Extreme Q-Learning: MaxEnt RL without Entropy

Anonymous Author(s)
Affiliation
Address
email

Abstract

Modern Deep Reinforcement Learning (RL) algorithms require estimates of the maximal Q-value, which are difficult to compute in continuous domains with an infinite number of possible actions. In this work, we introduce a new update rule for online and offline RL which directly models the maximal value using Extreme Value Theory (EVT) inspired by Economics. By doing so, we avoid computing Q-values using out-of-distribution actions which is often a substantial source of error. Our key insight is to introduce an objective that directly estimates the optimal soft-value functions (LogSumExp) in the maximum entropy (MaxEnt) RL setting without needing to sample from a policy. Using EVT, we derive our *Extreme Q-Learning* framework and consequently online and, for the first time, offline MaxEnt Q-learning algorithms, that do not explicitly require access to a policy or its entropy. Finally, our method obtains state-of-the-art results in the Offline D4RL benchmark strongly outperforming prior works by *10-20 points* while obtaining moderate improvements over SAC and TD3 on online DM Control tasks.

1 Introduction

Modern Deep Reinforcement Learning (RL) algorithms have shown broad success in challenging control [14, 33] and game-playing domains [25]. While tabular Q-iteration or value-iteration methods are well understood, state of the art RL algorithms often make theoretical compromises in order to deal with deep networks, high dimensional state spaces, and continuous action spaces. In particular, standard Q-learning algorithms require computing the max or soft-max over the Q-function in order to fit the Bellman equations. Yet, almost all current off-policy RL algorithms for continuous control only indirectly estimate the Q-value of the next state with separate policy networks. Consequently, these methods only estimate the Q-function of the current policy, instead of the optimal Q^*, and rely on policy improvement via an actor. Moreover, actor-critic approaches on their own have shown to be catastrophic in the offline settings where actions sampled from a policy are consistently out-of-distribution [19, 10]. As such, computing $\max Q$ for Bellman targets remains a core issue in deep RL.

One popular approach is to train Maximum Entropy (MaxEnt) policies, in hopes that they are more robust to modeling and estimation errors [42]. However, the Bellman backup $\mathcal{B}^*$ used in MaxEnt RL algorithms still requires computing the log-partition function over Q-values, which is usually intractable in high-dimensional action spaces. Instead, current methods like SAC [14] rely on auxiliary policy networks, and as a result do not estimate $\mathcal{B}^*$, the optimal Bellman backup. Our key insight is to apply extreme value analysis used in branches of Finance and Economics to Reinforcement Learning. Ultimately, this will allow us to directly model the LogSumExp over Q-functions in the MaxEnt Framework.

Intuitively, reward or utility-seeking agents will consider the maximum of the set of possible future returns. The Extreme Value Theorem (EVT) tells us that maximal values drawn from any exponential tailed distribution follows the Generalized Extreme Value (GEV) Type-1 distribution, also referred to as the Gumbel Distribution $\mathcal{G}$. The Gumbel distribution is thus a prime candidate for modeling errors

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

in Q-functions. In fact, McFadden's 2000 nobel-prize winning work in Economics on discrete choice models [24] showed that soft-optimal utility functions with logit (or softmax) choice probabilities naturally arise when utilities are assumed to have Gumbel-distributed errors. This was subsequently generalized to stochastic MDPs by Rust [32] in 1984. Nevertheless, these intriguing results have remained largely unknown in the RL community. By introducing a novel loss optimization framework, we bring them into the world of modern deep RL.

Empirically, we find that even modern deep RL approaches, for which errors are typically assumed to be Gaussian, exhibit errors that better approximate the Gumbel Distribution, see Figure 1. By assuming errors to be Gumbel distributed, we obtain Gumbel Regression, a consistent estimator over log-partition functions even in continuous spaces. Furthermore, making this assumption about Q-values lets us derive a new Bellman loss objective that directly solves for the optimal MaxEnt Bellman operator $\mathcal{B}^*$, instead of the operator under the current policy $\mathcal{B}^\pi$. As soft optimality gracefully emerges from our framework, we can run MaxEnt RL independently of the policy. In the online setting, we avoid using a policy network to explicitly compute entropies. In the offline setting, we completely avoid sampling from learned policy networks, minimizing the aforementioned extrapolation error. Our resulting algorithms surpass state-of-the-art (SOTA) while being practically simpler.

In this paper we outline the theoretical motivation for using Gumbel distributions in reinforcement learning, and show how it can be used to derive practical online and offline MaxEnt RL algorithms. Concretely, our contributions are as follows:

- We motivate Gumbel Regression and show it allows calculation of the log-partition function (LogSumExp) in continuous spaces. We apply it to MDPs to present a novel loss objective for RL using maximum-likelihood estimation.
- Our formulation extends soft-Q learning to offline RL as well as continuous action spaces without the need of policy entropies. It presents a way to directly calculate the optimal soft-values V^* and soft-Bellman updates $\mathcal{B}^*$ using SGD, which are usually intractable in continuous settings.
- We provide the missing theoretical link between soft and conservative Q-learning, showing how these formulations can be made equivalent. We also show how Max-Ent RL emerges naturally from vanilla RL as a conservatism in our framework.
- Finally, we empirically demonstrate state-of-art results in Offline RL, improving over prior methods by a large margin on the D4RL benchmark, and performing moderately better than SAC and TD3 in Online RL, while theoretically avoiding actor-critic formulations.

2 Preliminaries

In this section we introduce Maximium Entropy (MaxEnt) RL and Extreme Value Theory (EVT), which we use to motivate our framework to estimate extremal values in RL.

We consider an infinite-horizon Markov decision process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where $\mathcal{S}, \mathcal{A}$ represent state and action spaces, $\mathcal{P}(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ represents the environment dynamics, $r(\mathbf{s}, \mathbf{a})$ represents the reward function, and $\gamma \in (0, 1)$ represents the discount factor. In the offline RL setting, we are given a dataset $\mathcal{D} = (\mathbf{s}, \mathbf{a}, r, \mathbf{s}')$ of tuples sampled from trajectories under a behavior policy $\pi_\mathcal{D}$ without any additional environment interactions. We use $\rho_\pi(\mathbf{s})$ to denote the distribution of states that a policy $\pi(\mathbf{a}|\mathbf{s})$ generates. In the MaxEnt framework, an MDP with entropy-regularization is referred to as a soft-MDP [4] and we often use this notation.

2.1 Maximum Entropy RL

Standard RL seeks to learn a policy that maximizes the expected sum of (discounted) rewards $\mathbb{E}_\pi\left[\sum_{t=0}^\infty \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)\right]$, for $(\mathbf{s}_t, \mathbf{a}_t)$ drawn at timestep t from the trajectory distribution that π generates. We consider a generalized version of Maximum Entropy RL that augments the standard reward objective with the KL-divergence between the policy and a reference distribution μ: $\mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t (r(\mathbf{s}_t, \mathbf{a}_t) - \beta \log \frac{\pi(\mathbf{a}_t|\mathbf{s}_t)}{\mu(\mathbf{a}_t|\mathbf{s}_t)})]$, where β is the regularization strength. When μ is uniform $\mathcal{U}$, this becomes the standard MaxEnt objective used in online RL up to a constant. In the offline RL setting, we choose μ to be the behavior policy $\pi_\mathcal{D}$ that generated the fixed dataset $\mathcal{D}$. Consequently, this objective enforces a conservative KL-constraint on the learned policy, keeping it close to the behavior policy [28, 14].

In MaxEnt RL, the soft-Bellman operator $\mathcal{B}^* : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ is defined as $(\mathcal{B}^*Q)(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \gamma\mathbb{E}_{\mathbf{s}'\sim\mathcal{P}(\cdot|\mathbf{s},\mathbf{a})}V^*(\mathbf{s}')$ where Q is the soft-Q function and V^* is the optimal soft-value satisfying:

$$V^*(\mathbf{s}) = \beta \log \sum_{\mathbf{a}} \mu(\mathbf{a}|\mathbf{s}) \exp\left(Q(\mathbf{s},\mathbf{a})/\beta\right) := \mathbb{L}^{\beta}_{a\sim\mu(\cdot|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right], \tag{1}$$

where we denote the log-sum-exp (LSE) using an operator $\mathbb{L}^{\beta}$ for succinctness[1]. The soft-Bellman operator has a unique contraction Q^* [14] given by the soft-bellman equation: $Q^* = \mathcal{B}^*Q^*$ and the optimal policy satisfies [13]:

$$\pi^*(\mathbf{a}|\mathbf{s}) = \mu(\mathbf{a}|\mathbf{s}) \exp\left((Q^*(\mathbf{s},\mathbf{a}) - V^*(\mathbf{s}))/\beta\right). \tag{2}$$

Instead of estimating soft-values for a policy $V^{\pi}(\mathbf{s}) = \mathbb{E}_{\mathbf{a}\sim\pi(\cdot|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a}) - \beta\log\frac{\pi(\mathbf{a}|\mathbf{s})}{\mu(\mathbf{a}|\mathbf{s})}\right]$, our approach will seek to directly fit the optimal soft-values V^*, i.e. the log-sum-exp (LSE) of Q values.

2.2 Extreme Value Theorem

The Fisher-Tippett or extreme value theorem tells us that the maximum of i.i.d. samples from exponentially tailed distributions will asymptotically converge to the Gumbel distribution $\mathcal{G}(\mu,\beta)$, which has pdf $p(x) = \exp(-(z + e^{-z}))$ where $z = (x-\mu)/\beta$ with location parameter μ and scale parameter β.

Theorem 1 (Extreme Value Theorem (EVT) [26, 8])**.** *For i.i.d. random variables $X_1, ..., X_n \sim f_X$, with exponential tails, $\lim_{n\to\infty}\max_i(X_i)$ follows the Gumbel (GEV-1) distribution. Furthermore, $\mathcal{G}$ is max-stable, i.e. if $X_i \sim \mathcal{G}$, then $\max_i(X_i) \sim \mathcal{G}$ holds.*

This result is similar to the Central Limit Theorem (CLT), which states that means of i.i.d. errors approach the normal distribution. Thus, under a chain of max operations, any i.i.d. exponential tailed errors[2] will tend to become Gumbel distributed and stay as such. EVT will ultimately suggest us to characterize nested errors in Q-learning as following a Gumbel distribution. In particular, the Gumbel distribution $\mathcal{G}$ exhibits unique properties we will exploit.

One intriguing consequence of the Gumbel's max-stability is its ability to convert the maximum over a discrete set into a softmax. This is known as the *Gumbel-Max Trick* [29, 15]. Concretely for i.i.d. $\epsilon_i \sim \mathcal{G}(0,\beta)$ added to a set $\{x_1, ..., x_n\} \in \mathbb{R}$, $\max_i(x_i + \epsilon_i) \sim \mathcal{G}(\beta\log\sum_i \exp(x_i/\beta), \beta)$, and $\text{argmax}(x_i + \epsilon_i) \sim \text{softmax}(x_i/\beta)$. Furthermore, the Max-trick is unique to the Gumbel [23].

These properties lead into the McFadden-Rust model [24, 32] of MDPs as we state below.

McFadden-Rust model: An MDP following the standard Bellman equations with stochasticity in the rewards due to unobserved state variables will satisfy the soft-Bellman equations over the observed state with actual rewards $\bar{r}(\mathbf{s},\mathbf{a})$, given two conditions:

1. Additive separability (AS): observed rewards have additive i.i.d. Gumbel noise, i.e. $r(\mathbf{s},\mathbf{a}) = \bar{r}(\mathbf{s},\mathbf{a}) + \epsilon(\mathbf{s},\mathbf{a})$, with actual rewards $\bar{r}(\mathbf{s},\mathbf{a})$ and i.i.d. noise $\epsilon(\mathbf{s},\mathbf{a}) \sim \mathcal{G}(0,\beta)$.
2. Conditional Independence (CI): the noise $\epsilon(\mathbf{s},\mathbf{a})$ in a given state-action pair is conditionally independent of that in any other state-action pair.

