Machine Learning (2024) 113:443–488
https://doi.org/10.1007/s10994-023-06458-y

Balancing policy constraint and ensemble size in uncertainty-based offline reinforcement learning

Alex Beeson[1,3] · **Giovanni Montana**[2,3,4]

Received: 21 March 2023 / Revised: 1 August 2023 / Accepted: 17 October 2023 /
Published online: 6 November 2023

Abstract
Offline reinforcement learning agents seek optimal policies from fixed data sets. With environmental interaction prohibited, agents face significant challenges in preventing errors in value estimates from compounding and subsequently causing the learning process to collapse. Uncertainty estimation using ensembles compensates for this by penalising high-variance value estimates, allowing agents to learn robust policies based on data-driven actions. However, the requirement for large ensembles to facilitate sufficient penalisation results in significant computational overhead. In this work, we examine the role of policy constraints as a mechanism for regulating uncertainty, and the corresponding balance between level of constraint and ensemble size. By incorporating behavioural cloning into policy updates, we show empirically that sufficient penalisation can be achieved with a much smaller ensemble size, substantially reducing computational demand while retaining state-of-the-art performance on benchmarking tasks. Furthermore, we show how such an approach can facilitate stable online fine tuning, allowing for continued policy improvement while avoiding severe performance drops.

Keywords Offline reinforcement learning · Ensemble based uncertainty estimation · Behavioural cloning · Online fine-tuning · Pessimism

Editor: Tong Zhang.

✉ Giovanni Montana
g.montana@warwick.ac.uk

Alex Beeson
alex.beeson@warwick.ac.uk

[1] Warwick Medical School, University of Warwick, Coventry, UK

[2] Department of Statistics, University of Warwick, Coventry, UK

[3] WMG, University of Warwick, Coventry, UK

[4] Alan Turing Institute, London, UK

1 Introduction

Reinforcement learning (RL) is concerned with optimising sequential decision-making in dynamic environments (Tesauro, 1995; Sutton & Barto, 2018). Typically, RL is used to train autonomous agents to perform complex tasks that rely on long-term decision making, where the decisions themselves impact future decisions as well as the environment the agent learns in. The agent identifies the optimal sequence of decisions, or actions, through trial-and-error learning, constantly interacting with the environment and adjusting its behaviour based on the rewards received. The end goal is to discover a policy that maximizes environmental rewards. By combining RL with the powerful predictive capabilities of neural networks, deep reinforcement learning has produced notable success in areas such as gaming (Mnih et al., 2013; Hessel et al., 2018), robotics (Kalashnikov et al., 2018; Mahmood et al., 2018) and autonomous driving (Kiran et al., 2022), advancing each year as it garners increasing interest and attention.

Despite the remarkable achievements of RL, its reliance on continuous interaction with the environment restricts its application in areas where data collection is expensive, time-consuming, or hazardous. While simulators can partially alleviate this issue in fields such as robotics and autonomous driving (Todorov et al., 2012), there are numerous situations where these are unavailable, and the trial-and-error nature of RL is clearly unsuitable or even unethical (e.g. in healthcare). Furthermore, these settings often already possess a wealth of data amassed through routine data collection or experimentation, offering a rich information source before an agent even engages in any environmental interaction (Komorowski et al., 2018; Liu et al., 2020; Yu et al., 2021).

The ambition to extend RL into such domains has given rise to offline reinforcement learning (offline-RL) (Lange et al., 2012), a paradigm where agents are restricted from interacting with the environment and must learn exclusively from pre-existing interactions. Conventional RL algorithms typically falter in this offline setting, as the primary method for rectifying errors in action value estimates (i.e. online interaction) is no longer available. This often leads to a complete collapse of the learning process as these errors propagate and compound during training (Fujimoto et al., 2019). Essentially, it is difficult for an agent to accurately assess the value of actions never encountered before, undermining the process of learning a policy based on value estimation.

The most common approach for overcoming this problem is to perform some kind of regularisation during training, encouraging updates during policy evaluation and/or policy improvement to stay close to actions in the underlying data (Levine et al., 2020). To date, numerous approaches have been proposed, ranging from methods that directly target the policy and/or value estimates (Kumar et al., 2019; Wu et al., 2019; Kumar et al., 2020; Nair et al., 2020; Kostrikov et al., 2021; Brandfonbrener et al., 2021) through to those which incorporate models of the environment (Kidambi et al., 2020; Yu et al., 2021; Argenson & Dulac-Arnold, 2020; Janner et al., 2022), each with their own strengths and weaknesses in terms of performance, computational efficiency, reproducibility, hyperparameter optimisation and ease of implementation.

One such approach centres around uncertainty quantification with respect to the estimated value of actions (Abdar et al., 2021). For actions absent in data, commonly referred to as out-of-distrubtion (OOD) actions, values estimates are subject to higher uncertainty than those present in data. In online settings, this is often used to improve exploration by being optimistic in the face of uncertainty (Ciosek et al., 2019; Chen et al., 2017). Offline, this is used to stay closer to actions in the data by, conversely,

being pessimistic in the face of uncertainty (Buckman et al., 2020). Specifically, action-value estimates are penalised based on their level of uncertainty, in effect guiding the agent towards actions that are high-value/low-variance.

Although there are several techniques available for uncertainty quantification, ensemble-based methods in particular have found favour in offline-RL. SAC-N (An et al., 2021), for example, utilises an ensemble of value functions to approximate a value distribution, using the minimum value across the ensemble to penalise estimates pessimistically, attaining strong performance on offline benchmarks. However, the ensemble size needed to realise this minimum can be excessively large, resulting in substantial computational overhead and scalability issues. While alternative approaches attempt to alleviate this by promoting greater diversification across the ensemble (An et al., 2021) or incorporating elements of conservative value estimation (Ghasemipour et al., 2022), they still remain relatively computationally demanding.

Recognising the potential of ensemble-based approaches to offline-RL, in this work we aim to address this practical obstacle through the use of policy constraints. In offline-RL, policy constraints have been extensively employed as a method for ensuring OOD policy actions stay closer to data actions. Here, we investigate its role as a simple method for controlling the effective sample size of OOD actions, thus directly regulating the degree of epistemic uncertainty of value functions assessed for these actions.

Our findings indicate that when using unconstrained policies, the level of uncertainty in value estimates for OOD actions is relatively low, necessitating the use of large ensemble sizes to accurately estimate the tails of value distributions, and thus achieve the minimal values required for sufficient penalisation. Using a constrained policy on the other hand, results in increased epistemic uncertainty, proportional to the strength of constraint and distance from data actions. Due to the heightened uncertainty, the tails of the value distribution become elongated, allowing for the acquisition of similar minimal values with a considerably reduced ensemble size. We find this to be the case when using two alternative methods for training the ensemble of value functions, namely shared and independent target values.

We leverage these findings as part of two distinct implementations based on existing offline-RL algorithms: TD3-BC-N (an extension of the TD3-BC (Fujimoto & Gu, 2021)) and SAC-BC-N (an extension of SAC-N). In both cases, the policy constraint takes the form of behavioural cloning (BC), avoiding the need to explicitly model the behaviour of data actions, with inherent benefits in terms of simplicity and efficiency. Moreover, we use BC to extend these approaches to online fine-tuning, gradually diminishing its influence as the agent interacts with the environment.

