Implicitly Regularized RL with Implicit Q-Values

Anonymous Author(s)
Affiliation
Address
email

Abstract

The Q-function is a central quantity in many Reinforcement Learning (RL) algorithms for which RL agents behave following a (soft)-greedy policy w.r.t. to Q. It is a powerful tool that allows action selection without a model of the environment and even without explicitly modeling the policy. Yet, this scheme can only be used in discrete action tasks, with small numbers of actions, as the softmax cannot be computed exactly otherwise. Especially the usage of function approximation, to deal with continuous action spaces in modern actor-critic architectures, intrinsically prevents the exact computation of a softmax. We propose to alleviate this issue by parametrizing the Q-function *implicitly*, as the sum of a log-policy and of a value function. We use the resulting parametrization to derive a practical off-policy deep RL algorithm, suitable for large action spaces, and that enforces the softmax relation between the policy and the Q-value. We provide a theoretical analysis of our algorithm: from an Approximate Dynamic Programming perspective, we show its equivalence to a regularized version of value iteration, accounting for both entropy and Kullback-Leibler regularization, and that enjoys beneficial error propagation results. We then evaluate our algorithm on classic control tasks, where its results compete with state-of-the-art methods.

1 Introduction

A large body of reinforcement learning (RL) algorithms, based on approximate dynamic programming (ADP) [Bertsekas and Tsitsiklis, 1996, Scherrer et al., 2015], operate in two steps: A greedy step, where the algorithm learns a policy that maximizes a Q-function, and an evaluation step, that (partially) updates the Q-values towards the Q-values of the policy. A common improvement to these techniques is to use regularization, that prevents the new updated policy from being too different from the previous one, or from a fixed "prior" policy. For example, Kullback-Leibler (KL) regularization keeps the policy close to the previous iterate [Vieillard et al., 2020a], while entropy regularization keeps the policy close to the uniform one [Haarnoja et al., 2018a]. Entropy regularization, often used in this context [Ziebart, 2010], modifies both the greedy step and the evaluation step so that the policy jointly maximizes its expected return and its entropy. In this framework, the solution to the policy optimization step is simply a softmax of the Q-values over the actions. In small discrete action spaces, the softmax can be computed exactly: one only needs to define a *critic* algorithm, with a single loss that optimizes a Q-function. However, in large multi-dimensional – or even continuous – action spaces, one needs to estimate it. This estimation is usually done by adding an *actor* loss, that optimizes a policy to fit this softmax. It results in an *actor-critic* algorithm, with two losses that are optimized simultaneously [Haarnoja et al., 2018a]. This additional optimization step introduces supplementary errors to the ones already created by the approximation in the evaluation step.

To remove these extraneous approximations, we introduce the Implicit Q-Functions (IQ) algorithm, that deviates from classic actor-critics, as it optimizes a policy and a value in a single loss. The core idea is to implicitly represent the Q-function as the sum of a value function and a log-policy.

This representation ensures that the policy is an *exact* softmax of the Q-value, *despite the use of any approximation scheme*. We use this to design a practical model-free deep RL algorithm that optimizes with a single loss a policy network and a value network, built on this implicit representation of a Q-value. To better understand it, we abstract this algorithm to an ADP scheme, IQ-DP, and use this point of view to provide a detailed theoretical analysis. It relies on a key observation, that shows an equivalence between IQ-DP and a specific form of regularized Value Iteration (VI). This equivalence explains the role of the components of IQ: namely, IQ performs entropy and KL regularization. It also allows us to derive strong performance bounds for IQ-DP. In particular, we show that the errors made when following IQ-DP are compensated along iterations.

Parametrizing the Q-value as a sum of a log-policy and a value is reminiscent of the dueling architecture [Wang et al., 2016], that factorizes the Q-value as the sum of an advantage and a value. In fact, we show that it is a limiting case of IQ in a discrete actions setting. This link highlights the role of the policy in IQ, which calls for a discussion on the necessary parametrization of the policy.

Finally, we empirically validate IQ. We evaluate our method on several classic continuous control benchmarks: locomotion tasks from Openai Gym [Brockman et al., 2016], and hand manipulation tasks from the Adroit environment [Rajeswaran et al., 2017]. On these environments, IQ reaches performances competitive with state-of-the-art actor critic methods.

2 Implicit Q-value parametrization

We consider the standard Reinforcement Learning (RL) setting, formalized as a Markov Decision Process (MDP). An MDP is a tuple $\{\mathcal{S}, \mathcal{A}, P, r, \gamma\}$. $\mathcal{S}$ and $\mathcal{A}$ are the finite state and action spaces[1], $\gamma \in [0, 1)$ is the discount factor and $r : \mathcal{S} \times \mathcal{A} \to [-R_{max}, R_{max}]$ is the bounded reward function. Write Δ_X the simplex over the finite set X. The dynamics of an MDP are defined by a Markovian transition kernel $P \in \Delta_{\mathcal{S}}^{\mathcal{S}\times\mathcal{A}}$, where $P(s'|s, a)$ is the probability of transitioning to state s' after taking action a in s. An RL agent acts through a stochastic policy $\pi \in \Delta_{\mathcal{A}}^{\mathcal{S}}$, a mapping from states to distribution over actions. The quality of a policy is quantified by the value function, $V_\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|s_0 = s]$. The Q-function is a useful extension, which notably allows choosing a (soft)-greedy action in a model-free setting, $Q_\pi(s, a) = r(s, a) + \mathbb{E}_{s'|s,a}[V_\pi(s')]$. An optimal policy is one that achieve the highest expected return, $\pi_* = \operatorname{argmax}_\pi V_\pi$.