Moreover, the converse also holds: Any MDP satisfying the Bellman equations and following a softmax policy, necessarily has any i.i.d. noise in the rewards with $AS + CI$ conditions be Gumbel distributed.

These results were first shown to hold in discrete choice theory by McFadden [24], with the $AS + CI$ conditions derived by Rust [32] for discrete MDPs. We formalize these results in Appendix A and give succinct proofs using the developed properties of the Gumbel distribution. These results enable the view of a soft-MDP as an MDP with hidden i.i.d. Gumbel noise in the rewards.

Notably, this result gives a different interpretation of a soft-MDP than entropy regularization to allow us to recover the soft-Bellman equations.

[1] In continuous action spaces, the sum over actions is replaced with an integral over the distribution μ.

[2] Bounded random variables are sub-Gaussian [41] which have exponential tails.

3 Extreme Q-Learning

In this section, we motivate our Extreme Q-learning framework, which directly models the soft-optimal values V^*, and show it naturally extends soft-Q learning. Notably, we use the Gumbel distribution to derive a new optimization framework for RL via maximum-likelihood estimation and apply it to both online and offline settings.

3.1 Gumbel Error Model

Although assuming Gumbel errors in MDPs leads to intriguing properties, it is not obvious why the errors might be distributed as such. First, we empirically investigate the distribution of Bellman errors by computing them over the course of training. Specifically, we compute $r(\mathbf{s}, \mathbf{a}) - \gamma Q(\mathbf{s}', \pi(\mathbf{s}')) - Q(\mathbf{s}, \mathbf{a})$ for samples $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ from the replay-buffer using a single Q-function from SAC [14] (See Appendix E for more details). In Figure 1, we find the errors to be skewed and better fit by a Gumbel distribution. We explain this using EVT.

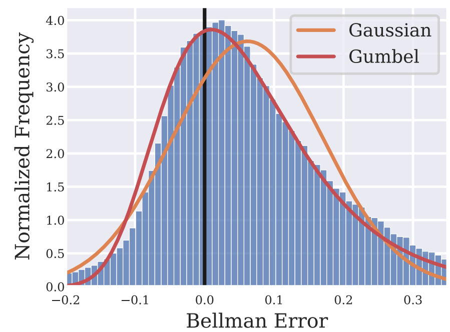

Figure 1: Bellman errors from SAC on Cheetah-Run [35]. The Gumbel distribution better captures the skew versus the Gaussian.

Consider fitting Q-functions by learning an unbiased function approximator $\hat{Q}$ to solve the Bellman equation. We will assume we have access to M such function approximators, each of which are assumed to be independent e.g. parallel runs of a model over an experiment. We can see approximate Q-iteration as performing:

$$\hat{Q}_t(\mathbf{s}, \mathbf{a}) = \bar{Q}_t(\mathbf{s}, \mathbf{a}) + \epsilon_t(\mathbf{s}, \mathbf{a}), \tag{3}$$

where $\mathbb{E}[\hat{Q}] = \bar{Q}_t$ is the expected value of our prediction $\hat{Q}_t$ for an intented target $\bar{Q}_t$ over our estimators, and ϵ_t is the (zero-centered) error in our estimate. Here, we assume the error ϵ_t comes from the same underlying distribution for each of our estimators, and thus are i.i.d. random variables with a zero-mean. Now, consider the bootstrapped estimate using one of our M estimators chosen randomly:

$$\hat{\mathcal{B}}^* \hat{Q}_t(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \max_{\mathbf{a}'} \hat{Q}_t(\mathbf{s}', \mathbf{a}') = r(\mathbf{s}, \mathbf{a}) + \gamma \max_{\mathbf{a}'} (\bar{Q}_t(\mathbf{s}', \mathbf{a}') + \epsilon_t(\mathbf{s}', \mathbf{a}')). \tag{4}$$

We now examine what happens after a subsequent update. At time $t+1$, suppose that we fit a fresh set of M independent functional approximators $\hat{Q}_{t+1}$ with the target $\hat{\mathcal{B}}^* \hat{Q}_t$, introducing a new unbiased error ϵ_{t+1}. Then, for $\bar{Q}_{t+1} = \mathbb{E}[\hat{Q}_{t+1}]$ it holds that

$$\bar{Q}_{t+1}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}'|\mathbf{s},\mathbf{a}}[\mathbb{E}_{\epsilon_t}[\max_{\mathbf{a}'} (\bar{Q}_t(\mathbf{s}', \mathbf{a}') + \epsilon_t(\mathbf{s}', \mathbf{a}'))]]. \tag{5}$$

As $\bar{Q}_{t+1}$ is an expectation over both the dynamics and the functional errors, it accounts forall uncertainty (here $\mathbb{E}[\epsilon_{t+1}] = 0$). But, the i.i.d. error ϵ_t remains and will be propagated through the Bellman equations and its chain of $\max$ operations. Due to Lemma 1, ϵ_t will become Gumbel distributed in the limit of t, and remain so due to the Gumbel distribution's max-stability.[3]

This highlights a fundamental issue with approximation-based RL algorithms that minimize the Mean-Squared Error (MSE) in the Bellman Equation: they implicitly assume, via maximum likelihood estimation, that errors are Gaussian. In Appendix A, we further study the propagation of errors using the McFadden-Rust MDP model, and use it to develop a simplified Gumbel Error Model (GEM) for errors under functional approximation.

In practice, the Gumbel nature of the errors may be weakened as, unlike in our model, estimators between timesteps share parameters and errors will be correlated across states and actions.

3.2 Gumbel Regression

The goal of our work is to directly model the log-partition function (LogSumExp) over $Q(s, a)$ to avoid all of the aforementioned issues with taking a $\max$ in the function approximation domain.

[3] Same holds for soft-MDPs as log-sum-exp can be expanded as a max over i.i.d. Gumbel random vars.

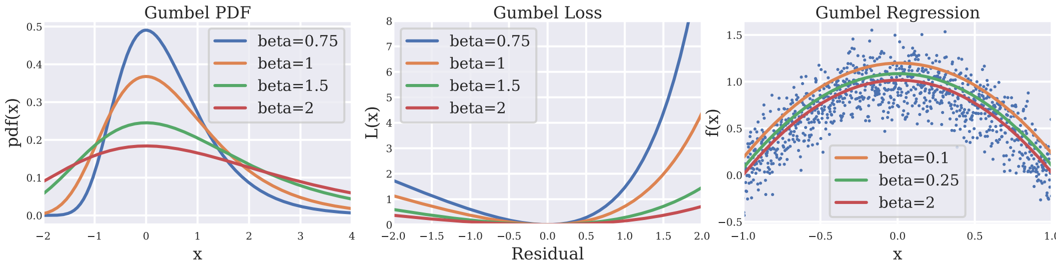

Figure 2: **Left**: The pdf of the Gumbel distribution with $\mu = 0$ and different values of β. **Center**: Our Gumbel loss for different values of β. **Right**: Gumbel regression applied to a two-dimensional random variable for different values of β. The smaller the value of β, the more the regression fits the extrema.

In this section we derive an objective function that models the LogSumExp by simply assuming errors to follow a gumbel distribution.

Consider estimating a parameter h for a random variable X using samples x_i from a dataset $\mathcal{D}$, which have Gumbel distributed noise, i.e. $x_i = h + \epsilon_i$ where $\epsilon_i \sim -\mathcal{G}(0, \beta)$. Then, the average log-likelihood of the dataset $\mathcal{D}$ as a function of h is given as:

$$\mathbb{E}_{x_i \sim \mathcal{D}} [\log p(x_i)] = \mathbb{E}_{x_i \sim \mathcal{D}} \left[-e^{((x_i - h)/\beta)} + (x_i - h)/\beta \right] \tag{6}$$

Maximizing the log-likelihood yields the following convex minimization objective in h,

$$\mathcal{L}(h) = \mathbb{E}_{x_i \sim \mathcal{D}} \left[e^{(x_i - h)/\beta} - (x_i - h)/\beta - 1 \right] \tag{7}$$

which forms our objective function $\mathcal{L}(\cdot)$[4] for Gumbel Regression. β is fixed as a hyper-parameter, and we show its affect on the loss in Figure 2. Critically, the minima of this objective under a fixed β is given by $h = \beta \log \mathbb{E}_{x_i \sim \mathcal{D}}[e^{x_i/\beta}]$, which resembles the LogSumExp with the summation replaced with an (empirical) expectation.

In fact, this solution is the the same as the operator $\mathbb{L}^{\beta}_{\mu}(X)$ defined for MaxEnt in Section 2.1 with x_i sampled from μ. In Figure 2, we show plots of Gumbel Regression on a simple dataset with different values of β. As this objective recovers $\mathbb{L}^{\beta}(X)$, we next use it to model soft-values in Max-Ent RL.

3.2.1 Theory

Here we show that our objective is well behaved, considering the previously defined operator $\mathbb{L}^{\beta}$ for random variables $\mathbb{L}^{\beta}(X) := \beta \log \mathbb{E}\left[e^{X/\beta}\right]$. First, we show it models the extremum.

Lemma 3.1. *For any $\beta_1 > \beta_2$, we have $\mathbb{L}^{\beta_1}(X) < \mathbb{L}^{\beta_2}(X)$. And $\mathbb{L}^{\infty}(X) = \mathbb{E}[X]$, $\mathbb{L}^{0}(X) = sup(X)$. Thus, for any $\beta \in (0, \infty)$, the operator $\mathbb{L}^{\beta}(X)$ is a measure that interpolates between the expectation and the max of X.*

The operator $\mathbb{L}^{\beta}(X)$ is known as the cumulant-generating function or the log-Laplace transform, and is a measure of the tail-risk closely linked to the entropic value at risk (EVaR) [1] .

Lemma 3.2. *The risk measure $\mathcal{L}$ has a unique minima at $\beta \log \mathbb{E}\left[e^{X/\beta}\right]$. And an empirical risk $\hat{\mathcal{L}}$ is an unbiased estimate of the true risk. Furthermore, for $\beta \gg 1$, $\mathcal{L}(\theta) \approx \frac{1}{2\beta^2} \mathbb{E}_{x_i \sim \mathcal{D}}[(x_i - \theta)^2]$, thus behaving as the MSE loss with errors $\sim \mathcal{N}(0, \beta)$.*

In particular, the empirical loss $\hat{\mathcal{L}}$ over a dataset of N samples can be minimized using stochastic gradient-descent (SGD) methods to give an unbiased estimate of the LogSumExp over the N samples.

Lemma 3.3. *$\hat{\mathbb{L}}^{\beta}(X)$ over a finite N samples is a consistent estimator of the log-partition function $\mathbb{L}^{\beta}(X)$. Similarly, $\exp(\hat{\mathbb{L}}^{\beta}(X)/\beta)$ is an unbiased estimator for the partition function $Z = \mathbb{E}\left[e^{X/\beta}\right]$*

We provide concentration bounds for Lemma 3.3, and further theoretical discussion on Gumbel Regression in Appendix A.

[4]We add -1 to make the loss 0 for a perfect fit, as $e^x - x - 1 \geq 0$ with equality at $x = 0$.

3.3 MaxEnt RL without Entropy

Given Gumbel Regression can be used to directly model the LogSumExp , we apply it to Q-learning. First, we establish the connection of our framework with conservative Q-learning [19].