Through an extensive empirical evaluation using the D4RL benchmarking suite (Fu et al., 2020), we show both implementations are able to produce state-of-the-art policies in a computationally efficient manner, which can then be fine-tuned during deployment while largely mitigating severe performance drops during the offline-to-online transition. In addition, we find this can be achieved without having to adjust hyperparameters based on data quality, an arguably necessary feature for real-world application where the performance properties of the data may be undetermined. We hope our work highlights the potential of such an approach and provides a useful benchmark for future

advancements to be evaluated against. For the purpose of transparency and reproducibility, the code base for this work is made freely available.[1]

The remainder of this manuscript is structured as follows. In Sect. 2 we outline related work on behavioural cloning, uncertainty quantification and online fine-tuning before providing background material in Sect. 3. We present our offline learning and online fine-tuning procedures in Sect. 4 and evaluate them in Sect. 5. We end with a discussion and concluding comments in Sect. 6.

2 Related work

In this Section, we provide an overview of related literature on offline-RL and online fine-tuning. With respect to offline-RL, we focus on methods that utilise behavioural cloning and uncertainty estimation as strategies to counteract overestimation bias for out-of-distribution actions. For online fine-tuning, we review methodologies that prioritize both stability and performance.

2.1 Methods based on behavioural cloning

In its most vanilla form, behavioural cloning (BC) is a form of imitation learning designed to mimic the actions of a demonstrator, most commonly an expert (Bain & Sammut, 1995). Its use in offline-RL is primarily to act as a policy constraint, preventing agents from choosing actions that stray too far from the source data.

One way of incorporating BC into offline-RL is through modelling the distribution of actions in the data, commonly referred to as the behaviour policy. In BCQ (Fujimoto et al., 2019), this is achieved using a Variational AutoEncoder (VAE) (Sohn et al., 2015), whose generated actions form the basis of a policy which is then optimally perturbed by a separate network in the DDPG (Lillicrap et al., 2015) framework. This approach is modified by PLAS (Zhou et al., 2020) to train policies within the latent space of VAE, naturally constraining policies as they are decoded from latent to action space. VAEs are also utilised by BRAC (Wu et al., 2019) and BEAR (Kumar et al., 2019), which instead seek to minimise divergence metrics (Kullback–Leibler, Wasserstein, Maximum Mean Discrepancy) between the behaviour and the learned policy. To account for multimodality, Fisher-BRC (Kostrikov et al., 2021) clones a behaviour policy using Gaussian mixtures and uses this for critic regularisation via the Fisher divergence metric. Implicit Q-learning (IQL) (Kostrikov et al., 2021) combines expectile regression and advantaged weighted BC to train agents without having to evaluate actions outside the data. TD3-BC (Fujimoto & Gu, 2021) favours a minimalist approach, directly incorporating BC into policy updates via a mean squared error between data and policy actions.

Despite their diversity, each of these methodologies effectively addresses overestimation bias, facilitating the learning of a policy that either matches or surpasses the original behaviour. Additionally, they achieve this in a computationally efficient manner, requiring only a limited number of networks and relatively few gradient updates. However, these approaches tend to be overly restrictive, hindering agents' abilities to discern optimal behaviour from suboptimal data. Consequently, their performance is often inferior to

[1] https://github.com/AlexBeesonWarwick/OfflineRLConstrainedEnsemble.

alternative methods (An et al., 2021; Ghasemipour et al., 2022). Nonetheless, as we suggest, these techniques can still be employed in a complementary capacity alongside ensemble-based approaches, improving computational efficiency via fostering uncertainty for OOD value estimates.

2.2 Methods based on uncertainty quantification

As is customary in machine learning, we distinguish between two distinct sources of uncertainty: *aleatoric* and *epistemic* (Hullermeier & Waegeman, 2021). The former stems from inherent stochasticity while the later arises due to incomplete information. In deep learning, various techniques for quantifying both sources of uncertainty have been proposed [for extensive reviews see e.g. (Abdar et al., 2021; Zhou et al., 2022)] and several studies have endeavoured to provide insights in the context of RL [for instance (Eriksson et al., 2022; Charpentier et al., 2022; Lee et al., 2021)]. These preliminary attempts have sought to address various challenges, including mitigating Q-learning instability, achieving equilibrium between exploration and exploitation, and facilitating risk-sensitive sequential decision-making.

In model-free RL, ensemble methods have garnered considerable interest for estimating epistemic uncertainty for action-value estimates. In online-RL, ensembles are frequently employed to enhance exploration by encouraging agents to seek out actions whose estimated values vary the most. This is achieved by constructing a distribution of action-value estimates using the ensemble and acting optimistically with respect to the upper bound, as demonstrated by Chen et al. (2017). In offline-RL these distributions direct agents towards actions within the dataset by, conversely, acting pessimistically with respect to the lower bound, prioritizing actions characterized by high value and low variance.

SAC-N (An et al., 2021), for example, adapts SAC (Haarnoja et al., 2018a, b) to the offline setting by increasing the number of critics from 2 to N, choosing the minimum across the ensemble to penalise action-value estimates that vary the most. While very effective in term of performance, in some cases the size of the ensemble needed to estimate this minimum is excessively large (up to 500) as is the number of gradient steps required to reach peak performance (up to 3 M). Even with parallelisation, this results in considerable computational overhead, both in terms of training time and memory requirements, affecting the capacity to scale up to more complex, real-world problems.

EDAC (An et al., 2021) attempts to reduce ensemble size by increasing uncertainty through diversification. The authors note that, without intervention, the gradients of the critic ensemble tend to align, requiring larger and larger ensembles to achieve sufficient penalisation. To counteract this, EDAC diversifies these gradients by minimising the pairwise cosine similarity within the ensemble, reducing its size by as much as a factor of ten without compromising performance. However, this diversity regulariser can still be relatively expensive for medium-sized ensembles and the large number of gradient updates remain. Our proposed solution is instead based on increasing uncertainty through the use of policy constraints.

The approach most similar to our own is MSG (Ghasemipour et al., 2022), which also uses an ensemble of critics for uncertainty estimation, but uses conservative Q-learning (CQL) (Kumar et al., 2020) to steer agents towards actions in the data instead of BC. In effect, CQL "pushes down" on value estimates for out-of-distribution actions and "pushes up" for actions in the data. MSG replaces the shared target of SAC-N/EDAC with independent targets to enforce pessimism, and when combined with CQL performs well on

challenging benchmarks. However, this performance is still dependent on relatively large ensembles and many gradient steps, with attempts to mitigate this using more efficient means such as multi-head (Lee et al., 2015) and multi-input/multi-outputs (Havasi et al., 2020) leading to detrimental impacts on performance. In contrast, our proposed solution emphasises mitigation through the application of BC.

In order to specifically characterise the uncertainty in value estimates for OOD-actions, PBRL (Bai et al., 2022) makes use of bootstrapping, sampling actions from the learned policy and penalising value estimates based on their deviation from the mean. Critic updates with respect to these estimates augment those based on non-bootstrapped uncertainty for in distribution actions. This idea is extended by RORL (Yang et al., 2022), which separately characterises uncertainty for three sets of state-action pairs (those in the data, perturbed states with data actions and perturbed states with policy actions at those states) in order to smooth Q-value estimates in regions outside the data, with the goal of learning policies that are robust to adversarial attacks. While these approaches are able to capture uncertainty effectively using a much reduced ensemble size (equal to our own), the techniques used to achieve this, most notably bootstrapping, are far less computationally efficient than the BC approach we propose, and less straightforward to implement.

In an attempt to remove the requirement for ensembles entirely, SAC-RND (Nikulin et al., 2023) estimates uncertainty using random network distillation (RND). The authors demonstrate that with an appropriate choice of prior and predictor, RND is able to discriminate between in-distribution and out-of-distribution actions sufficiently well enough so that anti-exploration bonuses can be used to regulate Q-values estimates, and thus agents are able to learn competitive policies. However, despite the fact this approaches uses only $N = 2$ critic networks, it is still less computationally efficient than our proposed approached, primarily stemming from the training associated with the RND component and comparatively large number of gradient updates.