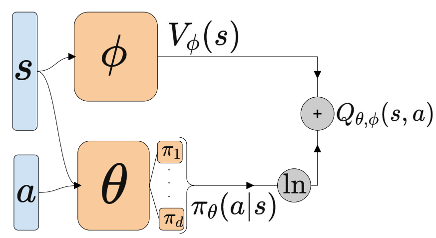

Figure 1: view of the IQ parametrization.

A classic way to design practical algorithms beyond the tabular setting is to adopt the Actor-Critic perspective. In this framework, an RL agent parametrizes a policy π_θ and a Q-value Q_ψ with function approximation, usually through the use of neural networks, and aims at estimating an optimal policy. The policy and the Q-function are then updated by minimizing two losses: the actor loss corresponds to the greedy step, and the critic loss to the evaluation step. The weights of the policy and Q-value networks are regularly frozen into *target* weights $\bar{\psi}$ and $\bar{\theta}$. With entropy regularization, the greedy step amounts to finding the policy that maximizes $\mathbb{E}_{s\sim\mathcal{S},a\sim\pi_\theta}[Q_{\bar{\psi}}(s, a) + \tau \ln \pi_\theta(a|s)]$ (maximize the Q-value with stochastic enough policy). The solution to this problem is simply $\pi_\theta(\cdot|s) = \operatorname{softmax}(Q_{\bar{\psi}}(s, \cdot)/\tau)$, which is the result of the greedy step of regularized Value Iteration (VI) [Geist et al., 2019] and, for example, how the optimization step of Soft Actor-Critic [Haarnoja et al., 2018a, SAC] is built. In a setting where the action space is discrete and small, it amounts to a simple softmax computation. However, on more complex action spaces (continuous, and/or with a higher number of dimensions: as a reference, the Humanoid-v2 environment from Openai Gym [Brockman et al., 2016] has an action space of dimension 17), it becomes prohibitive to use the exact solution. In this case, the common practice is to resort to a approximation with a parametric distribution model. In many actor critic algorithms (SAC, TD3[Fujimoto et al., 2018], ...), the policy is modelled as a Gaussian distribution over actions. It introduces approximation errors, resulting from the partial optimization process of the critic, and inductive bias, as a Gaussian policy cannot represent an arbitrary softmax distribution. We now turn to the description of our core contribution: the Implicit Q-value (IQ) algorithm, introduced to mitigate this discrepancy.

[1]We restrict to finite spaces for the sake of analysis, but our approach applies to continuous spaces.

IQ implicitly parametrizes a Q-value via an explicit parametrization of a policy and a value, as visualized in Fig. 1. Precisely, from a policy network π_θ and a value network V_ϕ, we define our implicit Q-value as

$$Q_{\theta,\phi}(s,a) = \tau \ln \pi_\theta(a|s) + V_\phi(s). \tag{1}$$

Since π_θ is constrained to be a distribution over the actions, we have by construction that $\pi_\theta(a|s) = \mathrm{softmax}(Q_{\theta,\phi}/\tau)$, the solution of the regularized greedy step (see Appx. A.1 for a detailed proof). Hence, the consequence of using such a parametrization is that the greedy step is performed exactly, even in the function approximation regime. Compared to the classic actor-critic setting, it thus gets rid of the errors created by the actor. Note that calling V_ϕ a value makes sense, since following the same reasoning we have that $V_\phi(s) = \tau \ln \sum_{a'} \exp(Q_{\theta,\phi}(s,a')/\tau)$, a soft version of the value. With this parametrization in mind, one could derive a deep RL algorithm from any value-based loss using entropy regularization. We conserve the fixed-point approach of the standard actor-critic framework, θ and ϕ are regularly copied to $\bar{\theta}$ and $\bar{\phi}$, and we design an off-policy algorithm, working on a replay buffer of transitions $(s_t, a_t, r_t, s_{st+1})$ collected during training. Consider two hyperparameters, $\tau \in (0,\infty)$ and $\alpha \in (0,1)$ that we will show in Sec. 3 control two forms of regularization.

The policy and value are optimized jointly by minimizing the loss

$$\mathcal{L}_{\mathrm{IQ}}(\theta,\phi) = \hat{\mathbb{E}}\left[\left(r_t + \alpha\tau \ln \pi_{\bar{\theta}}(a_t|s_t) + \gamma V_{\bar{\phi}}(s_{t+1}) - \tau \ln \pi_\theta(a_t|s_t) - V_\phi(s_t)\right)^2\right], \tag{2}$$

where $\hat{\mathbb{E}}$ denote the empirical expected value over a dataset of transitions. IQ consists then in a single loss that optimizes jointly a policy and a value. This brings a notable remark on the role of Q-functions in RL. Indeed, Q-learning was introduced by Watkins and Dayan [1992] – among other reasons – to make greediness possible without a model (using a value only, one needs to maximize it over all possible successive states, which requires knowing the transition model), and consequently derive practical, model-free RL algorithms. Here however, IQ illustrates how, with the help of regularization, one can derive a model-free algorithm that does not rely on an explicit Q-value.