Lemma 3.4. *Consider the loss objective over Q-functions:*

$$\mathcal{L}(Q) = \mathbb{E}_{\mathbf{s}\sim\rho_\mu,\mathbf{a}\sim\mu(\cdot|\mathbf{s})}\left[e^{(\mathcal{T}^\pi\hat{Q}^k(\mathbf{s},\mathbf{a})-Q(\mathbf{s},\mathbf{a}))/\beta}\right] - \mathbb{E}_{\mathbf{s}\sim\rho_\mu,\mathbf{a}\sim\pi(\cdot|\mathbf{s})}[(\mathcal{T}^\pi\hat{Q}^k(\mathbf{s},\mathbf{a}) - Q(\mathbf{s},\mathbf{a}))/\beta] - 1 \tag{8}$$

where $\mathcal{T}^\pi := r(\mathbf{s},\mathbf{a}) + \gamma\mathbb{E}_{\mathbf{s}'|\mathbf{s},\mathbf{a}}\mathbb{E}_{\mathbf{a}'\sim\pi}[Q(\mathbf{s}',\mathbf{a}')]$ *is the vanilla Bellman operator under the policy* $\pi(\mathbf{a}|\mathbf{s})$. *Then minimizing* $\mathcal{L}$ *gives the update rule:*

$$\forall \mathbf{s},\mathbf{a},k \;\; \hat{Q}^{k+1}(\mathbf{s},\mathbf{a}) = \mathcal{T}^\pi\hat{Q}^k(\mathbf{s},\mathbf{a}) - \beta\log\frac{\pi(\mathbf{a}\mid\mathbf{s})}{\mu(\mathbf{a}\mid\mathbf{s})} = \mathcal{B}^\pi\hat{Q}^k(\mathbf{s},\mathbf{a}).$$

Thus this transforms a regular Bellman backup into the soft-Bellman backup without the need of entropies.

Using the above lemma we can convert standard RL into Max-Ent RL. Here, $\mathcal{L}(\cdot)$ does a conservative Q-update similar to CQL [19] with the nice property that the implied conservative term is just the KL-constraint between π and μ.[5] This enforces a entropy-regularization on our policy with respect to the behavior policy without the need of entropy. Thus, soft-Q learning naturally emerges as a conservative update on regular Q-learning under our objective. Here, Equation 8 is the dual of the KL-divergence between μ and π [12], and we motivate this objective for RL and establish formal equivalence with conservative Q-learning in Appendix C.

In our framework, we use the MaxEnt Bellman operator $\mathcal{B}^*$ which gives our *ExtremeQ* loss:

$$\mathcal{L}(Q) = \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mu}\left[e^{(\hat{\mathcal{B}}^*\hat{Q}^k(\mathbf{s},\mathbf{a})-Q(\mathbf{s},\mathbf{a}))/\beta}\right] - \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mu}[(\hat{\mathcal{B}}^*\hat{Q}^k(\mathbf{s},\mathbf{a}) - Q(\mathbf{s},\mathbf{a}))/\beta] - 1 \tag{9}$$

and is the same as our Gumbel loss from the previous section.

This gives an update rule: $\hat{Q}^{k+1}(\mathbf{s},\mathbf{a}) = \mathcal{B}^*\hat{Q}^k(\mathbf{s},\mathbf{a})$. $\mathcal{L}(\cdot)$ here requires estimation of $\mathcal{B}^*$ which is very hard in continuous action spaces. Under deterministic dynamics, $\mathcal{L}$ can be obtained without $\mathcal{B}^*$ as shown in Appendix C. However, in general we still need to estimate $\mathcal{B}^*$. Next, we motivate how we can solve this issue.

Consider the soft-Bellman equation from Section 2.1 (Equation 1),

$$\mathcal{B}^*Q = r(\mathbf{s},\mathbf{a}) + \gamma\mathbb{E}_{\mathbf{s}'\sim P(\cdot|\mathbf{s},\mathbf{a})}[V^*(\mathbf{s}')], \tag{10}$$

where $V^*(\mathbf{s}) = \mathbb{L}^\beta_{\mathbf{a}\sim\mu(\cdot|\mathbf{s}')}[Q(\mathbf{s},\mathbf{a})]$. Then V^* can be directly estimated using Gumbel regression by setting the temperature β to the regularization strength in the MaxEnt framework.

This gives us the following *ExtremeV* loss objective:

$$\mathcal{J}(V) = \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mu}\left[e^{(\hat{Q}^k(\mathbf{s},\mathbf{a})-V(\mathbf{s}))/\beta}\right] - \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mu}[(\hat{Q}^k(\mathbf{s},\mathbf{a}) - V(\mathbf{s}))/\beta] - 1. \tag{11}$$

Lemma 3.5. *Minimizing* $\mathcal{J}$ *over values gives the update rule:* $\hat{V}^k(\mathbf{s}) = \mathbb{L}^\beta_{\mathbf{a}\sim\mu(\cdot|\mathbf{s})}[\hat{Q}^k(\mathbf{s},\mathbf{a})]$.

Then we can obtain V^* from $Q(s,a)$ using Gumbel regression and substitute in Equation 10 to estimate the optimal bellman backup $\mathcal{B}^*Q$. Thus, Lemma 3.4 and 3.5 give us a scheme to solve the Max-Ent RL problem without the need of entropy.

3.4 Learning Policies

In the above section we derived a Q-learning strategy that does not require explicit use of a policy π. However, in continuous settings we still often want to recover a policy that can be run in the environment. Per Eq. 2 (Section 2.2), the optimal MaxEnt policy $\pi^*(\mathbf{a}|\mathbf{s}) = \mu(\mathbf{a}|\mathbf{s})e^{(Q(\mathbf{s},\mathbf{a})-V(\mathbf{s}))/\beta}$. By minimizing the KL-divergence between π and the optimal π^* induced by Q and V we obtain the objective:

$$\pi^* = \operatorname*{argmin}_\pi \mathbb{E}_{\rho_\mu(\mathbf{s},\mathbf{a})}[e^{(Q(\mathbf{s},\mathbf{a})-V(\mathbf{s}))/\beta}\log\pi]. \tag{12}$$

[5]In fact, all theorems of CQL [19] hold for our objective by simple replacement of D_{CQL} with D_{KL}.

If we take ρ_μ to be a dataset $\mathcal{D}$ generated from a behavior policy $\pi_{\mathcal{D}}$, we exactly recover the AWAC objective used by prior works in Offline RL [27]. This objective does not require sampling actions, which may potentially take $Q(s, a)$ out of distribution. Alternatively, if we want to sample from the policy we can minimize the Reverse-KL divergence which gives us the SAC-like actor update:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_{\rho_\pi(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s}, \mathbf{a}) - \beta \log(\pi(\mathbf{a}|\mathbf{s})/\mu(\mathbf{a}|\mathbf{s}))]. \tag{13}$$

Interestingly, we note this doesn't depend on $V(s)$. If μ is chosen to be the last policy π_k, the second term becomes the KL-divergence between the current policy and π_k, performing a trust region update on π [33, 37].[6] While estimating the log ratio $\log(\pi(\mathbf{a}|\mathbf{s})/\mu(\mathbf{a}|\mathbf{s}))$ can be difficult depending on choice of μ, we can remove the need for an analytic form of μ using our Gumbel loss objective. As applying our Gumbel Loss $\mathcal{J}$ on samples from μ leads to soft-Q values of the form $Q(\mathbf{s}, \mathbf{a}) - \beta \log(\pi(\mathbf{a}|\mathbf{s})/\mu(\mathbf{a}|\mathbf{s}))$, our policy can be updated by simply maximizing the soft-Q values. Updating $\mathcal{J}$ with samples from π_k thus naturally gives rise to trust-region style policy updates.

3.5 Practical Algorithms

In this section we develop a practical approach to Extreme Q-learning ($\mathcal{X}$-QL) for both online and offline RL. We consider parameterized functions $V_\theta(\mathbf{s})$, $Q_\phi(\mathbf{s}, \mathbf{a})$, and $\pi_\psi(\mathbf{a}|\mathbf{s})$ and let $\mathcal{D}$ be the training data distribution. A core issue with directly optimizing Eq. 10 is over-optimism about dynamics [20] when using simple-sample estimates for the Bellman backup. To overcome this issue in stochastic settings, we separate out the optimization of V_θ from that of Q_ϕ following Section 3.3. We learn V_θ using Eq. 11 to directly fit the optimal soft-values $V^*(\mathbf{s})$ based on Gumbel regression. Using $V_\theta(\mathbf{s}')$ we can get single-sample estimates of $\mathcal{B}^*$ as $r(\mathbf{s}, \mathbf{a}) + \gamma V_\theta(\mathbf{s}')$. Now we can learn an unbiased expectation over the dynamics, $Q_\phi \approx \mathbb{E}_{\mathbf{s}'|\mathbf{s},\mathbf{a}}[r(\mathbf{s}, \mathbf{a}) + \gamma V_\theta(\mathbf{s}')]$ by minimizing the Mean-squared-error (MSE) loss between the single-sample targets and Q_ϕ:

Algorithm 1 Extreme Q-learning ($\mathcal{X}$-QL) (Under Stochastic Dynamics)

1: Init Q_ϕ, V_θ, and π_ψ
2: Let $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, r, \mathbf{s}')\}$ be data from $\pi_{\mathcal{D}}$ (offline) or replay buffer (online)
3: **for** step t in {1...N} **do**
4: Train Q_ϕ using $\mathcal{L}(\phi)$ from Eq. 14
5: Train V_θ using $\mathcal{J}(\theta)$ from Eq. 11 (with $\mathbf{a} \sim \mathcal{D}$ (offline) or $\mathbf{a} \sim \pi_\psi$ (online))
6: Update π_ψ via Eq. 12 (offline) or Eq. 13 (online)
7: **end for**

$$\mathcal{L}(\phi) = \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}}\left[(Q_\phi(\mathbf{s}, \mathbf{a}) - r(\mathbf{s}, \mathbf{a}) - \gamma V_\theta(\mathbf{s}'))^2\right]. \tag{14}$$

In deterministic dynamics, our approach is largely simplified and we directly learn a single Q_ϕ using Eq. 9 without needing to learn $\mathcal{B}^*$ or V^*. Similarly, we learn soft-optimal policies using Eq. 12 (offline) or Eq. 13 (online) settings.

Offline RL. In the offline setting, $\mathcal{D}$ is specified as an offline dataset assumed to be collected with the behavior policy $\pi_{\mathcal{D}}$. Here, learning values with Eq. 11 has a number of practical benefits. First, we are able to fit the optimal soft-values V^* *without sampling from a policy network*, which has been shown to cause large out-of-distribution errors in the offline setting where mistakes cannot be corrected by collecting additional data. Second, we inherently enforce a KL-constraint on the optimal policy π^* and the behavior policy $\pi_{\mathcal{D}}$. This provides tunable conservatism via the temperature β. After offline training of Q_ϕ and V_θ, we can recover the policy post-training using the AWAC objective (Eq. 12). Our practical implementation follows the training style of Kostrikov et al. [17], but we train value network using using our ExtremeQ loss.

Online RL. In the online setting, $\mathcal{D}$ is usually given as a replay buffer of previously sampled states and actions. In practice, however, obtaining a good estimate of $V^*(\mathbf{s}')$ requires that we sample actions with high Q-values instead of uniform sampling from $\mathcal{U}$. As online learning allows agents to correct over-optimism by collecting additional data, we use a previous version of the policy network π_ψ to sample actions for the Bellman backup, amounting to the trust-region policy update at the end of Section 3.4. In practice, we modify SAC and TD3 with our formulation. To embue SAC [14] with the benefits of Extreme Q-learning, we simply train V_θ using Eq. 11 with $\mathbf{s} \sim \mathcal{D}, \mathbf{a} \sim \pi_{\psi_k}(\mathbf{a}|\mathbf{s})$. TD3 by default doesn't use a value network, and thus we use our algorithm for deterministic dynamics by changing the loss to train Q in TD3 to directly follow Eq. 9.

[6] Choosing μ to be uniform $\mathcal{U}$ gives the regular SAC update.

4 Experiments

We compare our Extreme Q-Learning ($\mathcal{X}$-QL) approach to existing works obtaining state-of-the-art results across a wide set of continuous control tasks in both online and offline settings. More details, offline results on Adroit, and hyper-parameter settings can be found in Appendix E.