2.3 Methods for online fine-tuning

Depending on the quality of the dataset, offline trained agents may exhibit limited performance upon deployment, necessitating further online fine-tuning through interaction with the environment. It can be argued that the domains which necessitate offline learning to begin with also necessitate a smooth transition from offline to online learning, that improvements in performance should not be preceded by periods of policy degradation. In practice, this presents a formidable challenge due to the sudden distribution shift from offline to online data, which can introduce bootstrapping errors that distort the pre-trained policy (Lee et al., 2020). While continued regularisation can potentially mitigate this issue, it can also hinder the agent's ability to learn from newly acquired samples. As such, approaches that promote stability as well as performance are desirable.

An initial theoretical study of policy fine-tuning in episodic Markov Decision Processes in Xie et al. (2021), examines the potential benefits of granting online agents access to a reference policy that is, in a certain sense, already close to an optimal one. The policy expansion scheme proposed in Zhang et al. (2023) attempts to achieve stable learning by using offline-trained policies as potential candidates within a policy set, while REDQ + AdaptiveBC (Zhao et al., 2021) seeks stability through adaptively adjusting the BC component of TD3-BC based on online returns. We make use of a similar approach proposed by Beeson & Montana (2022), which adjust the influence of BC based on exponential decay, avoiding the need for prior domain knowledge as required by REDQ + AdaptiveBC.

Other related studies have investigated different setups or aspects, such as action-free offline datasets (i.e., datasets without logged actions) (Zhu et al., 2023) or "learning on the job" (Nair et al., 2022) to improve policy generalisation. The feasibility of employing existing off-policy methods to capitalize on offline data through minimal algorithmic adjustments has be examined in Ball et al. (2023). Their findings underscore the significance of sampling mechanisms for offline data, the crucial role of normalizing the critic update, and the advantages of large ensembles for improving sample efficiency.

3 Preliminaries

In this section, we present the common RL setup and outline the challenges encountered when adapting algorithms to the offline setting. We then provide details of ensemble-based uncertainty methods we adopt as part of our approach.

3.1 Offline reinforcement learning

We follow standard convention and define a Markov decision process (MDP) with state space S, action space A, transition dynamics $T(s' \mid s, a)$, reward function $R(s, a)$ and discount factor $0 < \gamma \leq 1$ (Sutton & Barto, 2018). An agent interacts with this MDP by following a policy $\pi(a \mid s)$, which can be deterministic or stochastic. The goal of reinforcement learning is to discover an optimal policy $\pi^*(a \mid s)$ that maximises the expected discounted sum of rewards,

$$\mathbb{E}_{\pi} \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t),$$

also know as the return. In actor-critic methods, this is achieved by alternating between policy evaluation and policy improvement using Q-functions $Q^{\pi}(s, a)$, which estimate the value of taking action a in state s following policy π thereafter. Policy evaluation consists of updating the Q-function (the critic) based on the Bellman expectation equation

$$Q^{\pi}(s, a) = r(s, a) + \gamma \, \mathbb{E}_{s' \sim T, a' \sim \pi}(Q^{\pi}(s', a')),$$

where s' and a' are used to denote the next state and next action, respectively. Policy improvement comes in the form of updating the policy (the actor) so as to maximise $Q(s, a)$.

In terms of objective functions, policy evaluation and policy improvement are defined as, respectively,

$$Q^{\pi} = \arg\min_{Q} \; \mathbb{E}_{(s,a,s') \sim D} \Big(Q(s, a) - r(s, a) - \gamma Q^{\pi}(s', \pi(s')) \Big)^2, \tag{1}$$

and

$$\pi = \arg\max_{\pi} \; \mathbb{E}_{s \sim D} \Big[Q(s, \pi(s)) \Big], \tag{2}$$

where $r(s, a) + \gamma Q^{\pi}(s', \pi(s'))$ is commonly referred to as the target value.

In practice, both actor and critic are parameterised functions, employing non-linear approximation methods such as neural networks. Parameters are updated according to sample based estimates, with the samples themselves coming from the agent's own interactions with the environment. To improve data efficiency, these interactions are stored in a replay buffer D which is constantly added to and sampled from during training. To encourage sufficient exploration of the environment, a level of randomness is induced into online action selection, such as by adding noise if policies are deterministic or sampling if policies are stochastic.

In offline reinforcement learning, also known as batch reinforcement learning (Lange et al., 2012), the agent no longer has access to the environment and instead must learn solely from pre-existing interactions $D = (s_i, a_i, r_i, s'_i)$. While it is possible to adapt existing algorithms to this setting by simply removing online interaction, in practice this often leads to highly sub-optimal policies or a complete collapse of the learning process. The primary cause of this is the propagation and compounding of overestimation bias for state-action pairs absent in D (Levine et al., 2020). Such overestimation bias results from the bootstrapped nature of Q-network updates and the maximisation carried out as part of policy improvement.

This can be seen more clearly by examining the general objectives of policy evaluation and improvement. In policy evaluation (1), Q-value estimates for $Q(s, a)$ and $Q(s', a')$ use actions sampled from different policies, namely the behaviour policy $\pi_\beta(s)$ (i.e. the policy/policies that collected previous interactions), and the learned policy $\pi(s)$. Errors that appear during policy evaluation propagate to policy improvement (2), biasing actions that maximise spurious Q-values estimates. This then feeds back into policy evaluation, compounding existing errors which then propagate to policy improvement, and so on. In the online setting such bias can be mitigated by trialing policy actions in the environment, observing rewards and correcting Q-value estimates accordingly. In the offline setting this is no longer permitted and hence additional measures must be implemented in order to stabilise training.

3.2 Regularisation through uncertainty estimation

A sensible approach to combating overestimation bias is to target its root cause, namely the Q-values estimates themselves. One tool for achieving this is uncertainty estimation, using the premise that Q-value estimates for out-of-distribution (OOD) actions are inherently more uncertain than for actions in the data. This uncertainty can be used in training to favour Q-values with low-variance in policy evaluation and high-value/low-variance in policy improvement, in effect guiding the agent towards actions in the vicinity of the data.

This idea forms the basis of approaches such as SAC-N and EDAC. Both use an ensemble of N Q-functions to approximate Q-value distributions, updating network parameters using the minimum across the ensemble for policy actions $\pi(s)$. In terms of the general objectives for policy evaluation and improvement, these become, respectively:

$$Q_i^\pi = \arg\min_Q \; \mathbb{E}_{(s,a,s')\sim D}\left(Q_i(s,a) - r(s,a) - \gamma \min_{i=1,\dots,N} Q_i^\pi(s', \pi(s')) \right)^2, \tag{3}$$

and

$$\pi = \arg\max_{\pi} \mathbb{E}_{s\sim D}\left[\min_{i=1,\ldots,N} Q_i(s,\pi(s))\right].$$

Alternatively, as is done in MSG, each Q-function can be updated towards its own (rather than a shared) target value, giving a modified policy evaluation objective of:

$$Q_i^{\pi} = \arg\min_{Q} \mathbb{E}_{(s,a,s')\sim D}\left(Q_i(s,a) - r(s,a) - \gamma Q_i^{\pi}(s',\pi(s'))\right)^2. \tag{4}$$

Using uncertainty estimation in this way constitutes a pessimistic approach to offline-RL. By using the minimum across the ensemble, Q-value estimates for OOD actions are penalised according to their level of uncertainty. By increasing the size of the ensemble, the minimum is realised more accurately, and hence with large enough N the level of penalisation is sufficient to prevent overestimation bias. In practice, such approaches attain strong performance, but the size of the ensemble required to accurately estimate this minimum is often very large, necessitating the use of considerable computational resource to implement.

