}+\frac{\pi_j}{\ell_j}\right)^{\left(x_{n+1}+\frac{\pi_j}{\ell_j}\right)}}{\left(x_{n+1}+\frac{\pi_j}{\ell_j}\right)^{\left(x_{n+1}+\frac{\pi_j}{\ell_j}\right)}} \right]^{\ell_n} \tag{43}$$

We now define the two following functions:

$$f_1(y,\delta) = (1+\delta/y)^{y+\delta} \tag{44}$$
$$f_2(y,\delta) = (1+\delta/y)^{y} \tag{45}$$

Therefore we can write (43) as:

$$\frac{\exp\Big(D_{KL}(\boldsymbol{\delta}^{\triangle}(s)||\boldsymbol{\delta}^{\bullet}(s))\Big)}{\exp\Big(D_{KL}(\boldsymbol{\delta}^{\square}(s)||\boldsymbol{\delta}^{\bullet}(s))\Big)} = \left[\frac{f_1(x_i\ell_j,\pi_j)}{f_1(x_i\ell_i,\pi_j)}\right] \prod_{n=1}^{j-1} \left[\left(\frac{1+\frac{\pi_j}{x_n \ell_j}}{1+\frac{\pi_j}{x_n \ell_i}}\right)^{\frac{\pi_n}{\ell_n}} \frac{f_2\left(x_{n+1}+\frac{\pi_j}{\ell_j},\frac{\pi_n}{\ell_n}\right)}{f_2\left(x_{n+1}+\frac{\pi_j}{\ell_i},\frac{\pi_n}{\ell_n}\right)} \right]^{\ell_n} \tag{46}$$

Noting that functions f_1 and f_2 is a strictly monotone decreasing and increasing in y, respectively, and remembering that $\ell_i > \ell_j$ implies that all terms in (46) are larger than 1, which in turns implies that $D_{KL}(\boldsymbol{\delta}^{\triangle}(s)||\boldsymbol{\delta}^{\bullet}(s)) > D_{KL}(\boldsymbol{\delta}^{\square}(s)||\boldsymbol{\delta}^{\bullet}(s))$.
We conclude the proof by showing that the value function corresponding to $\pi^{\square}$ is strictly larger than that of $\pi^{\triangle}$ and therefore $\pi^{\triangle}$ is not a uc-optimal policy

$$v^{\square}(s) = \sum_a \pi^{\square}(a|s)\widehat{q}(s,a) - \kappa D_{KL}(\boldsymbol{\delta}^{\square}(s)||\boldsymbol{\delta}^{\bullet}(s)) \tag{47}$$

$$= \sum_{\substack{a \neq a_j \\ a \neq a_i}} \pi^{\triangle}(a|s)\widehat{q}(s,a) + (\pi^{\triangle}(a_i|s) + \pi^{\triangle}(a_j|s))\widehat{q}(s,a_i) - \kappa D_{KL}(\boldsymbol{\delta}^{\square}(s)||\boldsymbol{\delta}^{\bullet}(s)) \tag{48}$$

$$\overset{(a)}{\geq} \sum_{a} \pi^{\triangle}(a|s)\widehat{q}(s,a) - \kappa D_{KL}(\boldsymbol{\delta}^{\square}(s)||\boldsymbol{\delta}^{\bullet}(s)) \tag{49}$$

$$\overset{(b)}{>} \sum_{a} \pi^{\triangle}(a|s)\widehat{q}(s,a) - \kappa D_{KL}(\boldsymbol{\delta}^{\triangle}(s)||\boldsymbol{\delta}^{\bullet}(s)) = v^{\triangle}(s) \tag{50}$$

where in (a) we used that the fact that a_j is Pareto dominated by a_i and hence $\widehat{q}(s,a_i) > \widehat{q}(s,a_j)$ and in (b) we used $D_{KL}(\boldsymbol{\delta}^{\triangle}(s)||\boldsymbol{\delta}^{\bullet}(s)) > D_{KL}(\boldsymbol{\delta}^{\square}(s)||\boldsymbol{\delta}^{\bullet}(s))$.

B Proof of Theorem 1

In this section we calculate $\pi^\star$ for a generic state s, for the sake of readability in this proof we write $\ell(s,a)$ as l_a with the understanding that the dependence on s is implicit. Further in this sections we assume 2 and that all actions are Pareto optimal. We start differentiating (5) with respect to $\pi(a|s)$:

$$\kappa^{-1}\widehat{q}(s,a) - \frac{\partial D_{KL}(\boldsymbol{\delta}^{\star}(s)||\boldsymbol{\delta}^{\bullet}(s))}{\partial \pi^{\star}(a|s)} = 0 \tag{51}$$