3 Analysis

In this section, we explain the workings of the IQ algorithm defined by Eq. (2) and detail the influence of its hyperparameters. We abstract IQ into an ADP framework, and show that, from that perspective, it is equivalent to a Mirror Descent VI (MD-VI) scheme [Geist et al., 2019], with both entropy and KL regularization. Let us first introduce some useful notations. We make use of the actions partial dot-product notation: for $u, v \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$, we define $\langle u, v\rangle = \left(\sum_{a\in\mathcal{A}} u(s,a)v(s,a)\right)_s \in \mathbb{R}^{\mathcal{S}}$. For any $V \in \mathbb{R}^{\mathcal{S}}$, we have for any $(s,a) \in \mathcal{S}\times\mathcal{A}$ $PV(s,a) = \sum_{s'} P(s'|s,a)V(s')$. We will define regularized algorithms, using the entropy of a policy, $H(\pi) = -\langle \pi, \ln \pi\rangle$, and the KL divergence between two policies, $\mathrm{KL}(\pi||\mu) = \langle \pi, \ln\pi - \ln\mu\rangle$. The Q-value of a policy is the unique fixed point of its Bellman operator T_π defined for any $Q \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ as $T_\pi Q = r + \gamma P\langle \pi, Q\rangle$. We denote $Q_* = Q_{\pi_*}$ the optimal Q-value (the Q-value of the optimal policy). When the MDP is entropy-regularized with a temperature τ, a policy π admits a *regularized* Q-value Q^τ_π, the fixed point of the regularized bellman operator $T^\tau_\pi Q = r + \gamma P\langle \pi, Q - \tau\ln\pi\rangle$. A regularized MDP admits an optimal *regularized* policy π^τ_* and a unique optimal *regularized* Q-value Q^τ_* [Geist et al., 2019].

3.1 Ideal case

First, let us look at the ideal case, *i.e.* when $\mathcal{L}_{\mathrm{IQ}}$ is exactly minimized at each iteration (tabular representation, dataset covering the whole state-action space, expectation rather than sampling for transitions). In this context, IQ can be understood as a Dynamic Programming (DP) scheme that iterates on a policy π_{k+1} and a value V_k. They are respectively equivalent to the target networks $\pi_{\bar{\theta}}$ and $V_{\bar{\phi}}$, while the next iterate (π_{k+2}, V_{k+1}) matches the solution (π_θ, V_ϕ) of the optimization problem in Eq. (2). We call the scheme IQ-DP(α,τ) and one iteration is defined by choosing (π_{k+2}, V_{k+1}) such that the squared term in Eq. (2) is 0, that is

$$\tau \ln \pi_{k+2} + V_{k+1} = r + \alpha\tau \ln \pi_{k+1} + \gamma P V_k. \tag{3}$$

This equation is well-defined, due to the underlying constraint that $\pi_{k+2} \in \Delta^{\mathcal{S}}_{\mathcal{A}}$ (the policy must be a distribution over actions), that is $\sum_{a\in\mathcal{A}} \pi(a|s) = 1$ for all $s \in \mathcal{S}$. The basis for our discussion will be the equivalence of this scheme to a version of regularized VI. Indeed, we have the following result, proved in Appendix A.3.

Theorem 1. *For any $k \geq 1$, let (π_{k+2}, V_{k+1}) be the solution of IQ-DP(α, τ) at step k. We have that*

$$\begin{cases} \pi_{k+2} = \operatorname{argmax}\langle \pi, r + \gamma P V_k \rangle + (1-\alpha)\tau \mathcal{H}(\pi) - \alpha\tau \operatorname{KL}(\pi || \pi_{k+1}) \\ V_{k+1} = \langle \pi_{k+2}, r + \gamma P V_k \rangle + (1-\alpha)\tau \mathcal{H}(\pi_{k+2}) - \alpha\tau \operatorname{KL}(\pi_{k+2} || \pi_{k+1}) \end{cases}$$

so IQ-DP(α, τ) produces the same sequence of policies as a value-based version of Mirror Descent VI, MD-VI$(\alpha\tau, (1-\alpha)\tau)$ [Vieillard et al., 2020a].

Discussion. The previous results sheds a first light on the nature of the IQ method. Essentially, IQ-DP is a parametrization of a VI scheme regularized with both entropy and KL divergence, MD-VI$(\alpha\tau, (1-\alpha)\tau)$. This first highlights the role of the hyperparameters, as its shows the interaction between the two forms of regularization. The value of α balances between those two: with $\alpha = 0$, IQ-DP reduces to a classic VI regularized with entropy; with $\alpha = 1$ only the KL regularization will be taken into account. The value of τ then controls the amplitude of this regularization. In particular, in the limit $\alpha = 0, \tau \to 0$, we recover the standard VI algorithm. This results also justifies the soundness of IQ-DP. Indeed, this MD-VI scheme is known to converge to $\pi_*^{(1-\alpha)\tau}$ the optimal policy of the regularized MDP [Vieillard et al., 2020a, Thm. 2] and this results readily applies to IQ[2]. Another consequence is that it links IQ to Advantage Learning (AL) [Bellemare et al., 2016]. Indeed, AL is a limiting case of MD-VI when $\alpha > 0$ and $\tau \to 0$ [Vieillard et al., 2020b]. Therefore, IQ also generalizes AL, and the α parameter can be interpreted as the advantage coefficient. Finally, a key observation is that IQ performs KL regularization implicitly, the way it was introduced by Munchausen RL [Vieillard et al., 2020b], by augmenting the reward with the $\alpha\tau \ln \pi_{k+1}$ term (Eq. (3)). This observation will have implications discussed next.