4 Policy constrained critic ensembles

The key issue we seek to address in this work is the high computational cost of ensemble-based approaches to offline reinforcement learning, approaches that are otherwise very effective due to their strong performance and straightforward implementation. These costs primarily stem from the need to use large ensembles to obtain accurate estimates of lower bounds, which form the basis of penalties applied to Q-value estimates for OOD actions.

As demonstrated by An et al. (2021), the strength of these penalties depend on both the size of the ensemble and the magnitude of the standard deviation. Using the same example for illustrative purposes [itself based on Royston (1982)], if $Q(s, a)$ follows a Gaussian distribution with mean $\mu(s, a)$ and standard deviation $\sigma(s, a)$, the approximate expected minimum of a set of N realisations is given by:

$$\mathbb{E}\left[\min_{j=1,\ldots,N} Q_j(s,a)\right] \approx \mu(s,a) - \Phi^{-1}\left(\frac{N-\frac{\pi}{8}}{N-\frac{\pi}{4}+1}\right)\sigma(s,a), \tag{5}$$

where Φ is the cumulative distribution function of the standard Gaussian.

In general the distribution of $Q(s, a)$ is unknown, but the same basic principles apply. In SAC-N, the size of the ensemble needed to sufficiently penalise Q-value estimates is high, as the standard deviation across the ensemble (i.e. level of uncertainty) is relatively small. In order to achieve similar levels of penalisation with a reduced ensemble size, the level of uncertainty across the ensemble must be increased. In EDAC this is achieved by diversifying the ensemble and in MSG by using conservative Q-learning.

Our proposed method for increasing this uncertainty is based on policy constraints. We note that, although policy constraints are primarily used to steer agents towards actions in the data, this also has an effect on the level of uncertainty of Q-values estimates of OOD actions. By constraining the policy, the Q-ensemble is trained on actions closer to the data, in effect reducing the effective sample size of OOD actions, which in turn increases epistemic uncertainty with respect to their Q-value estimates. The higher the level of

constraint, the greater the level of uncertainty as the tails of the value distribution expand. Thus, policy constraints provide an additional mechanism for controlling uncertainty in Q-value estimates, which can be used to achieve sufficient levels of penalisation with a much reduced ensemble size.

With this in mind, we modify existing ensemble-based approaches to directly incorporate behavioural cloning into policy updates, in a similar vein to TD3-BC (Fujimoto & Gu, 2021). While many other approaches for constraining policies exist (see Sect. 2), we favour this one in particular as it requires no explicit modelling of the behaviour policy π_β and is straightforward to implement, computationally cheap, flexible enough to accommodate deterministic and stochastic policies and requires no changes to policy evaluation using either shared (3) or independent (4) targets.

Let $\rho(a)$ be a function representing a divergence metric between policy and data actions a. The general policy improvement objective becomes:

$$\pi = \arg\max_{\pi} \quad \mathbb{E}_{(s,a)\sim D}\left[\min_{i=1,\ldots,N} Q_i(s, \pi(s)) - \beta\rho(a)\right]. \tag{6}$$

The hyperparameter β controls the balance between RL and BC, and by extension the level of uncertainty in Q-value estimate for OOD actions. Lower values favour RL but also lead to lower levels of uncertainty. Higher values increase uncertainty, but tip the balance towards BC, making it more difficult for the agent to discover high-value actions that lie beyond the data. Thus, the aim is to find a value of β that induces enough uncertainty without being too restrictive, allowing sufficient penalisation of Q-value estimates using a smaller ensemble.

Regardless of the form of $\rho(a)$, the balance in (6) is highly sensitive to Q-value estimates, which scale with rewards and vary across tasks. Therefore, to keep this balance in check, following the example of TD3-BC we normalise estimates by dividing by the mean of the absolute values, such that:

$$Q_{norm}(s, \pi(s)) = \frac{Q(s, \pi(s))}{\mathbb{E}_{s\sim D} \mid Q(s, \pi(s)) \mid}.$$

So far we have presented our approach within the general actor-critic framework, outlining the changes to policy evaluation and policy improvement from incorporating ensemble methods and behavioural cloning. In Sects. 4.1 and 4.2 we present two specific versions based on TD3 (Fujimoto et al., 2018) and SAC (Haarnoja et al., 2018b), respectively, which are then evaluated in Sect. 5 alongside our fine-tuning approach detailed in Sect. 4.3.

4.1 TD3-BC-N

Twin Delayed Deep Deterministic Policy Gradient (TD3) is an approach to reinforcement learning that proposes a number of techniques for addressing function approximation error in actor-critic methods, most notably DDPG. Based on a deterministic policy, TD3 makes use of a dual critic network for policy evaluation and updates Q-functions and policies at a ratio of 2:1. As is common with Q-learning approaches, target networks are used to stabilise training during policy evaluation. Exploration comes in the form of noise sampled from a Gaussian distribution.

We modify the baseline TD3 algorithm by increasing the number of critics from 2 to N and adding a BC term to policy updates in the form of a mean squared error (similar to

TD3-BC). Corresponding parameter updates and notation are as follows. Let θ_i and θ_i' represent the parameters of the ith Q-network and target Q-network, respectively, and ϕ and ϕ' represent the parameters for a policy network and target policy network, respectively. Let β represent the BC coefficient, N the ensemble size, τ the target network update rate, ϵ policy noise and B a sample of transitions from dataset D.

Each Q-network update is performed through gradient descent. For shared target values, we use:

$$\nabla_{\theta_i} \frac{1}{|B|} \sum_{(s,a,r,s')\sim B} \left(Q_{\theta_i}(s,a) - r - \gamma \min_{i=1,\ldots,N} Q_{\theta_i'}(s',a') \right)^2, \tag{7}$$

and for individual target values:

$$\nabla_{\theta_i} \frac{1}{|B|} \sum_{(s,a,r,s')\sim B} \left(Q_{\theta_i}(s,a) - r - \gamma Q_{\theta_i'}(s',a') \right)^2. \tag{8}$$

In either case $a' = (\pi_{\phi'}(s') + \text{noise})$ with noise sampled from an $N(0,\epsilon)$ distribution. The policy network update is performed through gradient ascent using:

$$\nabla_{\phi} \frac{1}{|B|} \sum_{(s,a)\sim B} \min_{i=1,\ldots,N} Q_{\theta_i}\big(s, \pi_\phi(s)\big) - \beta\big(\pi_\phi(s) - a\big)^2. \tag{9}$$

Target networks are updated using Polyak averaging:

$$\begin{aligned} \theta_i' &\leftarrow \tau\theta_i + (1-\tau)\theta_i' \\ \phi' &\leftarrow \tau\phi + (1-\tau)\phi'. \end{aligned} \tag{10}$$

The final procedure is presented in Algorithm 2.

Algorithm 1 TD3-BC-N

Require: Behavioural cloning coefficient β, ensemble size N, discount factor γ, policy noise ϵ, target network update rate τ and data set D
Initialise critic parameters θ_i, policy parameters ϕ and corresponding target parameters θ_i', ϕ'.
for $j = 0$ to J **do**
  Sample minibatch of transitions (s,a,r,s') from D
  Update Q-function parameters θ_i using equation (7) or (8)
  Update policy parameters ϕ using equation (9)
  Update target network parameters θ_i' using equation (10)
end for

4.2 SAC-BC-N

Soft Actor-Critic (SAC) is a maximum entropy approach to reinforcement learning. Based on a stochastic policy, SAC augments the standard policy evaluation and improvement objectives of actor-critic methods with an entropy regulariser, in effect encouraging agents to maximise

returns while acting as randomly as possible. This helps boost exploration, which comes in the form of sampling actions from the policy. Like TD3, SAC uses a dual critic with target networks to promote stability but uses a critic to actor update ratio of 1:1.