4.1 Offline RL

Table 1: Averaged normalized scores on MuJoCo locomotion and Ant Maze tasks.

	Dataset	BC	10%BC	DT	AWAC	Onestep RL	TD3+BC	CQL	IQL	$\mathcal{X}$-QL	(Gain)
Gym	halfcheetah-medium-v2	42.6	42.5	42.6	43.5	**48.4**	**48.3**	44.0	47.4	**48.3**	- 0.1
	hopper-medium-v2	52.9	56.9	67.6	57.0	59.6	59.3	58.5	66.3	**75.6**	**+ 8.0**
	walker2d-medium-v2	75.3	75.0	74.0	72.4	81.8	83.7	72.5	78.3	**86.0**	+ 4.2
	halfcheetah-medium-replay-v2	36.6	40.6	36.6	40.5	38.1	44.6	**45.5**	44.2	**45.4**	- 0.1
	hopper-medium-replay-v2	18.1	75.9	82.7	37.2	97.5	60.9	95.0	94.7	**102.4**	+ 4.9
	walker2d-medium-replay-v2	26.0	62.5	66.6	27.0	49.5	81.8	77.2	73.9	**86.1**	+ 4.3
	halfcheetah-medium-expert-v2	55.2	92.9	86.8	42.8	93.4	90.7	91.6	86.7	**94.2**	+ 0.8
	hopper-medium-expert-v2	52.5	110.9	107.6	55.8	103.3	98.0	105.4	91.5	**112.3**	+ 1.4
	walker2d-medium-expert-v2	107.5	109.0	108.1	74.5	**113.0**	110.1	108.8	109.6	**112.7**	- 0.3
AntMaze	antmaze-umaze-v0	54.6	62.8	59.2	56.7	64.3	78.6	74.0	87.5	**97.1**	**+ 9.6**
	antmaze-umaze-diverse-v0	45.6	50.2	53.0	49.3	60.7	71.4	84.0	62.2	**85.2**	+ 1.2
	antmaze-medium-play-v0	0.0	5.4	0.0	0.0	0.3	10.6	61.2	71.2	**83.2**	**+ 12.0**
	antmaze-medium-diverse-v0	0.0	9.8	0.0	0.7	0.0	3.0	53.7	70.0	**78.5**	**+ 8.5**
	antmaze-large-play-v0	0.0	0.0	0.0	0.0	0.0	0.2	15.8	39.6	**48.7**	**+ 9.1**
	antmaze-large-diverse-v0	0.0	6.0	0.0	1.0	0.0	0.0	14.9	47.5	**53.1**	+ 5.6
Franka	kitchen-complete-v0	65.0	-	-	-	-	-	43.8	62.5	**82.4**	**+ 17.4**
	kitchen-partial-v0	38.0	-	-	-	-	-	49.8	46.3	**75.0**	**+ 28.7**
	kitchen-mixed-v0	51.5	-	-	-	-	-	51.0	51.0	**62.5**	**+ 11.0**
	runtime	10m	10m	960m	20m	20m	20m	80m	20m	**10m***	

* We see very fast convergence for our method and training for 0.5M steps suffices on most tasks (others are with 1M steps).

Our offline approach strongly outperforms prior methods [6, 18, 19, 17, 9] by *10-20 absolute points* in many environments reaching the new *state-of-the-art* on the D4RL benchmark, as shown in Table 1. We particularly see large improvements on the AntMaze tasks, which require a significant amount of "stitching" between trajectories [17] and get **double-digit** improvement on the challenging Franka tasks. We find performance on the Gym locomotion tasks to be already largely saturated. Moreover, our method convergences significantly faster than IQL, and we show learning curves in Appendix D. $\mathcal{X}$–QL can be easily fine-tuned using online data to attain even higher performance as shown in Table 2, surpassing the final performance of prior works by a wide margin. On a variety of AntMaze tasks, our offline results *before* fine-tuning are on par-with IQL's performance *after* online fine-tuning. Like Kostrikov et al. [17] we tune β for different environments, which corresponds to the weight on our conservative KL penalty.

4.2 Online RL

We compare ExtremeQ variants of SAC [14] and TD3 [10], denoted $\mathcal{X}$-SAC and $\mathcal{X}$-TD3, to their vanilla versions on tasks in the DM Control, shown in Figure 3. Across all tasks an ExtremeQ variant matches or surpasses the performance of baselines. We see particularly large gains in the Hopper environment, and more significant gains in comparison to TD3 overall. Consistent with SAC [14], we find the temperature β needs to be tuned for different environments with different reward scales and sparsity. A core component of TD3 introduced by [10] is Double Q-Learning, which takes the minimum of two Q functions to remove overestimate bias in the Q-target. As we assume errors to be Gumbel distributed, we expect our $\mathcal{X}$-variants to be more robust to such errors. In all environments except Cheetah Run, our $\mathcal{X}$-TD3 without the Double-Q trick performs better than standard TD3. Interestingly, we find that in the DM Control environments Double-Q learning did not boost performance across the board. While the gains from applying Extreme-Q learning are modest in the online setting, none of our methods require acess to the policy distribution $\pi(a|s)$. In particular, $\mathcal{X}$-SAC only requires samples from π_ψ, unlike regular SAC which incorporates the log probabilities of π_ψ into value estimates.

Table 2: Finetuning results on the AntMaze environments

Dataset	CQL	IQL	$\mathcal{X}$-QL
umaze-v0	70.1 → **99.4**	86.7 → 96.0	**96.1** → **99.6**
umaze-diverse-v0	31.1 → **99.4**	75.0 → 84.0	**82.5** → **99.0**
medium-play-v0	23.0 → 0.0	72.0 → 95.0	**80.2** → **97.0**
medium-diverse-v0	23.0 → 32.3	68.3 → 92.0	**74.6** → **97.1**
large-play-v0	1.0 → 0.0	25.5 → 46.0	**45.1** → **59.3**
large-diverse-v0	1.0 → 0.0	42.6 → 60.7	**52.2** → **82.1**

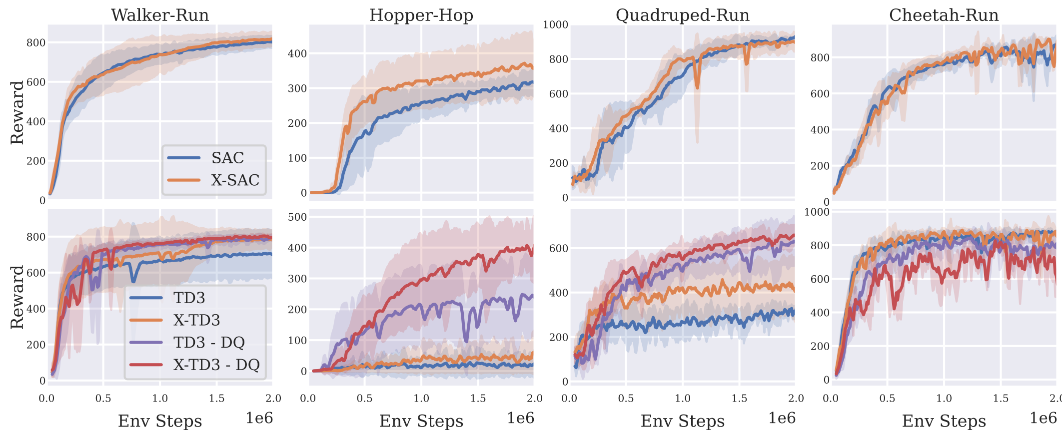

Figure 3: Results on the DM Control for SAC and TD3 based versions of ExtremeQ Learning.

5 Related Work

Our approach builds on top of literature in online & offline RL, and here we review the most salient works. It is worth noting that the inspiration for our novel framework comes from works on choice theory in econometrics [32, 24], and our Gumbel loss was motivated by the work of IQ-Learn [12].

Online RL. Our work bridges the theoretical gap between RL and Max-Ent RL by introducing our Gumbel loss function. Unlike past work in MaxEnt RL [14, 7], our method does not require explicit entropy estimation and instead addresses the problem of obtaining soft-value estimates (LogSumExp) in high-dimensional or continuous spaces [38] by directly modeling them via our proposed Gumbel loss, which to our knowledge has not previously been used in RL. Nevertheless, our loss objective is intrinsically linked to the KL divergence, and similar objectives have been used for mutual information estimation [31]. IQ-Learn [12] - an avant-garde approach to Imitation Learning (IL) - introduced the same loss in IL to obtain an unbiased dual form for the reverse KL-divergence between an expert and policy distribution. Prior work in RL has also examined using other types of loss functions [3] or other formulations of the $\arg\max$ in order to ease optimization [2]. Distinct from most off-Policy RL Methods [22, 10, 14], we directly model $\mathcal{B}^*$ like [13, 16] but attain significantly more stable results.

Offline RL. Prior works in offline RL can largely be categorized as relying on constrained or regularized Q-learning [39, 9, 11, 18, 19, 27], or extracting a greedy policy from the known behavior policy [30, 5, 6]. Most similar to our work, IQL [17] fits expectiles of the Q-function of the behavior policy, but is not motivated to solve a particular problem or remain conservative. On the other hand, conservatism in CQL [19] is motivated by lower-bounding the Q-function's value. Our method shares the best of both worlds – like IQL we do not evaluate the Q-function on potentially out of distribution actions and like CQL we enjoy the benefits of conservatism. Compared to CQL, our approach uses a KL constraint with the behavior policy, and for the first time successfully extends soft-Q learning to offline RL without needing a policy or explicit entropy values.

6 Conclusion

We propose Extreme Q-Learning, a new framework for MaxEnt RL that directly estimates the optimal Bellman backup $\mathcal{B}^*$ without relying on explicit access to a policy. Theoretically, we bridge the gap between the regular, soft, and conservative Q-learning formulations. Empirically, we show that our framework can be used to develop simple SOTA RL algorithms. A number of future directions remain – improving stability in training with exponential loss functions, better selection of policy distributions μ in the online setting, and integrating automatic tuning methods for β. Finally, we hope that our framework can find uses beyond RL in general Machine Learning using our Gumbel loss for estimation of log-partition functions.

References

[1] A. Ahmadi-Javid. Entropic value-at-risk: A new coherent risk measure. *Journal of Optimization Theory and Applications*, 155(3):1105–1123, 2012. URL https://EconPapers.repec.org/

RePEc:spr:joptap:v:155:y:2012:i:3:d:10.1007_s10957-011-9968-2. 5

[2] Kavosh Asadi and Michael L Littman. An alternative softmax operator for reinforcement learning. In *International Conference on Machine Learning*, pages 243–252. PMLR, 2017. 9

[3] Joan Bas-Serrano, Sebastian Curi, Andreas Krause, and Gergely Neu. Logistic q-learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3610–3618. PMLR, 2021. 9

[4] M. Bloem and N. Bambos. Infinite time horizon maximum causal entropy inverse reinforcement learning. *53rd IEEE Conference on Decision and Control*, pages 4911–4916, 2014. 2

[5] David Brandfonbrener, Will Whitney, Rajesh Ranganath, and Joan Bruna. Offline rl without off-policy evaluation. *Advances in Neural Information Processing Systems*, 34:4933–4946, 2021. 9

[6] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34, 2021. 8, 9

[7] Benjamin Eysenbach and Sergey Levine. If maxent {rl} is the answer, what is the question?, 2020. URL https://openreview.net/forum?id=SkxcZCNKDS. 9

[8] R. A. Fisher and L. H. C. Tippett. Limiting forms of the frequency distribution of the largest or smallest member of a sample. *Mathematical Proceedings of the Cambridge Philosophical Society*, 24(2):180–190, 1928. doi: 10.1017/S0305004100015681. 3

[9] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021. 8, 9

[10] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *ArXiv*, abs/1802.09477, 2018. 1, 8, 9, 15, 21