Now we need an expressions for the gradient of the KL divergence. Using (30) we get:

$$\frac{\partial D_{KL}(\boldsymbol{\delta}^{\star}(s)||\boldsymbol{\delta}^{\bullet}(s))}{\partial \pi^{\star}(a_j|s)} = \ell_j^{-1}\sum_{b=1}^{j}(\ell_b - \ell_{b-1})\log\left(\sum_{c=b}^{A}\pi^{\star}(c|s)\ell_c^{-1}\right) + \ell_j^{-1}\sum_{b=1}^{j}(\ell_b - \ell_{b-1}) \tag{52}$$

$$= \ell_j^{-1}\sum_{b=1}^{j}(\ell_b - \ell_{b-1})\log\left(\sum_{c=b}^{A}\pi^{\star}(c|s)\ell_c^{-1}\right) + 1 \tag{53}$$

$$= \ell_j^{-1}(\ell_j - \ell_{j-1})\log\left(\sum_{c=j}^{A}\pi^{\star}(c|s)\ell_c^{-1}\right) + \frac{\ell_{j-1}}{\ell_j}\left(\frac{\partial D_{KL}(\boldsymbol{\delta}^{\star}(s)||\boldsymbol{\delta}^{\bullet}(s))}{\partial \pi(a_{j-1}|s)} - 1\right) + 1 \tag{54}$$

$$= \ell_j^{-1}(\ell_j - \ell_{j-1})\log\left(\sum_{c=j}^{A}\pi^{\star}(c|s)\ell_c^{-1}\right) + \frac{\ell_{j-1}}{\ell_j}\frac{\partial D_{KL}(\boldsymbol{\delta}^{\star}(s)||\boldsymbol{\delta}^{\bullet}(s))}{\partial \pi^{\star}(a_{j-1}|s)} + \frac{\ell_j - \ell_{j-1}}{\ell_j} \tag{55}$$

Now we can solve for each action using the recursive form given in (55). Recall that due to the specific numbering of actions we assumed, a_A is the action whose bellman error has the greatest uncertainty. Hence, we can start solving for a_A as follows:

$$0 = \kappa^{-1}\widehat{q}(s,a) - \ell_A^{-1}(\ell_A - \ell_{A-1})\log\left(\pi^{\star}(a_A|s)\ell_A^{-1}\right) - \frac{\ell_{A-1}}{\ell_A}\frac{\partial D_{KL}(\boldsymbol{\delta}^{\star}(s)||\boldsymbol{\delta}^{\bullet}(s))}{\partial \pi^{\star}(a_{A-1}|s)} - \frac{\ell_A - \ell_{A-1}}{\ell_A} \tag{56}$$

$$= \kappa^{-1}\widehat{q}(s,a) - \ell_A^{-1}(\ell_A - \ell_{A-1})\log\left(\pi^{\star}(a_A|s)\ell_A^{-1}\right) - \kappa^{-1}\frac{\ell_{A-1}}{\ell_A}\widehat{q}(A-1,s) - \frac{\ell_A - \ell_{A-1}}{\ell_A} \tag{57}$$

$$\rightarrow \pi^{\star}(a_A|s) \propto \ell_A \exp\left[\frac{\kappa^{-1}}{\ell_A - \ell_{A-1}}\left(\ell_A q(A,s) - \ell_{A-1}q(A-1,s)\right)\right] \tag{58}$$

Following the same procedure we can solve for $\pi^{\star}(a_j|s)$.

$$0 = \kappa^{-1}\widehat{q}(j,s) - \frac{(\ell_j - \ell_{j-1})}{\ell_j}\log\left(\sum_{c=j}^{A}\frac{\pi^{\star}(c|s)}{\ell_c}\right) - \frac{\ell_{j-1}}{\ell_j}\frac{\partial D_{KL}(\boldsymbol{\delta}^{\star}(s)||\boldsymbol{\delta}^{\bullet}(s))}{\partial \pi^{\star}(a_{j-1}|s)} - \frac{\ell_j - \ell_{j-1}}{\ell_j} \tag{59}$$

$$= \kappa^{-1}\widehat{q}(j,s) - \ell_j^{-1}(\ell_j - \ell_{j-1})\log\left(\sum_{c=j}^{A}\pi^{\star}(c|s)\ell_c^{-1}\right) - \kappa^{-1}\frac{\ell_{j-1}}{\ell_j}\widehat{q}(j-1,s) - \frac{\ell_j - \ell_{j-1}}{\ell_j} \tag{60}$$

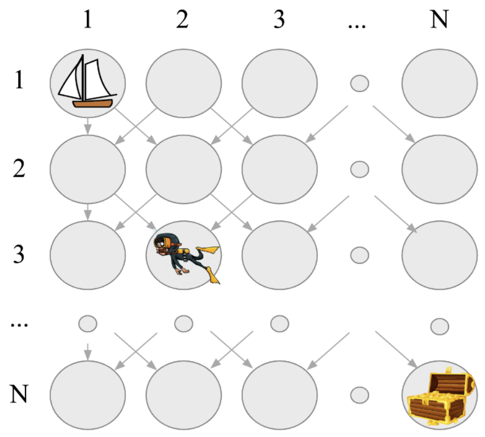


Figure 2: Image credit goes to [23].