We modify the baseline SAC algorithm by increasing the number of critics from 2 to N and by adding a BC term to policy updates. Since the policy is stochastic, this BC term can take the form of either a mean-squared error or log-likelihood. Corresponding parameter updates and notation are as follows. Let θ_i and $\theta_i^{'}$ represent the parameters of the ith Q-network and target Q-network, respectively, and ϕ represent the parameters for a policy network. Let α represent the entropy coefficient, $\mathcal{H}$ the minimum entropy, β the BC coefficient, N the ensemble size, τ the target network update rate and B a sample of transitions from dataset D.

Each Q-network update is performed through gradient descent. For shared target values we use:

$$\nabla_{\theta_i} \frac{1}{|B|} \sum_{\substack{(s,a,r,s') \sim B \\ a' \sim \pi_\phi(s')}} \left(Q_{\theta_i}(s,a) - r - \gamma \min_{i=1,\ldots,N} Q_{\theta_i'}(s',a') + \gamma\alpha \log \pi_\phi(a' \mid s') \right)^2, \tag{11}$$

and for individual target values:

$$\nabla_{\theta_i} \frac{1}{|B|} \sum_{\substack{(s,a,r,s') \sim B \\ a' \sim \pi_\phi(s')}} \left(Q_{\theta_i}(s,a) - r - \gamma Q_{\theta_i'}(s',a') + \gamma\alpha \log \pi_\phi(a' \mid s') \right)^2. \tag{12}$$

The policy network update is performed through gradient ascent. For mean-squared error we use:

$$\nabla_{\phi} \frac{1}{|B|} \sum_{\substack{(s,a) \sim B \\ a_p \sim \pi_\phi(s)}} \min_{i=1,\ldots,N} Q_{\theta_i}\big(s, a_p\big) - \alpha \log \pi_\phi(a_p \mid s) - \beta \big(\pi_\phi(s) - a\big)^2. \tag{13}$$

and for log-likelihood:

$$\nabla_{\phi} \frac{1}{|B|} \sum_{\substack{(s,a) \sim B \\ a_p \sim \pi_\phi(s)}} \min_{i=1,\ldots,N} Q_{\theta_i}\big(s, a_p\big) - \alpha \log \pi_\phi(a_p \mid s) + \beta \log \pi_\phi(a \mid s). \tag{14}$$

The entropy coefficient update is performed through gradient ascent using:

$$\nabla_{\alpha} \frac{1}{|B|} \sum_{\substack{s \sim B \\ a_p \sim \pi_\phi(s)}} \alpha\big(\log \pi_\phi(a_p \mid s) + \mathcal{H}\big). \tag{15}$$

Target networks are updated using Polyak averaging:

$$\theta_i^{'} \leftarrow \tau\theta_i + (1-\tau)\theta_i^{'}. \tag{16}$$

The final procedure is presented in Algorithm 4.

Algorithm 2 SAC-BC-N

Require: Behavioural cloning coefficient β, ensemble size N, discount factor γ, minimum entropy $\mathcal{H}$, target network update rate τ and data set D
Initialise critic parameters θ_i and corresponding target parameters θ'_i.
Initialise policy parameters ϕ and entropy coefficient α
for $j = 0$ to J **do**
  Sample minibatch of transitions (s, a, r, s') from D
  Update Q-function parameters θ_i using equation (11) or (12)
  Update policy parameters ϕ using equation (13) or (14)
  Update entropy parameter α using equation (15)
  Update target network parameters θ'_i using equation (16)
end for

4.3 Stable online fine-tuning

The main goal in offline-RL is to discover optimal behavioural from existing data sets, allowing agents to learn effective policies before being deployment in the environment. Following deployment however, agents can collect more information about the environment, presenting opportunities for continued improvement via online fine-tuning. As agents can now correct for value estimates through online interaction, it may seem natural to remove constraints imposed during offline learning, but in practice this can often result in an initial phase of policy degradation due to the abrupt transition from constrained to unconstrained learning (see Sect. 2). In many situations, such degradation is deemed undesirable, emphasising the need for approaches that prioritize stability alongside performance.

During the transition from offline to online learning, an agent's policy should exhibit consistent improvement, surpassing its offline performance without experiencing periods of substantial deterioration. Our approach is well-suited to accomplishing these objectives. First, by making minimal modifications to existing algorithms, we largely preserve the core characteristics that contribute to their success online. Second, our utilisation of BC offers a convenient mechanism for stabilizing the transition by gradually reducing its influence over time. Numerous methods can achieve this, but for simplicitly we adopt an approach based on exponential decay as in (Beeson & Montana, 2022). Let β_{start} and β_{end} be the initial and final values of the BC component β, respectively, and S the number of decay steps. The exponential decay rate κ_β is given by:

$$\kappa_\beta = \exp\left[\frac{1}{S}\log\left(\frac{\beta_{end}}{\beta_{start}}\right)\right]. \tag{17}$$

Determining the appropriate use of existing data is also an important aspect of online fine-tuning. One option is to supplement the existing data with new transitions, enabling a seamless transition as the agent gradually acquires new information online. However, if the original data is sub-optimal, the online fine-tuning process may be slow, as the agent's offline-trained policy is not fully utilised. Alternatively, discarding the data allows the

agent to improve its policy without being hampered by data it has already improved upon. However, this could compromise stability in the initial stages due to limited experience and a paucity of data. We propose an approach that strikes a balance, adding new transitions to a portion of the original data before training. We outline this fine-tuning procedure using TD3-BC-N in Algorithm 3. The corresponding procedure for SAC-BC-N is provided in the Appendix.

Algorithm 3 Online fine-tuning (TD3-BC-N)

Require: Ensemble size N, discount factor γ, policy variance ϵ, target network update rate τ, data set D, exploration noise σ and decay parameters $\beta_{start}, \beta_{end}, S$
Initialise pre-trained critic parameters θ_i, policy parameters ϕ and corresponding target parameters θ'_i, ϕ'.
Initialise environment and replay buffer R
Populate R with a proportion of transitions from D.
for $k = 0$ to K **do**
  Act in environment with exploration, $a \sim \pi_\phi(s) + N(0, \sigma)$
  Store resulting transition (s, a, r, s') in R
end for
Set decay rate κ_β as per equation (17)
Set $\beta = \beta_{start}$
for $j = 0$ to J **do**
  Act in environment with exploration, $a \sim \pi_\phi(s) + N(0, \sigma)$
  Store resulting transition (s, a, r, s') in R
  Sample minibatch of transitions (s, a, r, s') from R
  Update Q-function parameters θ_i using equation (7) or (8)
  Update policy parameters ϕ using equation (9)
  Update target network parameters θ'_i using equation (10)
  Update BC coefficient $\beta = \max(\beta_{end}, \kappa_\beta \beta)$
end for

5 Experimental results

In this section, we present a comprehensive evaluation of our offline learning and online fine-tuning procedures using the open-source D4RL benchmarking suite. Section 5.1 provides an overview of this benchmark and the domains we consider, with Sect. 5.2 outlining implementation details. In Sect. 5.3 we investigate our claims regarding the impact of policy constraints on uncertainty estimation, and examine the trade-off between ensemble size and level of constraint. This is followed by a comparison of performance and computational efficiency in Sect. 5.4, as well as a number of supplementary experiments to highlight the importance of individual components and implementation choices. We end in Sect. 5.5 with an assessment of our fine-tuning strategy.

5.1 Benchmark datasets

D4RL is a popular resource for benchmarking offline reinforcement learning algorithms. The suite contains a wide range of tasks and data sets designed to test an agent's ability to learn effective policies in various settings. We outline the domains considered in this work, and refer the reader to the original paper for further details (Fu et al., 2020).