[11] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR, 2019. 9

[12] Divyansh Garg, Shuvam Chakraborty, Chris Cundy, Jiaming Song, and Stefano Ermon. Iq-learn: Inverse soft-q learning for imitation. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=Aeo-xqtb5p. 6, 9, 20

[13] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. 2017. 3, 9, 21

[14] 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*, pages 1861–1870. PMLR, 2018. 1, 3, 4, 7, 8, 9

[15] Tamir Hazan and Tommi Jaakkola. On the partition function and random maximum a-posteriori perturbations. *arXiv preprint arXiv:1206.6410*, 2012. 3

[16] Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. *Advances in neural information processing systems*, 28, 2015. 9

[17] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021. 7, 8, 9, 12, 21

[18] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019. 8, 9

[19] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191, 2020. 1, 6, 8, 9, 19

[20] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018. 7

[21] Qing Li. Continuous control benchmark of deepmind control suite and mujoco. https://github.com/LQNew/Continuous_Control_Benchmark, 2021. 22

[22] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015. 9

[23] R.Duncan Luce. The choice axiom after twenty years. *Journal of Mathematical Psychology*, 15 (3):215–233, 1977. ISSN 0022-2496. doi: https://doi.org/10.1016/0022-2496(77)90032-3. URL https://www.sciencedirect.com/science/article/pii/0022249677900323. 3, 15

[24] Daniel McFadden. Conditional logit analysis of qualitative choice behavior. 1972. 2, 3, 9, 14, 15

[25] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 1

[26] Alexander McFarlane Mood. Introduction to the theory of statistics. 1950. 3

[27] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020. 7, 9

[28] Gergely Neu, Anders Jonsson, and V. Gómez. A unified view of entropy-regularized markov decision processes. *ArXiv*, abs/1705.07798, 2017. 3

[29] George Papandreou and Alan L Yuille. Gaussian sampling by local perturbations. *Advances in Neural Information Processing Systems*, 23, 2010. 3

[30] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019. 9

[31] Ben Poole, Sherjil Ozair, Aäron van den Oord, Alexander A. Alemi, and G. Tucker. On variational bounds of mutual information. In *ICML*, 2019. 9

[32] John Rust. Structural estimation of markov decision processes. In R. F. Engle and D. McFadden, editors, *Handbook of Econometrics*, volume 4, chapter 51, pages 3081–3143. Elsevier, 1 edition, 1986. URL https://editorialexpress.com/jrust/papers/handbook_ec_v4_rust.pdf. 2, 3, 9

[33] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015. 1, 7

[34] Slavko Simić. On a new converse of jensen's inequality. *Publications De L'institut Mathematique*, 85:107–110, 01 2009. doi: 10.2298/PIM0999107S. 17

[35] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018. 4

[36] Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. 1999. 15

[37] Nino Vieillard, Tadashi Kozuno, Bruno Scherrer, Olivier Pietquin, Rémi Munos, and Matthieu Geist. Leverage the average: an analysis of kl regularization in rl. *34th Conference on Neural Information Processing Systems*, 2020. 7

[38] Nino Vieillard, Marcin Andrychowicz, Anton Raichuk, Olivier Pietquin, and Matthieu Geist. Implicitly regularized rl with implicit q-values. *arXiv preprint arXiv:2108.07041*, 2021. 9

[39] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019. 9

[40] Denis Yarats and Ilya Kostrikov. Soft actor-critic (sac) implementation in pytorch. `https://github.com/denisyarats/pytorch_sac`, 2020. 21, 22

[41] G. Alastair Young. High-dimensional statistics: A non-asymptotic viewpoint, martin j. wainwright, cambridge university press, 2019, xvii 552 pages, £57.99, hardback isbn: 978-1-1084-9802-9. *International Statistical Review*, 88(1):258–261, 2020. doi: https://doi.org/10.1111/insr.12370. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/insr.12370`. 3

[42] Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010. 1

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See the results section and intro/abstract
   (b) Did you describe the limitations of your work? [Yes] See the conclusion, section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See the appendix
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See sections 2 and 3
   (b) Did you include complete proofs of all theoretical results? [Yes] We have largely expanded on results included in Section 2 and 3 in the Appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Supplementary material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See the last section of the appendix
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Error bars are shown for Online RL, for offline RL we report the average over seeds as done in [17] which we compare to.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See the last section of the appendix.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cited DM Control, Open AI Gym, D4RL, etc.
   (b) Did you mention the license of the assets? [N/A] All assets have open licences
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Extreme Theory of Q-learning

In this section, we functionally analyze Q-learning using our framework and further develop the Gumbel Error Model (GEM) for MDPs.

A.1 Rust-McFadden Model of MDPs

For an MDP following the Bellman equations, we assume the observed rewards to be stochastic due to an unobserved component of the state. Let $\mathbf{s}$ be the observed state, and $(\mathbf{s}, \mathbf{z})$ be the actual state with hidden component $\mathbf{z}$. Then,

$$Q(\mathbf{s}, \mathbf{z}, \mathbf{a}) = R(\mathbf{s}, \mathbf{z}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim P(\cdot|\mathbf{s},\mathbf{a})}[\mathbb{E}_{\mathbf{z}'|\mathbf{s}'}[V(\mathbf{s}', \mathbf{z}')], \tag{15}$$

$$V(\mathbf{s}, \mathbf{z}) = \max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{z}, \mathbf{a}). \tag{16}$$

Lemma A.1. *Given, 1) conditional independence (CI) assumption that $\mathbf{z}'$ depends only on $\mathbf{s}'$, i.e. $p(\mathbf{s}', z'|\mathbf{s}, \mathbf{z}, \mathbf{a}) = p(\mathbf{z}'|\mathbf{s}')p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ and 2) additive separablity (AS) assumption on the hidden noise: $R(\mathbf{s}, \mathbf{a}, \mathbf{z}) = r(\mathbf{s}, \mathbf{a}) + \epsilon(\mathbf{z}, \mathbf{a})$.*

Then for i.i.d. $\epsilon(\mathbf{z}, \mathbf{a}) \sim \mathcal{G}(0, \beta)$, we recover the soft-Bellman equations for $Q(\mathbf{s}, \mathbf{z}, \mathbf{a}) = q(\mathbf{s}, \mathbf{a}) + \epsilon(\mathbf{z}, \mathbf{a})$ and $v(\mathbf{s}) = \mathbb{E}_z[V(\mathbf{s}, \mathbf{z})]$, with rewards $r(\mathbf{s}, \mathbf{a})$ and entropy regularization β.

Hence, a soft-MDP in MaxEntRL is equivalent to an MDP with an extra hidden variable in the state that introduces i.i.d. Gumbel noise in the rewards and follows the AS+CI conditions.

Proof. We have,

$$q(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim P(\cdot|\mathbf{s},\mathbf{a})}[\mathbb{E}_{\mathbf{z}'|\mathbf{s}'}[V(\mathbf{s}', \mathbf{z}')] \tag{17}$$

$$v(\mathbf{s}) = \mathbb{E}_{\mathbf{z}}[V(\mathbf{s}, \mathbf{z})] = \mathbb{E}_z[\max_{\mathbf{a}}(q(\mathbf{s}, \mathbf{a}) + \epsilon(\mathbf{z}))]. \tag{18}$$

From this, we can get fixed-point equations for q and π,

$$q(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim P(\cdot|\mathbf{s},\mathbf{a})}[\mathbb{E}_{\mathbf{z}'|\mathbf{s}'}[\max_{\mathbf{a}'}(q(\mathbf{s}', \mathbf{a}') + \epsilon(\mathbf{z}', \mathbf{a}'))]], \tag{19}$$

$$\pi(\cdot|s) = \mathbb{E}_{\mathbf{z}}[\operatorname*{argmax}_{\mathbf{a}}(q(\mathbf{s}, \mathbf{a}) + \epsilon(\mathbf{z}, \mathbf{a}))] \in \Delta_{\mathcal{A}}, \tag{20}$$

where $\Delta_{\mathcal{A}}$ is the set of all policies.

Now, let $\epsilon(\mathbf{z}, \mathbf{a}) \sim \mathcal{G}(0, \beta)$ and assumed independent for each $(\mathbf{z}, \mathbf{a})$ (or equivalently $(\mathbf{s}, \mathbf{a})$ due to the CI condition). Then we can use the Gumbel-Max trick to recover the soft-Bellman equations for $q(\mathbf{s}, \mathbf{a})$ and $v(\mathbf{s})$ with rewards $r(\mathbf{s}, \mathbf{a})$:

$$q(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim P(\cdot|\mathbf{s},\mathbf{a})}[\mathbb{L}^{\beta}_{\mathbf{a}'}[q(\mathbf{s}', \mathbf{a}')]], \tag{21}$$

$$\pi(\cdot|s) = \operatorname*{softmax}_{\mathbf{a}}(q(\mathbf{s}, \mathbf{a})). \tag{22}$$

Thus, we have that the soft-Bellman optimality equation and related optimal policy can arise either from the entropic regularization viewpoint or from the Gumbel error viewpoint for an MDP.

Corollary A.1.1. *Converse: An MDP following the Bellman optimality equation and having a policy that is* softmax *distributed, necessarily has any i.i.d. noise in the rewards due to hidden state variables be Gumbel distributed, given the AS+CI conditions hold.*

Proof. McFadden [24] proved this converse in his seminal work on discrete choice theory, that for i.i.d. ϵ satisfiying Equation 19 with a choice policy $\pi \sim \mathrm{softmax}$ has ϵ be Gumbel distributed. And we show a proof here similar to the original for MDPs.

Considering Equation 20, we want $\pi(a|s)$ to be $\mathrm{softmax}$ distributed. Let ϵ have an unknown CDF F and we consider there to be N possible actions. Then,

$$\begin{aligned} P(\operatorname*{argmax}_{\mathbf{a}}(q(\mathbf{s}, \mathbf{a}) + \epsilon(z, \mathbf{a})) = \mathbf{a}_i|\mathbf{s}, \mathbf{z}) &= P(q(\mathbf{s}, \mathbf{a}_i) + \epsilon(\mathbf{z}, \mathbf{a}_i) \geq q(\mathbf{s}, \mathbf{a}_j) + \epsilon(z, \mathbf{a}_j)\ \forall i \neq j\ |\mathbf{s}, \mathbf{z}) \\ &= P(\epsilon(\mathbf{z}, \mathbf{a}_j) - \epsilon(\mathbf{z}, \mathbf{a}_i) \leq q(\mathbf{s}, \mathbf{a}_i) - q(\mathbf{s}, \mathbf{a}_j)\ \forall i \neq j\ |\mathbf{s}, \mathbf{z}) \end{aligned}$$

Simplifying the notation, we write $\epsilon(\mathbf{z}, \mathbf{a}_i) = \epsilon_i$ and $q(\mathbf{s}, \mathbf{a}_i) = q_i$. Then $\epsilon_1, ..., \epsilon_N$ has a joint CDF G:

$$G(\epsilon_1, ..., \epsilon_N) = \prod_{j=1}^{N} P(\epsilon_j \leq \epsilon_i + q_i - q_j) = \prod_{j=1}^{N} F(\epsilon_i + q_i - q_j)$$

and we can get the required probability $\pi(i)$ as:

$$\pi(i) = \int_{\varepsilon=-\infty}^{+\infty} \prod_{j=1, j\neq i}^{N} F(\varepsilon + q_i - q_j) dF(\varepsilon) \tag{23}$$

For $\pi = \mathrm{softmax}(q)$, McFadden [24] proved the uniqueness of F to be the Gumbel CDF, assuming translation completeness property to hold for F. Later this uniqueness was shown to hold in general for any $N \geq 3$ [23]. $\square$

A.2 Gumbel Error Model (GEM) for MDPs

To develop our Gumbel Error Model (GEM) for MDPs under functional approximation as in Section 3.1, we follow our simplified scheme of M independent estimators $\hat{Q}$, which results in the following equation over $\bar{Q} = \mathbb{E}[\hat{Q}]$:

$$\bar{Q}_{t+1}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}'|\mathbf{s},\mathbf{a}}[\mathbb{E}_{\epsilon_t}[\max_{\mathbf{a}'}(\bar{Q}_t(\mathbf{s}', \mathbf{a}') + \epsilon_t(\mathbf{s}', \mathbf{a}'))]]. \tag{24}$$

Here, the maximum of random variables will generally be greater than the true max, i.e. $\mathbb{E}_\epsilon[\max_{\mathbf{a}'}(\bar{Q}(\mathbf{s}', \mathbf{a}') + \epsilon(\mathbf{s}', \mathbf{a}'))] \geq \max_{\mathbf{a}'} \bar{Q}(\mathbf{s}', \mathbf{a}')$ [36]. As a result, even initially zero-mean error can cause Q updates to propagate consistent overestimation bias through the Bellman equation. This is a known issue with function approximation in RL [10].