3.2 Error propagation result

Now, we are interested in understanding how errors introduced by the function approximation used propagate along iterations. At iteration k of IQ, denote π_{k+1} and V_k the target networks. In the approximate setting, we do not solve Eq. (3), but instead, we minimize $\mathcal{L}(\theta, \phi)$ with stochastic gradient descent. This means that π_{k+2} and V_{k+1} are the result of this optimization, and thus the next target networks. The optimization process introduces errors, that come from many sources: partial optimization, function approximation (policy and value are approximated with neural networks), finite data, etc. We study the impact of these errors on the distance between the optimal Q-value of the MDP and the regularized Q-value of the current policy used by IQ, $Q_{\pi_{k+1}}^{(1-\alpha)\tau}$. We insist right away that $Q_{\pi_{k+1}}^{(1-\alpha)\tau}$ is not the learned, implicit Q-value, but the actual Q-value of the policy computed by IQ in the regularized MDP. We have the following result concerning the error propagation.

Theorem 2. *Write π_{k+1} and V_k the k^{th} update of respectively the target policy and value networks. Consider the error at step k, $\epsilon_k \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$, as the difference between the ideal and the actual updates of IQ. Formally, we define the error as, for all $k \geq 1$,*

$$\epsilon_k = \tau \ln \pi_{k+2} + V_{k+1} - (r + \alpha\tau \ln \pi_{k+1} + \gamma P V_k),$$

and the moving average of the errors as

$$E_k = (1-\alpha) \sum_{j=1}^{k} \alpha^{k-j} \epsilon_j.$$

We have the following results for two different cases depending on the value of α. Note that when $\alpha < 1$, we bound the distance to regularized optimal Q-value.

1. *General case: $0 < \alpha < 1$ and $\tau > 0$, entropy and KL regularization together:*

$$\|Q_*^{(1-\alpha)\tau} - Q_{\pi_k}^{(1-\alpha)\tau}\|_\infty \leq \frac{2}{(1-\gamma)^2} \left((1-\gamma) \sum_{j=1}^{k} \gamma^{k-j} \|E_j\|_\infty \right) + o\left(\frac{1}{k}\right).$$

2. *Specific case $\alpha = 1$, $\tau > 0$, use of KL regularization alone:*

$$\|Q_* - Q_{\pi_k}\|_\infty \leq \frac{2}{1-\gamma} \left\| \frac{1}{k} \sum_{j=1}^{k} \epsilon_j \right\|_\infty + O\left(\frac{1}{k}\right).$$

[2] Vieillard et al. [2020a] show this for Q-functions, but it can straightforwardly be extended to value functions.

Sketch of proof. The full proof is provided in Appendix A.4. We build upon the connection we established between IQ-DP and a VI scheme regularized by both KL and entropy in Thm. 1. By injecting the proposed representation into the classic MD-VI scheme, we can build upon the analysis of Vieillard et al. [2020a, Thm. 1 and 2] to provide these results. □

Impact of KL regularization. The KL regularization term, and specifically in the MD-VI framework, is discussed extensively by Vieillard et al. [2020a], and we refer to them for in-depth analysis of the subject. We recall here the main interests of KL regularization, as illustrated by the bounds of Thm 2. In the second case, where it is the clearest (only KL is used), we observe a beneficial property of KL regularization: Averaging of errors. Indeed, in a classic non-regularized VI scheme [Scherrer et al., 2015], the error $\|Q_* - Q_{\pi_\theta}\|$ would depend on a moving average of the norms of the errors $(1-\gamma)\sum_{j=1}^{k}\gamma^{k-j}\|\epsilon_k\|_\infty$, while with the KL it depends on the norm of the average of the errors $(1/k)\|\sum_{j=1}^{k}\epsilon_k\|$. In a simplified case where the errors would be i.i.d. and zero mean, this would allow convergence of approximate MD-VI, but not of approximate VI. In the case $\alpha < 1$, where we introduce entropy regularization, the impact is less obvious, but we still transform a sum of norm of errors into a sum of moving average of errors, which can help by reducing the underlying variance.

Link to Munchausen RL. As stated in the sketched proof, Thm. 2 is a consequence of [Vieillard et al., 2020a, Thm. 1 and 2]. A crucial limitation of this work is that the analysis only applies when no errors are made in the greedy step. This is possible in a relatively simple setting, with tabular representation, or with a linear parametrization of the Q-function. However, in the general case with function approximation, exactly solving the optimization problem regularized by KL is not immediately possible: the solution of the greedy step of MD-VI$(\alpha\tau, (1-\alpha)\tau)$ is $\pi_{k+2} \propto \exp(Q_{k+1}/\tau)\pi_k^\alpha$ (where $Q_{k+1} = r + \gamma P V_k$), so computing it exactly would require remembering every π_j during the procedure, which is not feasible in practice. A workaround to this issue was introduced by Vieillard et al. [2020b] as Munchausen RL: the idea is to augment the reward by the log-policy, to implicitly define a KL regularization term, while reducing the greedy step to a softmax. As mentioned before, in small discrete action spaces, this allows to compute the greedy step exactly, but it is not the case in multidimensional or continuous action spaces, and thus Munchausen RL loses its interest in such domains. With IQ, we utilize the Munchausen idea to implicitly define the KL regularization; but with our parametrization, the exactness of the greedy step holds even for complex action spaces: recall that the parametrization defined in Eq. (1) enforces that the policy is a softmax of the (implicit) Q-value. Thus, IQ can be seen as an extension of Munchausen RL to multidimensional and continuous action spaces.