- *MuJoCo*. This setting makes use of the hopper, halfcheetah and walker2d environments of the MuJoCo physics simulator (Todorov et al., 2012), assessing how well agents learn from sub-optimal and/or narrow data distributions. Each environment has four associated data sets: "expert" which contains transitions collected from an agent trained to expert level using SAC; "medium" which contains transitions collected from an agent trained to 1/3 expert level using SAC; "medium-replay" which contains the transitions used to train the medium-level agent; "medium-expert" which contains the combined transitions from "medium" and "expert". In general this setting is considered one of the easier among the benchmark, with environments having well defined rewards structures and data sets comprising a decent proportion of near-optimal trajectories.
- *Maze2D*. This settings involves moving a force actuated ball to a fixed target location. Data is collected via a controller which starts and ends at random goal locations. The purpose of this setting is to test an agent's ability to stitch together previous trajectories to reach the evaluation goal. There are three increasingly difficult mazes: "umaze", "medium" and "large". We focus on the more challenging sparse reward setting, in which the agent receives a reward of 1 when within a 0.5 unit radius of the target goal and 0 otherwise.
- *AntMaze*. This setting replaces the ball from Maze2d with an more complex Ant robot, with episodes terminating once the Ant reaches the goal location. Data is collected via a controller using two different methods: "play" in which the controller moves from hand-picked starting locations to hand-picked goals; "diverse' in which the controller moves from random starting locations to random goals. This setting is considered one of the more challenging as agents must learn to both control the Ant and stitch trajectories together using only sparse rewards.
- *Adroit*. This setting makes use of the Adroit environment, controlling a high-dimensional robotic hand to perform specifics tasks. The aim is to assess whether agents can learn from narrow data distributions ("cloned") and human demonstrations ("human") with sparse rewards. We focus on the "pen" task as, similar to other approaches, this is the only task in which notable performance is achieved (see Appendix).

5.2 Implementation details

Following the protocol of D4RL, we train agents using offline data sets and evaluate their performance in the simulated environment. Performance is measured in terms of normalised score, with 0 and 100 representing random and expert policies, respectively. Each experiment is repeated across five random seeds with reported results the mean normalised score $\pm$ one standard error across 50 evaluations for MuJoCo and 500 evaluations for Maze2d/AntMaze/Adroit (10 and 100 evaluations per seed, respectively).

For both TD3-BC-N and SAC-BC-N each Q-network comprises a 3-layer MLP with ReLU activation functions and 256 nodes, taking as input a state-action pair and outputting

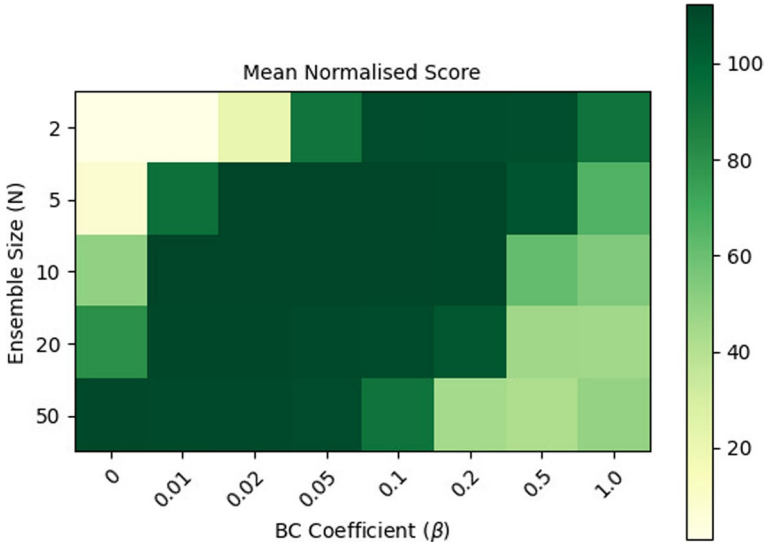

Fig. 1 Performance as a function of N and β. Lower values of β require larger values of N and smaller values of N require higher values of β. If both N and β are large, the uncertainty in Q-value estimates for OOD actions is too high, and thus the penalty applied too severe, leading the agent to prefer actions similar to those of the data

a Q-value. For TD3-BC-N the policy network comprises a 3-layer MLP with ReLU activation functions and 256 nodes, taking as input a state and outputting an action bound to $[-1, 1]$ via tanh transformation. For SAC-BC-N the policy network comprises the same architecture but instead outputs the mean and standard deviation of a Gaussian distribution which is also bound to $[-1, 1]$ via tanh transformation. Each approach retains the hyperparameters values of their online counterpart (full details are provided in the Appendix).

Across all data sets, we train agents for 1 M gradient steps using an ensemble size of $N = 10$. To help stabilise training for narrow data distributions, we inflate the value of the BC coefficient β by a factor of 10 for the first 50 k gradient steps (alternatively the policy can be updated using only BC). We use shared targets for MuJoCo and Maze2d tasks and independent targets for AntMaze and Adroit. We investigate the impact of each of these designs decisions as part of our ablations studies.

For the BC component, we find the characteristics of each environment necessitate varying intensities, and for SAC-BC-N dictate its form (mean-squared error or log-likelihood). We therefore adjust its intensity and/or form based on task type, but to better reflect real-world scenarios where the quality of the data is often unknown, we prohibit adjustments within the same task. Values for each task and data set are provided in the Appendix.

5.3 The impact of policy constraints on uncertainty

Before we consider the full range of tasks and data sets, we first investigate the claims made in previous sections relating to the impact of policy constraints on uncertainty levels in Q-value estimates for OOD actions. To do this, we train a number of agents using

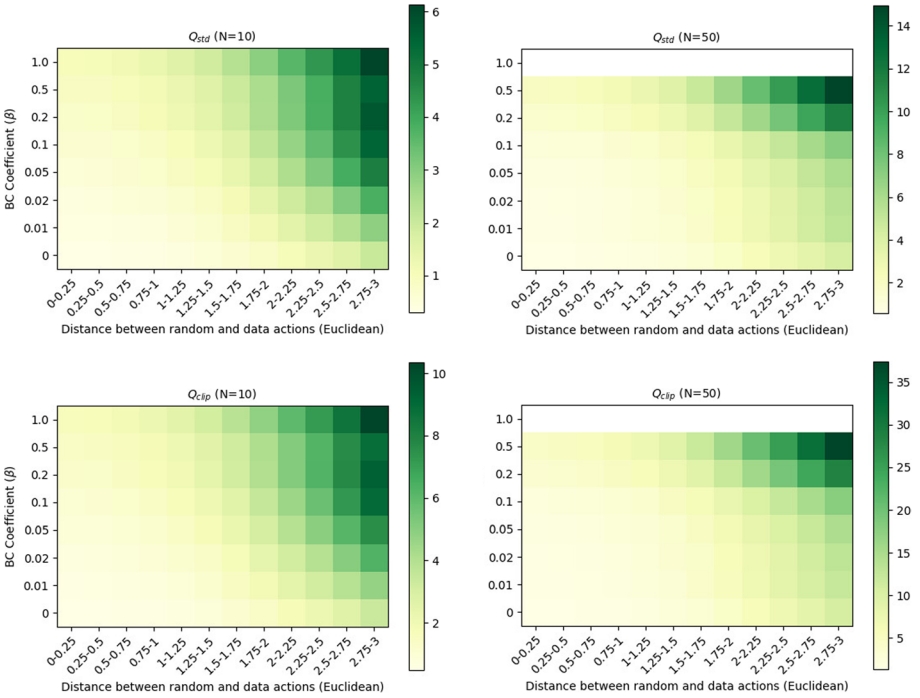

Fig. 2 Uncertainty as a function of distance, N and β. Top row standard deviation, bottom row clip penalty. As the distance between random and data actions increases so too does the level of uncertainty, becoming more pronounced as β and N get larger. White space is used to represent erroneous values due to unreliable Q-values estimates resulting from divergent critic loss during training