$$\rightarrow \pi^\star(a_j|s) = \ell_j \exp\left[\frac{\ell_j\widehat{q}(a_j,s) - \ell_{j-1}\widehat{q}(a_{j-1},s)}{\kappa(\ell_j - \ell_{j-1})}\right]\exp(-1) - \ell_j \sum_{c=j+1}^{A} \pi^\star(c|s)\ell_c^{-1} \tag{61}$$

Unwinding one step of the recursion we get:

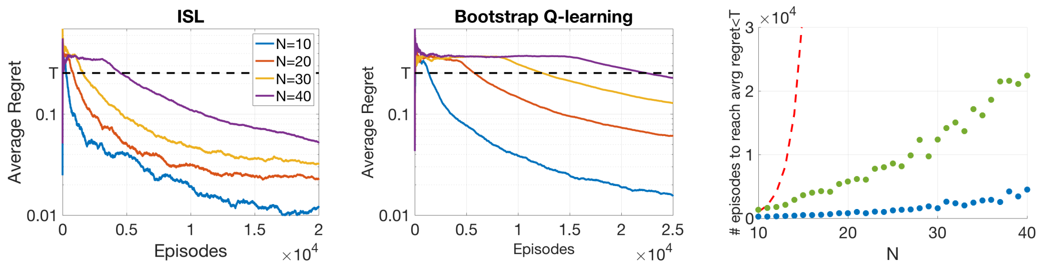
$$\pi^\star(a_j|s) = \frac{\ell_j}{N(\pi^\star)}\exp\left[\frac{\ell_j\widehat{q}(a_j,s) - \ell_{j-1}\widehat{q}(a_{j-1},s)}{\kappa(\ell_j - \ell_{j-1})}\right] - \frac{\ell_j}{N(\pi^\star)}\exp\left[\frac{l_{j+1}\widehat{q}(a_{j+1},s) - \ell_j\widehat{q}(a_j,s)}{\kappa(l_{j+1} - \ell_j)}\right] \tag{62a}$$

$$N(\pi^\star) = \sum_{j=1}^{A}(\ell_j - \ell_{j-1})\exp\left[\frac{\kappa^{-1}}{\ell_j - \ell_{j-1}}\left(\ell_j\widehat{q}(a_j,s) - \ell_{j-1}\widehat{q}(a_{j-1},s)\right)\right] \tag{62b}$$

where $N(\pi^\star)$ is the normalization constant. Note that expression (62) is strictly positive for all $a \in \mathcal{E}_s$

C Deep sea implementation details

The Deep Sea game was introduced in [22] and the parameters of our implementation follow the original setup, which we reproduce here for the readers convenience. The state space of the game is composed of an $N \times N$ grid (see figure 2). The player always starts at the top-left corner and descends one row per timestep. At every timestep the agent can choose between 2 actions, one of which moves to the left and the other to the right. Every time the agent moves to the left the agent receives $r_t = 0$, and if it moves to the right it receives $r_t = -0.01/N$. If the agent arrives at the state in the bottom-right corner it receives an additional reward $r_t = 1$. Note that the optimal policy is to always move right for which the cumulative reward is given by $\sum_t r_t = 0.99$. In the stochastic version of the game every reward is subject to gaussian noise with variance equal to 0.1, and every time an agent chooses to move to the right it does so with a probability equal to $1 - 1/N$. Like we clarified in the main body of the paper, we implemented ISL and Bootstrap Q-learning in tabular form. We implemented tabular forms as a first experiment to be able to characterize the dynamics of the learning algorithms without being affected by the intricacies of training neural networks. In the extended version of this paper we present the 'deep' evaluation of ISL in the full bsuite [23]. The parameters of the ISL implementation for the deterministic version of the game were set to $\alpha = 1$, $\kappa = 1$, $\eta = 0$, $\mu_\omega = 1.2$, $\mu_\nu = 1$ and $\mu_\theta = 1$, further the size of the replay buffer was set proportional to the size of the state space. For the Bootrsap Q-learning implementation the step size was set to $\mu_q = 1$ and we used 3 q-tables to make sure that both algorithms had the same memory requirements and therefore set a fair comparison. For the stochastic version of the game, the ISL parameters were $\alpha = 1$, $\kappa = 0.75$, $\eta = 1$, $\mu_\omega = 0.01$, $\mu_\nu = 0.01$ and $\mu_\theta = 0.03$. And for Bootstrap Q-learning the step size was set to $\mu_q = 0.002$. In figure 3 we show the results for the stochastic version of the game. Note that the same conclusions apply as in the deterministic case; the amount of episodes to learn the optimal policy seems to be $\mathcal{O}(N)$ for both algorithms but the constant corresponding to ISL is lower.[4]



(a) (b) (c)

Figure 3: The legend for (b) is the same as (a). In (c) the red dashed line plots 2^N, the blue and green dots correspond to ISL and Bootstrap Q-learning, respectively.

[4]Code is available at the author's github, https://github.com/lcassano/ISL-deep-sea-matlab.