3.3 Link to the dueling architecture

Dueling Networks (DN) were introduced as a variation of the seminal Deep Q-Networks (DQN, Mnih et al. [2015]), and has been empirically proven to be efficient (for example by Hessel et al. [2018]). The idea is to represent the Q-value as the sum of a value and an advantage. In this setting, we work with a notion of advantage defined over Q-functions (as opposed to defining the advantage as a function of a policy). For any $Q \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$, its advantage A_Q is defined $A_Q(s,a) = Q(s,a) - \max_{a'\in\mathcal{A}} Q(s,a')$. The advantage encodes a sub-optimality constraint: it has negative values and its maximum over actions (the action maximizing the Q-value) is 0. Wang et al. [2016] propose to learn a Q-value by defining and advantage network F_Θ and a value network V_Φ, which in turn define a Q-value $Q_{\Theta,\Phi}$ as

$$Q_{\Theta,\Phi}(s,a) = \underbrace{F_\Theta(s,a) - \max_{a'\in\mathcal{A}} F_\Theta(s,a')}_{\text{advantage}} + V_\Phi(s).$$

Subtracting the maximum over the actions ensures that the advantage network indeed represents an advantage. Note that dueling DQN was designed for discrete action settings, where computing the maximum over actions is not an issue.

In IQ, we need a policy network that represents a distribution over the actions. There are several practical ways to represent the policy, that are discussed in Sec 4. For the sake of simplicity, let us for now assume that we are in a mono-dimensional discrete action space, and that we use a common scaled softmax representation. Specifically, our policy is represented by a neural network (eg. fully connected) F_θ, that maps state-action pairs to logits $F_\theta(s,a)$. The policy is then defined as $\pi_\theta(\cdot|s) = \mathrm{softmax}(F_\theta(s,\cdot)/\tau)$. Directly from the definition of the softmax, we observe that $\tau \ln \pi_\theta(a|s) =$

$F_\theta(s,a) - \tau \ln \sum_{a' \in \mathcal{A}} \exp(F_\theta(s,a')/\tau)$. The second term is a classic scaled logsumexp over the actions, a soft version of the maximum: when $\tau \to 0$, we have that $\tau \ln \sum_a' \exp(F(s,a')/\tau) \to \max_{a'} F(s,a')$. Within the IQ parametrization, we have

$$Q_{\theta,\phi} = \underbrace{F_\theta(s,a) - \tau \ln \sum_{a' \in \mathcal{A}} \exp(F(s,a')/\tau)}_{\text{soft-advantage}} + V_\phi(s),$$

which makes a clear link between IQ and DN. In this case (scaled softmax representation), the IQ parametrization generalizes the dueling architecture, retrieved when $\tau \to 0$ (and with an additional AL term whenever $\alpha > 0$, see Sec. 3). In practice, Wang et al. [2016] use a different parametrization of the advantage, replacing the maximum by a mean, defining $Q_{\Theta,\Phi}(s,a) = A_\Theta(s,a) - |\mathcal{A}|^{-1} \sum_{a' \in \mathcal{A}} A_\Theta(s,a') + V_\Phi(s)$. We could use a similar trick and replace the logsumexp by a mean in our policy parametrization, but in our case this did not prove to be efficient in practice.

We showed how the log-policy represents a soft version of the advantage. While this makes its role in the learning procedure clearer, it also raises questions about what sort of representation would be the most suited for optimization.

4 Practical considerations

We now describe key practical issues encountered when choosing a policy representation. The main one comes from the delegation of the representation power of the algorithm to the policy network. In a standard actor-critic algorithm – take SAC for example, where the policy is parametrized as a Gaussian distribution – the goal of the policy is mainly to track the maximizing action of the Q-value. Thus, estimation errors can cause the policy to choose sub-optimal actions, but the inductive bias caused by the Gaussian representation may not be a huge issue in practice, as long as the mean of the Gaussian policy is not too far from the maximizing action. In other words, the representation capacity of an algorithm such as SAC lies mainly in the representation capacity of its Q-network.

In IQ, we have a parametrization of the policy that enforces it to be a softmax of an implicit Q-value. By doing this, we trade in estimation error – our greedy step is exact by construction – for representation power. More precisely, as the Q-value is not parametrized explicitly, but through the policy, the representation power of IQ is in its policy network, and a "simple" representation might not be enough anymore. For example, if we parameterized the policy as a Gaussian, this would amount to parametrize an advantage as a quadratic function of the action: this would drastically limit what the IQ could represent.

Multicategorical policies. To address this issue, we turn to other, richer, distribution representations. In practice, we consider a multi-categorical discrete softmax distribution. Precisely, we are in the context of a multi-dimensional action space $\mathcal{A}$ of dimension d, each dimension being a bounded interval. We discretize each dimension of the space uniformly in n values δ_j, for $0 \leq j \leq n-1$. It effectively defines a discrete action space $\mathcal{A}' = \bigtimes_{j=1}^d \mathcal{A}_j$, with $\mathcal{A}_j = \{\delta_0, \dots \delta_{n-1}\}$. A multidimensional action is a vector $a \in \mathcal{A}'$, and we denote a^j the j^{th} component of the action a. Assuming independence between actions conditioned on states, a policy π_θ can be factorized as the product of d marginal mono-dimensional policies $\pi_\theta(a|s) = \prod_{j=1}^d \pi_\theta^j(a^j|s)$. We represent each policy as the softmax of the output of a neural network F_θ^j, an thus we get the full representation

$$\pi_\theta(a|s) = \prod_{j=1}^d \text{softmax}\left(F_\theta^j(\cdot|s)\right)(a^j).$$

The F_θ^j functions can be represented as neural networks with a shared core, which only differ in the last layer. This type of multicategorical policy can represent any distribution (with n high enough) that does not encompass a dependency between the dimensions. The independence assumption is quite strong, and does not hold in general. From an advantage point of view, it assumes that the soft-advantage (*i.e.* the log-policy) can be linearly decomposed along the actions. While this somehow limits the advantage representation, it is a much weaker constraint than paramterizing the advantage as a quadratic function of the action (which would be the case with a Gaussian policy). In practice, these types of multicategorical policies have been experimented [Akkaya et al., 2019, Tang and Agrawal, 2020], and have proven to be efficient on continuous control tasks.