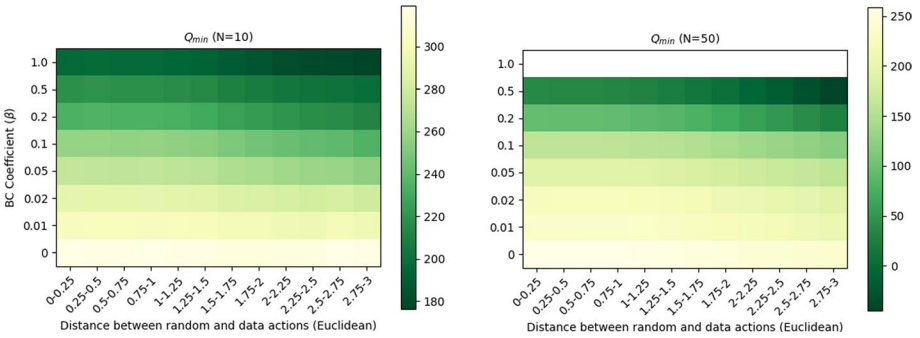

Fig. 3 Q_{min} as a function of distance, N and β. As the distance between random and data actions increases, Q_{min} decreases, with this decrease more pronounced as β and N get larger. White space is used to represent erroneous values due to unreliable Q-values estimates resulting from divergent critic loss during training

TD3-BC-N with dependent target values across a range of N and β on the "hopper-medium-expert" dataset, and examine the performance of resulting policies and uncertainty of Q-values estimates from resulting ensembles.

Beginning with performance, we summarise this via a heatmap in Fig. 1, using shade to represent mean normalised score. For the lowest values of β we see that larger ensembles

are required to prevent overestimation bias through sufficient penalisation of OOD actions. As the value of β increases, the size of the ensemble required to achieve this level of penalty decreases, allowing the same level of performance to be attained as for larger ensembles. We also see that the larger the value of N, the smaller the value of β before performance starts to degrade. In these cases, the level of uncertainty resulting from both a large ensemble and high level of policy constraint leads to over-penalisation of Q-values estimates, in effect driving the agent towards actions in the data at an increased rate.

In terms of uncertainty of Q-value estimates, we consider both the standard deviation across the ensemble and the clip penalty $Q_{clip}(s, a)$, which measures the size of the difference between the mean and minimum:

$$Q_{clip}(s,a) = \frac{1}{N}\sum_{j=1}^{N} Q(s,a) - \min_{j=1,\ldots,N} Q(s,a).$$

In particular, we examine how each of these measures of uncertainty varies according to how far actions are from the data and the values of N and β.

To this effect, we sample 50,000 states from the data and 50,000 actions from a random policy and calculate (a) the Euclidean distance between random and data actions and (b) the standard deviation/clip penalty. We then group distances into equally sized bins and within each bin calculate the average standard deviation/clip penalty. We summarise results for $N = 10$ and $N = 50$ in Fig. 2 via heatmaps, using shade to represent the size of the corresponding uncertainty metric. Similar plots for $N = [2, 5, 20]$ can be found in the Appendix. In general, we see that as the distance between random and data actions increases, so too does the level of uncertainty (standard deviation and penalty gap), and this becomes more pronounced as the value of β increases. This supports our hypothesis that policy constraints can be used to control uncertainty in Q-value estimates. We also see that the highest levels of uncertainty occur when both N and β are large, supporting our explanation of declining performance as observed in Fig. 1.

Finally, we also examine the distribution of the minimum across the ensemble, Q_{min}, as this value is the one used in updates during policy evaluation and policy improvement. Using the same format as for uncertainty, we summarise results for $N = 10$ and $N = 50$ in Fig. 3, using shade to represent the value of Q_{min}. In general, we see that Q_{min} decreases as the distance between random and data actions increases, being more pronounced as either N or β increase. This culminates in the lowest Q_{min} values when the size of the ensemble and level of constraint are at their highest, mirroring the findings based on uncertainty measures.

For completeness, we reproduce these plots for agents trained using independent target values in the Appendix, finding in general the same features. We also provide additional plots examining (a) the distribution of Q_{min} for policy actions and (b) the shape of the distribution of Q-value estimates for individual actions, providing more insights into the impact of β on uncertainty.

5.4 Performance and efficiency comparisons

As one of our objectives is to attain the same-level of performance as ensemble-based methods, we compare to published results from SAC-N, EDAC and MSG. As the leading BC based approaches we also compare to published results from IQL and TD3-BC. Finally, since MSG makes use of CQL we also compare to updated CQL results as published in the

Table 1 Performance comparison across D4RL benchmark

Task/data set	CQL	IQL	TD3-BC	EDAC	SAC-N	MSG	TD3-BC-N	SAC-BC-N
halfcheetah-v2								
-medium	44.0	47.4	48.3	65.9	67.5	–	63.3 ± 0.1	65.6 ± 0.2
-medium-replay	45.5	44.2	44.6	61.3	63.9	–	55.3 ± 0.1	61.5 ± 0.1
-medium-expert	91.6	86.7	90.7	106.3	107.1	–	101.7 ± 0.3	102.6 ± 0.5
-expert	–	–	96.7	106.8	105.2	–	103.8 ± 0.5	105.3 ± 0.1
hopper-v2								
-medium	58.5	66.3	59.3	101.6	100.3	–	101.5 ± 0.3	101.2 ± 0.1
-medium-replay	95.0	94.7	60.9	101.0	101.8	–	99.1 ± 0.1	100.8 ± 0.2
-medium-expert	105.4	91.5	98.0	110.7	110.1	–	112.3 ± 0.0	111.3 ± 0.1
-expert	–	–	107.8	110.1	110.3	–	112.7 ± 0.1	111.5 ± 0.1
walker2d-v2								
-medium	72.5	78.3	83.7	92.5	87.9	–	90.9 ± 0.2	85.3 ± 0.1
-medium-replay	77.2	73.9	81.8	87.1	78.7	–	91.4 ± 0.4	90.8 ± 0.2
-medium-expert	108.8	109.6	110.1	114.7	116.7	–	113.5 ± 0.1	110.9 ± 0.0
-expert	–	–	110.2	115.1	107.4	–	113.2 ± 0.0	110.4 ± 0.0
mujoco average	–	–	82.3	97.8	96.4	–	96.6	96.4
(exc. expert)	77.6	77.0	75.3	93.5	92.7	–	92.2	92.2
maze2d-v1								
-umaze	–	–	–	–	–	101.1	155.9 ± 1.4	155.8 ± 2.9
-medium	–	–	–	–	–	57.0	159.9 ± 2.1	163.1 ± 1.3
-large	–	–	–	–	–	159.3	219.4 ± 2.0	210.8 ± 2.0
maze2d average	–	–	–	–	–	105.8	178.4	176.6
antmaze-v0								
-umaze	74.0	87.5	78.6	–	–	97.8	98.3 ± 0.7	98.6 ± 0.5
-umaze-diverse	84.0	62.2	71.4	–	–	81.8	90.6 ± 1.3	91.2 ± 1.3
-medium-play	61.2	71.2	10.6	–	–	89.6	87.0 ± 1.5	85.8 ± 1.6
-mediumdiverse	53.7	70.0	3.0	–	–	88.6	86.2 ± 1.5	73.8 ± 2.0
-large-play	15.8	39.6	0.2	–	–	72.6	76.2 ± 1.9	65.8 ± 2.1
-large-diverse	14.9	47.5	0	–	–	71.4	74.2 ± 2.0	75.8 ± 1.9
antmaze average	50.6	63	27.3	–	–	83.6	85.5	81.8
adroit-v1								
-pen-cloned	39.2	37.3	–	68.2	64.1	–	67.2 ± 2.9	58.0 ± 2.8
-pen-human	37.5	71.5	–	52.1	9.5	–	72.8 ± 2.7	70.3 ± 2.9
adroit average	38.4	54.4	–	60.2	36.8	–	70.0	64.2