Now, we can use the Rust-McFadden model from before. To account for the stochasticity, we consider extra unobserved state variables z in the MDP to be the model parameters θ used in the functional approximation. The errors from functional approximation ϵ_t can thus be considered as noise added in the reward. Here, CI condition holds as ϵ is separate from the dynamics and becomes conditionally independent for each state-action pair and AS condition is implied. Then for $\bar{Q}$ satisfying Equation 24, we can apply the McFadden-Rust model, which implies that for the policy to be soft-optimal i.e. a softmax over $\bar{Q}$, ϵ will be Gumbel distributed.

Conversely, for the i.i.d. $\epsilon \sim \mathcal{G}$, $\bar{Q}(\mathbf{s}, \mathbf{a})$ follows the soft-Bellman equations and $\pi(\mathbf{a}|\mathbf{s}) = \mathrm{softmax}(Q(\mathbf{s}, \mathbf{a}))$.

This indicates an optimality condition on the MDP – for us to eventually attain the optimal $\mathrm{softmax}$ policy in the presence of functional boostrapping (Equation 24), the errors should follow the Gumbel distribution.

A.2.1 Time Evolution of Errors in MDPs under Deterministic dynamics

In this section, we characterize the time evolution of errors in an MDP using GEM. We assume deterministic dynamics to simplify our analysis.

We suppose that we know the distribution of Q-values at time t and model the evolution of this distribution through the Bellman equations. Let $Z_t(\mathbf{s}, \mathbf{a})$ be a random variable sampled from the distribution of Q-values at time t, then the following Bellman equation holds:

$$Z_{t+1}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \max_{\mathbf{a}'} Z_t(\mathbf{s}', \mathbf{a}'). \tag{25}$$

Here, $Z_{t+1}(\mathbf{s}, \mathbf{a}) = \max_{\mathbf{a}'}[r(\mathbf{s}, \mathbf{a}) + \gamma Z_t(\mathbf{s}', \mathbf{a}')]$ is a maximal distribution and based on EVT should eventually converge to an extreme value distribution, which we can model as a Gumbel.

Concretely, let's assume that we fix $Z_t(\mathbf{s}, \mathbf{a}) \sim \mathcal{G}(Q_t(\mathbf{s}, \mathbf{a}), \beta)$ for some $Q_t(\mathbf{s}, \mathbf{a}) \in \mathbb{R}$ and $\beta > 0$. Furthermore, we assume that the Q-value distribution is jointly independent over different state-actions i.e. $Z(\mathbf{s}, \mathbf{a})$ is independent from $Z(\mathbf{s}', \mathbf{a}')$ for $\forall$ $(\mathbf{s}, \mathbf{a}) \neq (\mathbf{s}', \mathbf{a}')$. Then $\max_{\mathbf{a}'} Z_t(\mathbf{s}', \mathbf{a}') \sim \mathcal{G}(V(\mathbf{s}'), \beta)$ with $V(\mathbf{s}) = \mathbb{L}^\beta_{\mathbf{a}}[Q(\mathbf{s}, \mathbf{a})]$ using the Gumbel-max trick.

Then substituting in Equation 25 and rescaling Z_t with γ, we get:

$$Z_{t+1}(\mathbf{s},\mathbf{a}) \sim \mathcal{G}\left(r(\mathbf{s},\mathbf{a}) + \gamma \mathbb{L}^{\beta}_{\mathbf{a}'}[Q(\mathbf{s}',\mathbf{a}')], \gamma\beta\right). \tag{26}$$

So very interestingly the Q-distribution becomes a Gumbel process, where the location parameter $Q(\mathbf{s},\mathbf{a})$ follows the optimal soft-Bellman equation. Similarly, the temperature scales as $\gamma\beta$ and the distribution becomes sharper after every timestep.

After a number of timesteps, we see that $Z(\mathbf{s},\mathbf{a})$ eventually collapses to the Delta distibution over the unique contraction $Q^*(\mathbf{s},\mathbf{a})$. Here, γ controls the rate of decay of the Gumbel distribution into the collapsed Delta distribution. Thus we get the expected result in deterministic dynamics that the optimal Q-function will be deterministic and its distribution will be peaked.

So if a Gumbel error enters into the MDP through a functional error or some other source at a timestep t in some state s, it will trigger off an wave that propagates the Gumbel error into its child states following Equation 26. Thus, this Gumbel error process will decay at a γ rate every timestep and eventually settle down with Q-values reaching the the steady solution Q^*. The variance of this Gumbel process given as $\frac{\pi^2}{6}\beta^2$ will decay as γ^2, similarly the bias will decay as γ-contraction in the $\mathcal{L}^\infty$ norm.

Hence, GEM gives us an analytic characterization of error propogation in MDPs under deterministic dynamics.

Neertheless under stochastic dynamics, characterization of errors using GEM becomes non-trivial as Gumbel is not mean-stable unlike the Gaussian distribution. We hypothesise that the errors will follow some mix of Gumbel-Gaussian distributions, and leave this characterization as a future open direction.

A.3 Extreme Q-Learning ($\mathcal{X}$-QL)

For the soft-Bellman equation given as:

$$Q(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}'\sim P(\cdot|\mathbf{s},\mathbf{a})} V(\mathbf{s}), \tag{27}$$

$$V(\mathbf{s}) = \mathbb{L}^{\beta}_{\mu(\cdot|s)}(Q(\mathbf{s},\mathbf{a})), \tag{28}$$

we have the fixed-point characterization, that can be found with a recurrence:

$$V(\mathbf{s}) = \mathbb{L}^{\beta}_{\mu(\cdot|s)}\left(r(\mathbf{s},\mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}'\sim P(\cdot|\mathbf{s},\mathbf{a})} V(\mathbf{s})\right). \tag{29}$$

We discuss the case with stochastic dynamics in the main paper which requires the estimation of $\mathcal{B}^*$. Under deterministic dynamic this can be avoided and we detail our method below.

A.3.1 $\mathcal{X}$-QL in Deterministic dynamics

For deterministic dynamics we can avoid having an expectation over the next states to simplify the Bellman equation.

We can develop two simple algorithms for this case without needing $\mathcal{B}^*$.

Value Iteration. We can write the value-iteration objective as:

$$Q(\mathbf{s},\mathbf{a}) \leftarrow r(\mathbf{s},\mathbf{a}) + \gamma V_\theta(\mathbf{s}'), \tag{30}$$

$$\mathcal{J}(\theta) = \mathbb{E}_{s\sim\rho_\mu, a\sim\mu(\cdot|s)}\left[e^{(Q(\mathbf{s},\mathbf{a})-V_\theta(\mathbf{s}))/\beta} - (Q(\mathbf{s},\mathbf{a}) - V_\theta(\mathbf{s}))/\beta - 1\right]. \tag{31}$$

Here, we learn a single model of the values $V_\theta(\mathbf{s})$ to directly solve Equation 29. For the current value estimate $V_\theta(\mathbf{s})$, we calculate targets $r(\mathbf{s},\mathbf{a}) + \gamma V_\theta(\mathbf{s})$ and find a new estimate $V'_\theta(\mathbf{s})$ by fitting $\mathbb{L}^{\beta}_{\mu}$ with our objective $\mathcal{J}$. Using our Gumbel Regression framework, we can guarantee that as $\mathcal{J}$ finds a consistent estimate of the $\mathbb{L}^{\beta}_{\mu}$, and $V_\theta(\mathbf{s})$ will converge to the optimal $V(\mathbf{s})$ upto some sampling error.

Q-Iteration. Alternatively, we can develop a Q-iteration objective solving the recurrence:

$$Q_{t+1}(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \gamma \mathbb{L}^{\beta}_{\mathbf{a}'\sim\mu}\left[Q_t(\mathbf{s}',\mathbf{a}')\right] \tag{32}$$

$$= r(\mathbf{s},\mathbf{a}) + \mathbb{L}^{\gamma\beta}_{\mathbf{a}'\sim\mu}\left[\gamma Q_t(\mathbf{s}',\mathbf{a}')\right] \tag{33}$$

$$= \mathbb{L}^{\gamma\beta}_{\mathbf{a}'\sim\mu}\left[r(\mathbf{s},\mathbf{a}) + \gamma Q_t(\mathbf{s}',\mathbf{a}')\right]. \tag{34}$$

where we can rescale β to $\gamma\beta$ to move $\mathbb{L}$ out.

This gives the objective:

$$Q^t(\mathbf{s},\mathbf{a}) \leftarrow r(\mathbf{s},\mathbf{a}) + \gamma Q_\theta(\mathbf{s}',\mathbf{a}'), \tag{35}$$

$$\mathcal{J}(Q_\theta) = \mathbb{E}_{\mu(\mathbf{s},a,\mathbf{s}')}\left[e^{(Q^t(\mathbf{s},\mathbf{a})-Q_\theta(\mathbf{s},\mathbf{a}))/\gamma\beta} - (Q^t(\mathbf{s},\mathbf{a}) - Q_\theta(\mathbf{s},\mathbf{a}))/\gamma\beta - 1\right]. \tag{36}$$

Thus, this gives a method to directly estimate Q_θ without learning values, and forms our $\mathcal{X}$-TD3 method in the main paper. Note, that β is a hyperparameter, so we can use an alternative hyperparameter $\beta' = \gamma\beta$ to simplify the above.

We can formalize this as a Lemma in the deterministic case:

Lemma A.2. *Let*

$$\mathcal{J}(\mathcal{T}_\mu Q - Q') = \mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}',\mathbf{a}'\sim\mu}\left[e^{(\mathcal{T}_\mu Q(\mathbf{s},\mathbf{a})-Q'(\mathbf{s},\mathbf{a}))/\gamma\beta} - (\mathcal{T}_\mu Q(\mathbf{s},\mathbf{a}) - Q'(\mathbf{s},\mathbf{a}))/\gamma\beta - 1\right].$$

where $\mathcal{T}_\mu$ is a linear operator that maps Q from current $(\mathbf{s},\mathbf{a})$ to the next $(\mathbf{s}',\mathbf{a}')$: $\mathcal{T}_\mu Q(\mathbf{s},\mathbf{a}) := r(\mathbf{s},\mathbf{a}) + \gamma Q(\mathbf{s}',\mathbf{a}')$

Then we have $\mathcal{B}^* Q^t = \underset{Q'\in\Omega}{\operatorname{argmin}}\, \mathcal{J}(\mathcal{T}_\mu Q^t - Q')$, where Ω is the space of Q-functions.

Proof. We use that in deterministic dynamics,

$$\mathbb{L}^{\gamma\beta}_{\mathbf{a}'\sim\mu}[\mathcal{T}_\mu Q(\mathbf{s},\mathbf{a})] = r(\mathbf{s},\mathbf{a}) + \gamma\mathbb{L}^{\beta}_{\mathbf{a}'\sim\mu}[Q(\mathbf{s}',\mathbf{a}')] = \mathcal{B}^* Q(\mathbf{s},\mathbf{a})$$

Then solving for the unique minima for $\mathcal{J}$ establishes the above results.