Even richer policy classes can be explored. To account for dependency between dimensions, one could envision auto-regressive multicategorical representations, used for example to parametrize a Q-value by Metz et al. [2017]. Another approach is to use richer continuous distributions, such as normalizing flows [Rezende and Mohamed, 2015, Ward et al., 2019]. In this work, we restrict ourselves to the multicategorical setting, which is sufficient to get satisfying results (Sec. 6), and we leave the other options for future work.

5 Related work

Similar parametrizations. Other algorithms make use of a similar parametrization. First, Path Consistency Learning (PCL, [Nachum et al., 2017]) also parametrize the Q-value as a sum of a log-policy and a value. Trust-PCL [Nachum et al., 2018], builds on PCL by adding a trust region constraint on the policy update, similar to our KL regularization term. A key difference with IQ is that (Trust-)PCL is a residual algorithm, while IQ works around a fixed-point scheme. Shortly, Trust-PCL can be seen as a version of IQ without the target value network $V_{\bar{\phi}}$. These entropy-regularized residual approaches are derived from the softmax temporal consistency principle, which allows to consider extensions to a specific form of multi-step learning (strongly relying on the residual aspect), but they also come with drawbacks, such as introducing a bias in the optimization when the environment is stochastic [Geist et al., 2017]. Second, Quinoa [Degrave et al., 2018] uses a similar loss to Trust-PCL and IQ (without reference to the former Trust-PCL), but do not propose any analysis, and is evaluated only on a few tasks. Third, Normalized Advantage Function (NAF, Gu et al. [2016]) is designed with similar principles. In NAF, a Q-value is parametrized as a value and and an advantage, the former being quadratic on the action. It matches the special case of IQ with a Gaussian policy, where we recover this quadratic parametrization.

Regularization. Entropy and KL regularization are used by many other RL algorithms. Notably, from a dynamic programming perspective, IQ-DP(0, τ) performs the same update as SAC – an entropy regularized VI. This equivalence is however not true in the function approximation regime. Due to the empirical success of SAC and its link to IQ, it will be used as our main baseline on continuous control tasks. Other algorithms also use KL regularization, notably Maximum a posteriori Policy Optimization (MPO, Abdolmaleki et al. [2018]). We refer to Vieillard et al. [2020a] for an exhaustive review of algorithms encompassed within the MD-VI framework.

6 Experiments

Environments and metrics. We evaluate IQ first on the Mujoco environment from OpenAI Gym [Brockman et al., 2016]. It consists of 5 locomotion tasks, with action spaces ranging from 3 (Hopper-v2) to 17 dimensions (Humanoid-v2). We use a rather long time horizon setting, evaluating our algorithm on 20M steps on each environments. We also provide result on the Adroit manipulation dataset [Rajeswaran et al., 2017], with a similar setting of 20M environment steps. Adroit is a collection of 4 hand manipulation tasks. This environment is often use in an offline RL setting, but here we use it only as a direct RL benchmark. Out of these 4 tasks, we only consider 3 of them: We could not find any working algorithm (baseline or new) on the "relocate" task. To summarize the performance of an algorithm, we report the baseline-normalized score along iterations: It normalizes the score so that 0% corresponds to a random score, and 100% to a given baseline. It is defined for one task as score $= \frac{\text{score}_{\text{algorithm}} - \text{score}_{\text{random}}}{\text{score}_{\text{baseline}} - \text{score}_{\text{random}}}$, where the baseline is the best version of SAC on Mujoco and Adroit after 20M steps. We then report aggregated results, showing the mean and median of these normalized scores along the tasks. Each score is reported as the average over 20 random seeds. For each experiment, the corresponding standard deviation is reported in B.3

IQ algorithms. We implement IQ with the Acme [Hoffman et al., 2020] codebase. It defines two deep neural networks, a policy network π_θ and a value network V_ϕ. IQ interacts with the environment through π_θ, and collect transitions that are stored in a FIFO replay buffer. At each interaction, IQ updates θ and ϕ by performing a step of stochastic gradient descent with Adam [Kingma and Ba, 2015] on $\mathcal{L}_{\text{IQ}}$ (Eq. (2)). During each step, IQ updates a copy of the weights θ, $\bar{\theta}$, with a smooth

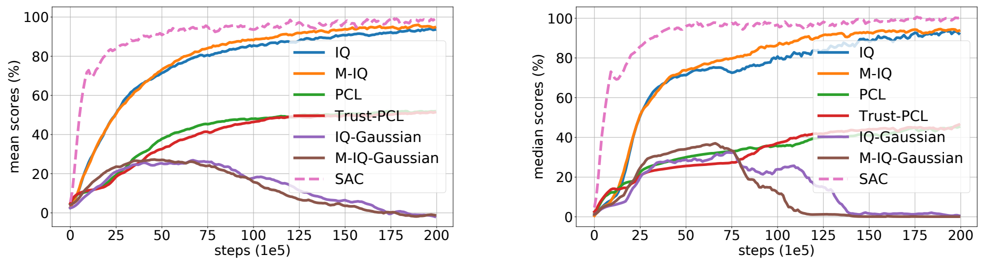

Figure 2: SAC-normalized scores on Gym. **Left**: Mean scores. **Right:** Median scores.

update $\bar{\theta} \leftarrow (1-\lambda)\bar{\theta} + \lambda\theta$, with $\lambda \in (0,1)$. It tracks a similar copy $\bar{\phi}$ of ϕ. We keep almost all common hyperparameters (networks architecture, λ, etc.) the same as our main baseline, SAC. We only adjust the learning rate for two tasks, Humanoid and Walker, where we used a lower value: we found that IQ benefits from this, while for SAC we did not observe any improvement (we provide more details and complete results in Appx. B.3). Our value network has the same architecture as the SAC Q-networks except that the input size is only the state size (as it does not depend on the action). The policy network has the same architecture as the SAC policy network, and differs only by its output: IQ policy outputs a multicategorical policy (so $n \cdot d$ values, where d is the dimensionality of the action space and n is the number of discrete action on each dimension), while SAC policy outputs 2 d-dimensional vectors (mean and diagonal covariance matrix of a Gaussian). We use $n = 11$ in our experiments. IQ introduces two hyperparameters, α and τ. We tested several values of τ between 10^{-4} and 1, and selected a value per task suite: we use $\tau = 0.01$ on Mujoco tasks and $\tau = 0.001$ on Adroit. We tested values of α in $\{0., 0.1, 0.5, 0.9, 0.99\}$. To make the distinction between the cases when $\alpha = 0$ and $\alpha > 0$, we denote IQ($\alpha > 0$) as M-IQ, for Munchausen-IQ, since it makes use of the Munchausen regularization term. For M-IQ, we found $\alpha = 0.9$ to be the best performing value, which is consistent with the findings of Vieillard et al. [2020b]. We report results for non-optimal values of τ in the ablation study (Section 6). Extended explanations are provided in Appendix B.2.

Baselines. On continuous control tasks, our main baseline is SAC, as it reaches state-of-the-art performance on Mujoco tasks. We compare to the version of SAC that uses an adaptive temperature for reference, but note that for IQ we keep a fixed temperature (τ) setting. To reach its best performance, SAC either uses a specific temperature value per task, or an adaptive scheme that controls the entropy of the policy. This method could be extended to multicategorical policies, but we leave this for future work, and for IQ we use the same value of τ for all tasks of an environment (10^{-2} on Gym, 10^{-3} on Adroit). We use SAC with the default parameters from Haarnoja et al. [2018b] on Gym, and a specificly tuned version of SAC on Adroit. Remarkably, SAC and IQ work with similar hyperparameter ranges on both benchmarks. We only found that using a learning rate of $3 \cdot 10^{-5}$ (instead of $3 \cdot 10^{-4}$) gave better performance on Adroit. We also compare IQ to Trust-PCL. It is the closest algorithm to IQ, with a similar parametrization. To be fair, we compare to our version of Trust-PCL, which is essentially a residual version of IQ, where the target value network $V_{\bar{\phi}}$ is replaced by the online one. We use Trust-PCL with a fixed temperature, and we tuned this temperature to the environment. We found that Trust-PCL reaches its best performance with significantly lower values of τ compared to IQ. In the ablation (Fig. 2) we used $\tau = 10^{-4}$ for PCL and Trust-PCL.

Comparison to baselines. We report aggregated results of IQ and M-IQ on Gym in Fig. 2 and on Adroit in Fig. 3, and corresponding standard deviations in Appx. B.3. IQ reaches competitive performance to SAC. It is less sample efficient on Gym (SAC reaches higher performance sooner), but faster on Adroit, and IQ reaches a close final performance on both environments. These results also show the impact of the α parameter. Although the impact of the Munchausen term (*i.e* KL regularization) might not seem as impressive as in discrete actions, these results show that using that term is never detrimental, and can even bring a slight improvement on Gym; while it does not add any compute complexity to the algorithm. We also report scores on each individual task in Appx. B.3, along with in-depth discussion on the performance and the impact of hyperparameters.

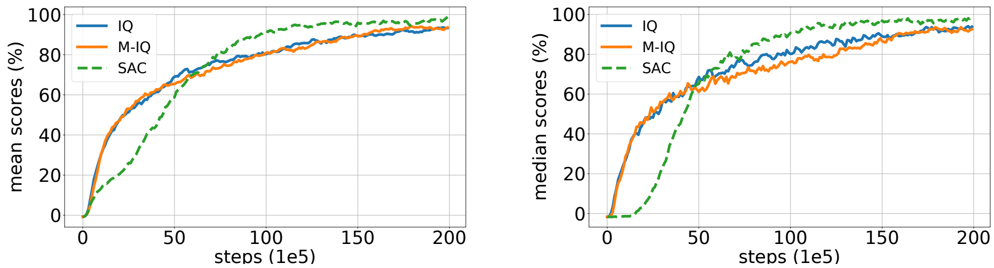

Figure 3: SAC-normalized scores on Adroit. **Left**: Mean scores. **Right:** Median scores.

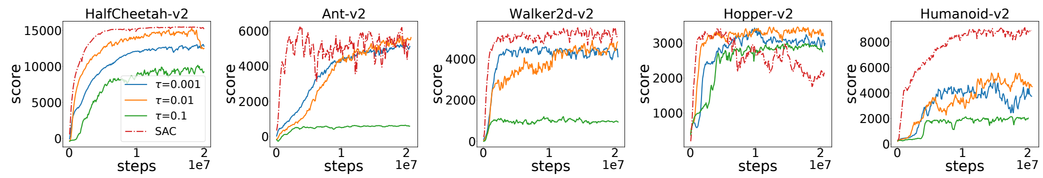

Figure 4: Influence of τ on IQ with $\alpha = 0$.