Thus, optimizing $\mathcal{J}$ with a fixed-point is equivalent to Q-iteration with the Bellman operator.

□

B Gumbel Regression

We characterize the concentration bounds for Gumbel Regression in this section. First, we bound the bias on applying $\mathbb{L}^\beta$ to inputs containing errors. Second, we bound the error due to an empirical $\hat{\mathbb{L}}^\beta$ over finite N samples.

B.1 Overestimation Bias

Let $\hat{Q}(\mathbf{s},\mathbf{a})$ be a random variable representing a Q-value estimate for a state and action pair $(\mathbf{s},\mathbf{a})$. We assume that it is an unbiased estimate of the true Q-value $Q(\mathbf{s},\mathbf{a})$ with $\mathbb{E}[\hat{Q}(\mathbf{s},\mathbf{a})] = Q(\mathbf{s},\mathbf{a})$. Let $Q(\mathbf{s},\mathbf{a}) \in [-Q_{max}, Q_{max}]$

Then, $V(\mathbf{s}) = \mathbb{L}^\beta_{a\sim\mu} Q(\mathbf{s},\mathbf{a})$ is the true value function, and $\hat{V}(\mathbf{s}) = \mathbb{L}^\beta_{a\sim\mu}\hat{Q}(\mathbf{s},\mathbf{a})$ is its estimate.

Lemma B.1. *We have $V(\mathbf{s}) \le \mathbb{E}[\hat{V}(\mathbf{s})] \le \mathbb{E}_{a\sim\mu}[Q(\mathbf{s},\mathbf{a})] + \beta\log\cosh(Q_{max}/\beta)$.*

Proof. The lower bound $V(\mathbf{s}) \le \mathbb{E}[\hat{V}(\mathbf{s})]$ is easy to show using Jensen's Inequality as log_sum_exp is a convex function.

For the upper bound, we can use a reverse Jensen's inequality [34] that for any convex mapping f on the interval $[a,b]$ it holds that:

$$\sum_i p_i f(x_i) \le f\left(\sum_i p_i x_i\right) + f(a) + f(b) - f\left(\frac{a+b}{2}\right)$$

Setting $f = -\log(\cdot)$ and $x_i = e^{\hat{Q}(\mathbf{s},\mathbf{a})/\beta}$, we get:

$$\mathbb{E}_{\mathbf{a}\sim\mu}[-\log(e^{\hat{Q}(\mathbf{s},\mathbf{a})/\beta})] \le -\log(\mathbb{E}_{\mathbf{a}\sim\mu}[e^{\hat{Q}(\mathbf{s},\mathbf{a})/\beta}]) - \log(e^{Q_{max}/\beta}) - \log(e^{-Q_{max}/\beta}) + \log\left(\frac{e^{Q_{max}/\beta} + e^{-Q_{max}/\beta}}{2}\right)$$

On simplifying,

$$\hat{V}(\mathbf{s}) = \beta \log(\mathbb{E}_{\mathbf{a}\sim\mu} e^{\hat{Q}(\mathbf{s},\mathbf{a})/\beta}) \le \mathbb{E}_{\mathbf{a}\sim\mu}[\hat{Q}(\mathbf{s},\mathbf{a})] + \beta \log\cosh(Q_{max}/\beta)$$

Taking expectations on both sides, $\mathbb{E}[\hat{V}(\mathbf{s})] \le \mathbb{E}_{\mathbf{a}\sim\mu}[Q(\mathbf{s},\mathbf{a})] + \beta\log\cosh(Q_{max}/\beta)$. This gives an estimate of how much the LogSumExp overestimates compared to taking the expectation over actions for random variables $\hat{Q}$. This bias monotonically decreases with β, with $\beta = 0$ having a max bias of Q_{max} and for large β decaying as $\frac{1}{2\beta}Q_{max}^2$.

$\square$

B.2 PAC learning Bounds for Gumbel Regression

Lemma B.2. $\exp(\hat{\mathbb{L}}^\beta(X)/\beta)$ *over a finite* N *samples is an unbiased estimator for the partition function* $Z^\beta = \mathbb{E}\left[e^{X/\beta}\right]$ *and with a probability at least* $1-\delta$ *it holds that:*

$$\exp(\hat{\mathbb{L}}^\beta(X)/\beta) \le Z^\beta + \sinh(X_{max}/\beta)\sqrt{\frac{2\log(1/\delta)}{N}}.$$

Similarly, $\hat{\mathbb{L}}^\beta(X)$ *over a finite* N *samples is a consistent estimator of* $\mathbb{L}^\beta(X)$ *and with a probability at least* $1-\delta$ *it holds that:*

$$\hat{\mathbb{L}}^\beta(X) \le \mathbb{L}^\beta(X) + \frac{\beta\sinh(X_{max}/\beta)}{Z^\beta}\sqrt{\frac{2\log(1/\delta)}{N}}.$$

Proof. To prove these concentration bounds, we consider random variables $e^{X_1/\beta}, ..., e^{X_n/\beta}$ with $\beta > 0$, such that $a_i \le X_i \le b_i$ almost surely, i.e. $e^{a_i/\beta} \le e^{X_i/\beta} \le e^{b_i/\beta}$.

We consider the sum $S_n = \sum_{i=1}^N e^{X_i/\beta}$ and use Hoeffding's inequality, so that for all $t > 0$:

$$P\left(S_n - \mathbb{E}S_n \ge t\right) \le \exp\left(\frac{-2t^2}{\sum_{i=1}^n \left(e^{b_i/\beta} - e^{a_i/\beta}\right)^2}\right) \tag{37}$$

To simplify, we let $a_i = -X_{max}$ and $b_i = X_{max}$ for all i. We also rescale t as $t = Ns$, for $s > 0$. Then

$$P\left(S_n - \mathbb{E}S_n \ge Ns\right) \le \exp\left(\frac{-Ns^2}{2\sinh^2(X_{max}/\beta)}\right) \tag{38}$$

We can notice that L.H.S. is same as $P(\exp(\hat{\mathbb{L}}^\beta(X)/\beta) - \exp(\mathbb{L}^\beta(X)/\beta) \ge s)$, which is the required probability we want. Letting the R.H.S. have a value δ, we get

$$s = \sinh(X_{max}/\beta)\sqrt{\frac{2\log(1/\delta)}{N}}$$

Thus, with a probability $1-\delta$, it holds that:

$$\exp(\hat{\mathbb{L}}^\beta(X)/\beta) \le \exp(\mathbb{L}^\beta(X)/\beta) + \sinh(X_{max}/\beta)\sqrt{\frac{2\log(1/\delta)}{N}} \tag{39}$$

Thus, we get a concentration bound on $\exp(\hat{\mathbb{L}}^\beta(X)/\beta)$ which is an unbiased estimator of the partition function $Z^\beta = \exp(\mathbb{L}^\beta(X)/\beta)$. This bound becomes tighter with increasing β, and asymptotically behaves as $\frac{X_{max}}{\beta}\sqrt{\frac{2\log(1/\delta)}{N}}$.

Similarly, to prove the bound on the log-partition function $\hat{\mathbb{L}}^\beta(X)$, we can further take $\log(\cdot)$ on both sides and use the inequality $\log(1+x) \leq x$, to get a direct concentration bound on $\hat{\mathbb{L}}^\beta(X)$,

$$\hat{\mathbb{L}}^\beta(X) \leq \mathbb{L}^\beta(X) + \beta \log\left(1 + \sinh(X_{max}/\beta) e^{-\mathbb{L}^\beta(X)/\beta} \sqrt{\frac{2\log(1/\delta)}{N}}\right) \tag{40}$$

$$= \mathbb{L}^\beta(X) + \beta \sinh(X_{max}/\beta) e^{-\mathbb{L}^\beta(X)/\beta} \sqrt{\frac{2\log(1/\delta)}{N}} \tag{41}$$

$$= \mathbb{L}^\beta(X) + \frac{\beta \sinh(X_{max}/\beta)}{Z^\beta} \sqrt{\frac{2\log(1/\delta)}{N}} \tag{42}$$

This bound also becomes tighter with increasing β, and asymptotically behaves as $\frac{X_{max}}{Z^\beta}\sqrt{\frac{2\log(1/\delta)}{N}}$.

$\square$

C Bridging soft and conservative Q-Learning

C.1 Inherent Convervatism in $\mathcal{X}$-QL

Our method is inherently conservative similar to CQL [19] in that it underestimates the value function (in vanilla Q-learning) $V^\pi(\mathbf{s})$ by $-\beta\ \mathbb{E}_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}\left[\log \frac{\pi(\mathbf{a}|\mathbf{s})}{\pi_{\mathcal{D}}(\mathbf{a}|\mathbf{s})}\right]$, whereas CQL understimates values by a factor $-\beta\ \mathbb{E}_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}\left[\frac{\pi(\mathbf{a}|\mathbf{s})}{\pi_{\mathcal{D}}(\mathbf{a}|\mathbf{s})} - 1\right]$, where $\pi_{\mathcal{D}}$ is the behavior policy. Notice that the underestimation factor transforms V^π in vanilla Q-learning into V^π used in the soft-Q learning formulation. Thus, we observe that KL-regularized Q-learning is inherently conservative, and this conservatism is built into our method.

Furthermore, it can be noted that CQL conservatism can be derived as adding a χ^2 regularization to an MDP and although not shown by the original work [19] or any follow-ups to our awareness, the last term of Eq. 14 in CQL's Appendix B [19], is simply $\chi^2(\pi||\pi_{\mathcal{D}})$ and what the original work refers to as D_{CQL} is actually the χ^2 divergence. Thus, it is possible to show that all the results for CQL hold for our method by simply replacing D_{CQL} with D_{KL} i.e. the χ^2 divergence with the KL divergence everywhere.

We show a simple proof below that D_{CQL} is the χ^2 divergence:

$$\begin{aligned}
D_{CQL}(\pi, \pi_{\mathcal{D}})(\mathbf{s}) &:= \sum_{\mathbf{a}} \pi(\mathbf{a} \mid \mathbf{s}) \left[\frac{\pi(\mathbf{a} \mid \mathbf{s})}{\pi_{\mathcal{D}}(\mathbf{a} \mid \mathbf{s})} - 1\right] \\
&= \sum_{\mathbf{a}} \left(\pi(\mathbf{a} \mid \mathbf{s}) - \pi_{\mathcal{D}}(\mathbf{a} \mid \mathbf{s}) + \pi_{\mathcal{D}}(\mathbf{a} \mid \mathbf{s})\right) \left[\frac{\pi(\mathbf{a} \mid \mathbf{s})}{\pi_{\mathcal{D}}(\mathbf{a} \mid \mathbf{s})} - 1\right] \\
&= \sum_{\mathbf{a}} \left(\pi(\mathbf{a} \mid \mathbf{s}) - \pi_{\mathcal{D}}(\mathbf{a} \mid \mathbf{s})\right) \left[\frac{\pi(\mathbf{a} \mid \mathbf{s}) - \pi_{\mathcal{D}}(\mathbf{a} \mid \mathbf{s})}{\pi_{\mathcal{D}}(\mathbf{a} \mid \mathbf{s})}\right] + \sum_{\mathbf{a}} \pi_{\mathcal{D}}(\mathbf{a} \mid \mathbf{s}) \left[\frac{\pi(\mathbf{a} \mid \mathbf{s})}{\pi_{\mathcal{D}}(\mathbf{a} \mid \mathbf{s})} - 1\right] \\
&= \sum_{\mathbf{a}} \pi_{\mathcal{D}}(\mathbf{a} \mid \mathbf{s}) \left[\frac{\pi(\mathbf{a} \mid \mathbf{s})}{\pi_{\mathcal{D}}(\mathbf{a} \mid \mathbf{s})} - 1\right]^2 + 0 \text{ since, } \sum_{\mathbf{a}} \pi(\mathbf{a} \mid \mathbf{s}) = \sum_{\mathbf{a}} \pi_{\mathcal{D}}(\mathbf{a} \mid \mathbf{s}) = 1 \\
&= \chi^2(\pi(\cdot \mid \mathbf{s}) \mid\mid \pi_{\mathcal{D}}(\cdot \mid \mathbf{s})), \text{ using the definition of chi-square divergence}
\end{aligned}$$

C.2 Why $\mathcal{X}$–QL is better than CQL for offline RL

In light of the above results, we know that CQL adds a χ^2 regularization to the policy π with respect to the behavior policy $\pi_{\mathcal{D}}$, whereas our method does the same using the reverse-KL divergence.