Influence of the temperature. We study the influence of the temperature on the Mujoco tasks in Fig. 4. We report the score of IQ for several values of τ (with $\alpha = 0$ here,and with $\alpha > 0$ in Appx.B.3), on all environments of Mujoco. It shows that τ needs to be selected carefully: while it helps learning, too high values of τ can be detrimental to the performance, and it highlights that its optimal value might be dependant on the task. Another observation is that τ has a much stronger influence on IQ than α. This is a key empirical difference regarding the performance of M-DQN [Vieillard et al., 2020b]. M-DQN shares similar τ and α parameters, but is specific to discrete actions. It benefits from a high value of α: M-DQN with $\alpha = 0.9$ largely outperforms M-DQN with $\alpha = 0$ on the Atari benchmark. While this term still has effect in IQ on some tasks, it is empirically less useful, even though it is never detrimental; this discrepancy is yet to be understood.

Ablation study. We perform an ablation on important components of IQ in Fig. 2. **(1)** We replace the target network by its online counterpart in Eq. (2), which gives us Trust-PCL (and PCL is obtained by setting $\alpha = 0$), a residual version of our method. IQ and M-IQ both outperform Trust-PCL and PCL on Mujoco. **(2)** We use a Gaussian parametrization of the policy instead of a multicategorical distribution. We observe on Fig. 2 that this causes the performance to drop drastically. This empirically validates the considerations about the necessary complexity of the policy from Section 4.

7 Conclusion

We introduced IQ, a parametrization of a Q-value that mechanically preserves the softmax relation between a policy and an implicit Q-function. Building on this parametrization, we derived an off-policy algorithm, that learns a policy and a value by minimizing a single loss, in a fixed-point fashion. We provided insightful analysis that justifies our parametrization and the algorithm. Specifically, IQ performs entropy and (implicit) KL regularization on the policy. While this kind of regularization had already been used and analyzed in RL, it was limited by the difficulty of estimating the softmax of Q-function in continuous action settings. IQ ends this limitation by avoiding any approximation in this softmax, effectively extending the analysis of this regularization. This parametrization comes at a cost: it shifts the representation capacity from the Q-network to the policy, which makes the use of Gaussian representation ineffective. We solved this issue by considering simple multicategorical policies, which allowed IQ to reach performance comparable to state-of-the-art methods on classic continuous control benchmarks. Yet, we envision that studying even richer policy classes may results in even better performance. In the end, this work brings together theory and practice: IQ is a theory-consistent manner of implementing an algorithm based on regularized VI in continuous actions settings.

References

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on learning Representations (ICLR)*, 2018.

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Marc G Bellemare, Georg Ostrovski, Arthur Guez, Philip S Thomas, and Rémi Munos. Increasing the action gap: New operators for reinforcement learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2016.

Dimitri P Bertsekas and John N Tsitsiklis. *Neuro dynamic programming*. Athena Scientific Belmont, MA, 1996.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Jonas Degrave, Abbas Abdolmaleki, Jost Tobias Springenberg, Nicolas Heess, and Martin Riedmiller. Quinoa: a q-function you infer normalized over actions. *Deep RL Workshop at NeurIPS*, 2018.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, pages 1587–1596. PMLR, 2018.

Matthieu Geist, Bilal Piot, and Olivier Pietquin. Is the bellman residual a bad proxy? *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A Theory of Regularized Markov Decision Processes. In *International Conference on Machine Learning (ICML)*, 2019.

Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning (ICML)*, 2016.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Fundamentals of convex analysis*. Springer Science & Business Media, 2004.

Matt Hoffman, Bobak Shahriari, John Aslanides, Gabriel Barth-Maron, Feryal Behbahani, Tamara Norman, Abbas Abdolmaleki, Albin Cassirer, Fan Yang, Kate Baumli, Sarah Henderson, Alex Novikov, Sergio Gómez Colmenarejo, Serkan Cabi, Caglar Gulcehre, Tom Le Paine, Andrew Cowie, Ziyu Wang, Bilal Piot, and Nando de Freitas. Acme: A research framework for distributed reinforcement learning. *arXiv preprint arXiv:2006.00979*, 2020. URL `https://arxiv.org/abs/2006.00979`.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *nternational Conference for Learning Representations (ICLR)*, 2015.

Luke Metz, Julian Ibarz, Navdeep Jaitly, and James Davidson. Discrete sequential prediction of continuous actions for deep rl. *arXiv preprint arXiv:1705.05035*, 2017.

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, 2015.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Trust-pcl: An off-policy trust region method for continuous control. *International Conference on Learning Representations (ICLR)*, 2018.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning (ICML)*. PMLR, 2015.

Bruno Scherrer, Mohammad Ghavamzadeh, Victor Gabillon, Boris Lesner, and Matthieu Geist. Approximate modified policy iteration and its application to the game of Tetris. *Journal of Machine Learning Research*, 16:1629–1676, 2015.

Yunhao Tang and Shipra Agrawal. Discretizing continuous action space for on-policy optimization. In *AAAI Conference on Artificial Intelligence 5AAAI)*, volume 34, pages 5981–5988, 2020.

Nino Vieillard, Tadashi Kozuno, Bruno Scherrer, Olivier Pietquin, Rémi Munos, and Matthieu Geist. Leverage the average: an analysis of kl regularization in rl. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020a.

Nino Vieillard, Olivier Pietquin, and Matthieu Geist. Munchausen reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020b.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning (ICML)*, pages 1995–2003. PMLR, 2016.

Patrick Nadeem Ward, Ariella Smofsky, and Avishek Joey Bose. Improving exploration in soft-actor-critic with normalizing flows policies. *arXiv preprint arXiv:1906.02771*, 2019.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Brian D Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, University of Washington, 2010.