Now, the reverse-KL divergence has a mode-seeking behavior, and thus our method will find a policy that better fits the mode of the behavior policy and is more robust to random actions in the offline dataset. CQL does not have such a property and can be easily affected by noisy actions in the dataset.

C.3 Connection to Dual KL representation

For given distributions μ and π, we can write their KL-divergence using the dual representation proposed by IQ-Learn [12]:

$$D_{KL}(\pi \,||\, \mu) = \max_{x \in \mathbb{R}} \mathbb{E}_{\mu}[-e^{-x}] - \mathbb{E}_{\pi}[x] - 1,$$

which is maximized for $x = -\log(\pi/\mu)$.

We can make a clever substitution to exploit the above relationship. Let $x = (Q - \mathcal{T}^{\pi}\hat{Q}^k)/\beta$ for a variable $Q \in \mathbb{R}$ and a fixed constant $\mathcal{T}^{\pi}\hat{Q}^k$, then on variable substitution we get the equation:

$$\mathbb{E}_{s\sim\rho_\mu}[D_{KL}(\pi(\cdot|\mathbf{s}) \,||\, \mu(\cdot|\mathbf{s}))] = \min_{Q} \mathcal{L}(Q), \text{with}$$

$$\mathcal{L}(Q) = \mathbb{E}_{\mathbf{s}\sim\rho_\mu, \mathbf{a}\sim\mu(\cdot|\mathbf{s})}\left[e^{(\mathcal{T}^{\pi}\hat{Q}^k(\mathbf{s},\mathbf{a}) - Q(\mathbf{s},\mathbf{a}))/\beta}\right] - \mathbb{E}_{\mathbf{s}\sim\rho_\mu, \mathbf{a}\sim\pi(\cdot|\mathbf{s})}[(\mathcal{T}^{\pi}\hat{Q}^k(\mathbf{s},\mathbf{a}) - Q(\mathbf{s},\mathbf{a}))/\beta] - 1$$

This gives us Equation 8 in Section 3.3 of the main paper, and is minimized for $Q = \mathcal{T}^{\pi}\hat{Q}^k - \beta\log(\pi/\mu)$ as we desire. Thus, this lets us transform the regular Bellman update into the soft-Bellman update.

D Broader Impacts

We introduce new methods for offline and online RL. RL has the potential to power modern robotics and decision making systems. Care must be taken, however, to ensure that the objectives of these systems are well aligned with humans so that they can improve lives.

E Experiments

In this section we provide additional experimental results and further details on all experimental procedures.

E.1 $\mathcal{X}$-QL offline Learning Curves

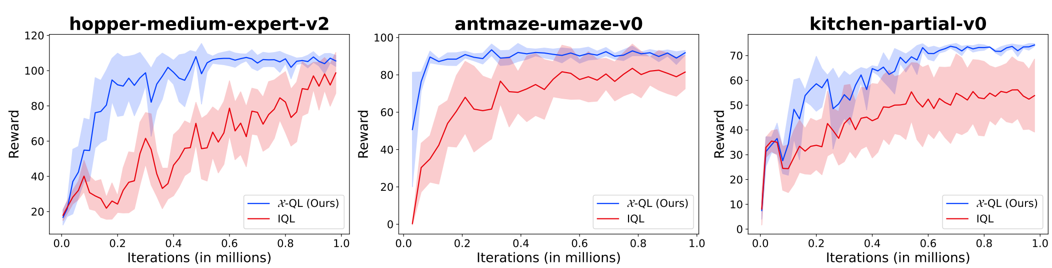

Figure 4: **Offline RL Results**. We show the returns vs number of training iterations. (Averaged over 5 seeds)

E.2 Additional Offline Environments

Table 3: Evaluation on Franka Kitchen and Adroit tasks from D4RL

Dataset	BC	BRAC-p	BEAR	Onestep RL	CQL	IQL	$\mathcal{X}$-QL
pen-human-v0	63.9	8.1	-1.0	-	37.5	71.5	**91.4**
hammer-human-v0	1.2	0.3	0.3	-	4.4	1.4	**5.4**
door-human-v0	2	-0.3	-0.3	-	9.9	4.3	**10.1**
relocate-human-v0	0.1	-0.3	-0.3	-	0.2	0.1	**1.1**
pen-cloned-v0	37	1.6	26.5	**60.0**	39.2	37.3	**58.2**
hammer-cloned-v0	0.6	0.3	0.3	2.1	2.1	2.1	**4.6**
door-cloned-v0	0.0	-0.1	-0.1	0.4	0.4	1.6	**8.9**
relocate-cloned-v0	-0.3	-0.3	-0.3	-0.1	-0.1	-0.2	**-0.1**

E.3 Additional Bellman Error Plots

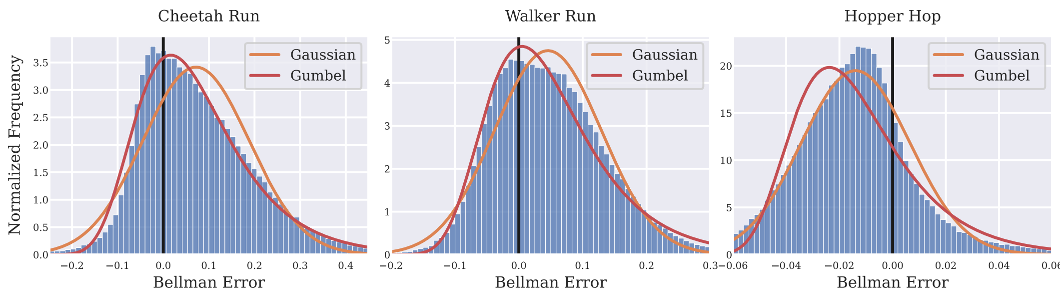

Figure 5: Additional plots of the error distributions for different environments. We find that the Gumbel distribution strongly fit the errors in first two environments, Cheetah and Walker, but provides a worse fit in the Hopper environment. Nonetheless, we see performance improvements in Hopper using our approach.

E.4 Numeric Stability

In practice, a naive implementation of the Gumbel loss function $\mathcal{J}$ from Equation 11 suffers from stability issues due to the exponential term. We found that stabilizing the loss objective was essential for training. Practically, we follow the common max-normalization trick used in softmax computation. This amounts to factoring out $e^{\max_z z}$ from the loss and consequently scaling the gradients. We additionally clip loss inputs that are too large. An example code snippet in Pytorch is included below:

```
def gumbel_loss(pred, label, beta, clip):
    z = (label - pred)/beta
    z = torch.clamp(z, -clip, clip)
    max_z = torch.max(z)
    max_z = torch.where(max_z < -1.0, torch.tensor(-1.0), max_z)
    max_z = max_z.detach() # Detach the gradients
    loss = torch.exp(z - max_z) - z*torch.exp(-max_z) - torch.exp(-max_z)
    return loss.mean()
```

In some experiments we additionally clip the value of the gradients for stability.

E.5 Offline Hyperparameters

We base our implementation of $\mathcal{X}$-QL off of the official implementation of IQL from Kostrikov et al. [17], and use their code to run our experiments. We use the same network architecture and also apply the Double-Q trick. We also apply the same data preprocessing which is described in their appendix. We additionally take their baseline results and use them in Table 1, Table 2, and Table 3 for accurate comparison. Kostrikov et al. [17] did not release code for Adroit Binary-v0, and as such we do not have comparison on these tasks. We contacted the authors regarding the release of these environments and are awaiting a response.

We keep our general algorithm hyper-parameters and evaluation procedure the same but tune β and the gradient clipping value for each environment. For MuJoCo locomotion tasks we average mean returns over 10 evaluation trajectories and 5 random seeds. For the AntMaze tasks, we average over 1000 evaluation trajectories. When reporting results for the Adroit environments we use smoothing of 0.7 as evaluations fluctuate significantly.

E.6 Online Hyperparameters

We base our implementation of SAC off pytorch_sac [40] but modify it to use a Value function as described in Haarnoja et al. [13]. Empirically we see similar performance with and without using the value function, but leave it in for fair comparison against our $\mathcal{X}$-SAC variant. We base our implementation of TD3 on the original author's code from [10]. Like in offline experiments, hyper-parameters were left as default except for β, which we tuned for each environment. In online experiments we clipped the exponential term to be at maximum 8. For SAC we ran three seeds in each environment as it tended to be more stable. For TD3 we ran four. Occasionally, our $\mathcal{X}$-

variants would experience exploding loss terms causing performance to collapse. We re-ran these seeds to obtain runs that did not become numerically unstable. For all online experiments we used an exponential clip value of 8. We verified our SAC results by comparing to [40] and our TD3 results by comparing to [21] . We found that our TD3 implementation performed marginally better overall.

Table 4: Offline RL Hyperparameters used for $\mathcal{X}$–QL

Env	Beta	Grad Clip
halfcheetah-medium-v2	1	7
hopper-medium-v2	5	7
walker2d-medium-v2	10	7
halfcheetah-medium-replay-v2	1	7
hopper-medium-replay-v2	2	7
walker2d-medium-replay-v2	5	7
halfcheetah-medium-expert-v2	1	7
hopper-medium-expert-v2	1	7
walker2d-medium-expert-v2	2	7
antmaze-umaze-v0	2	7
antmaze-umaze-diverse-v0	5	7
antmaze-medium-play-v0	0.8	5
antmaze-medium-diverse-v0	1	5
antmaze-large-play-v0	0.8	5
antmaze-large-diverse-v0	0.5	5
kitchen-complete-v0	2	7
kitchen-partial-v0	5	7
kitchen-mixed-v0	2	7

Table 5: Hyperparameters for online RL Algorithms

Parameter	SAC	TD3
Batch Size	1024	256
Learning Rate	0.0001	0.001
Critic Freq	1	1
Actor Freq	1	2
Actor and Critic Arch	1024, 1024	256, 256
Buffer Size	1,000,000	1,000,000
Actor Noise	Auto-tuned	0.1, 0.05 (Hopper)
Target Noise	–	0.2

Table 6: Values of β used for online experiments

Env	$\mathcal{X}$-SAC	$\mathcal{X}$-TD3	$\mathcal{X}$-TD3 - DQ
Cheetah Run	2	5	4
Walker Run	1	2	4
Hopper Hop	2	2	3
Quadruped Run	5	5	20

E.7 Error Figure 1

Figure 1 was generated by running the SAC implementation from Yarats and Kostrikov [40] for 100,000 timesteps on the Cheetah Run environment and the logging the Bellman errors every 5,000 steps. In particular, the Bellman errors were computed as:

$$r(\mathbf{s},\mathbf{a}) + \gamma Q_{\theta_1}(\mathbf{s}', \pi_\psi(\mathbf{s}')) - Q_{\theta_1}(\mathbf{s},\mathbf{a})$$

In the above equation Q_{θ_1} represents the first of the two Q networks used in the Double Q trick. We do not use target networks to compute the bellman error, and instead compute the fully online quantity. $\pi_\psi(\mathbf{s}')$ represents the mean or deterministic output of the current policy distribution. Also note that the entropy term was not added as we seek to characterize the standard bellman error and not the soft-bellman error. Before generating the plot the errors were clipped to the ranges shown in the plot. This tended prevented over-fitting to large outliers.

E.8 Resources

We ran experiments largely on Nvidia Titan XP GPUs via an internal cluster. Tuning values of β was done largely via trial and error.