Figures are normalised scores, with 0 and 100 representing random and expert policies, respectively. For TD3-BC-N and SAC-BC-N we report the mean normalised score ± one standard error across 50 evaluations for MuJoCo tasks (10 evaluations over 5 seeds) and 500 evaluations for Maze2d, AntMaze and Adroit tasks (100 evaluations over 5 seeds). Both TD3-BC-N and SAC-BC-N are able to match the state-of-the-art performance across all domains. This is the case even with the restriction preventing BC adjustments within the same task

Fig. 4 Comparing the performance of TD3-BC-N (green) and SAC-BC-N (red) against the best method from Table 1 (blue). Performance is competitive across all tasks (Color figure online)

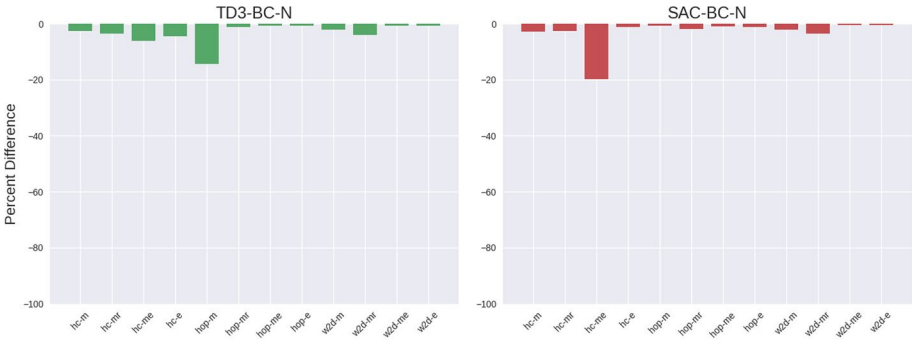

Fig. 5 Evaluating robustness of learn policies for MuJoCo tasks. Each plot shows the percentage difference between the mean and worst performing episode across 50 evaluations (10 evaluations per 5 seeds). With the exception of one data set, both TD3-BC-N and SAC-BC-N are able to produce robust policies regardless of data quality

IQL paper.[2] For completeness, we provide additional comparisons to approaches outlined in Sect. 2 in the Appendix.

We present results for all tasks and data sets in Table 1. Where figures are not published for a given task we denote the entry as "–".[3] To help better visualise performance levels, we compare our results to the best performing method in Fig. 4, which with a few exceptions is SAC-N/EDAC for MuJoCo and Adroit, and MSG for maze tasks. For the MuJoCo and Adroit environments, we see that in general both TD3-BC-N and SAC-BC-N can match the performance of SAC-N and EDAC, and for Maze2d and AntMaze they can match the performance of MSG. Note that this is achieved without adjusting hyperparameters within the same task, in contrast to SAC-N, EDAC and MSG. In the Appendix we investigate the effect of removing this restriction using the MuJoCo environments, finding performance can be slightly enhanced.

[2] These results are based on updated D4RL data sets following minor bug fixes.

[3] While MSG does consider the MuJoCo environments, results are only presented visually and are in general on-par or below those of SAC-N/EDAC.

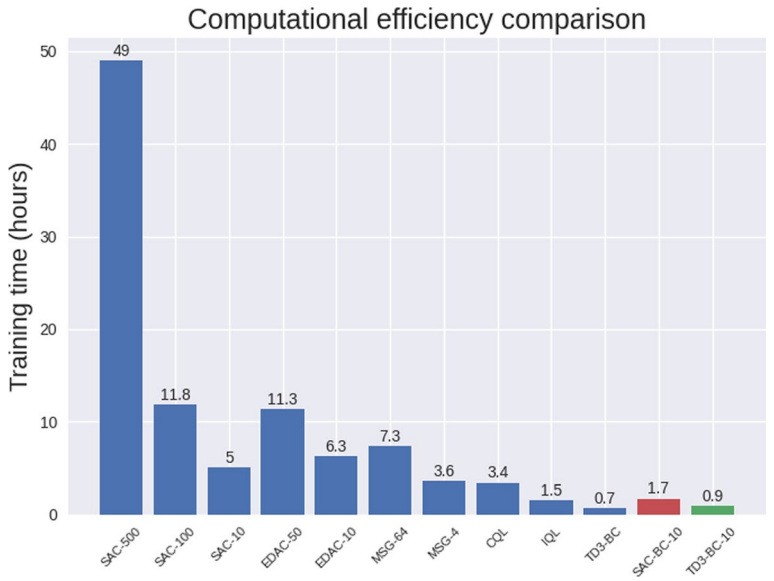

Fig. 6 Computational efficiency. A smaller ensemble size coupled with fewer gradient updates allows TD3-BC-N and SAC-BC-N to significantly reduce computation time to levels similar to that of more minimalist approaches such as TD3-BC

Fig. 7 Performance and efficiency. Average training time and normalised score across MuJoCo and AntMaze tasks. TD3-BC-N and SAC-BC-N can match the performance of ensemble-based approaches while retaining the computational efficiency of those based on behavioural cloning

For the MuJoCo domain in particular we note there is very little variation in performance across seeds/evaluations, demonstrating our approach is able to learn robust as well as performant policies. This is further evidenced in Fig. 5 where we plot the percentage difference between the mean and worst score across the 50 evaluations, which in most cases is negligible. Since real-world application will typically only involve single policy deployment, such a property is highly desirable.

After demonstrating our approach can match state-of-the-art alternatives in terms of performance, we turn our attention to computational efficiency. To ensure a fair comparison,

Fig. 8 Ablations studies. Each plot shows the percentage difference in mean normalised score between each ablation and the main results from Table 1. Ablations 1 and 2 show that both behavioural cloning and an ensemble of critics are necessary to achieve strong performance. Ablations 3 and 4 show the importance of our implementation choices, namely the use of an initial period of inflated BC and independent targets for AntMaze/Adroit environments

we implement our own versions of baselines based on author published source code and the CORL repository (Tarasov et al., 2022), and run them on the same hardware/software configuration. We use exactly the same network architecture across ensemble-based approaches, training each member of the ensemble in parallel. For CQL, IQL and TD3-BC we use the network architecture as described in their respective papers. Full details are provided in the Appendix.

In Fig. 6 we plot the training time in hours of each approach, considering several variations of SAC-N, EDAC and MSG based on ensemble size, which varies according to task type. We see that TD3-BC-N and SAC-BC-N are easily the most efficient among the ensemble-based approaches, a direct consequence of a smaller ensemble size and need for fewer gradient updates to reach peak performance. In particular, the computation time for TD3-BC-N is comparable to the minimalist approach of TD3-BC.

To get a clearer sense of how performance and efficiency compare across algorithms, in Fig. 7 we plot the average training time and normalised score for MuJoCo[4] and AntMaze tasks. We see that ensemble-based approaches (SAC-N, EDAC, MSG) are the most performant, but also the most computationally expensive. Conversely, BC based approaches

[4] We exclude the "expert" data sets since these are not reported for CQL or IQL.

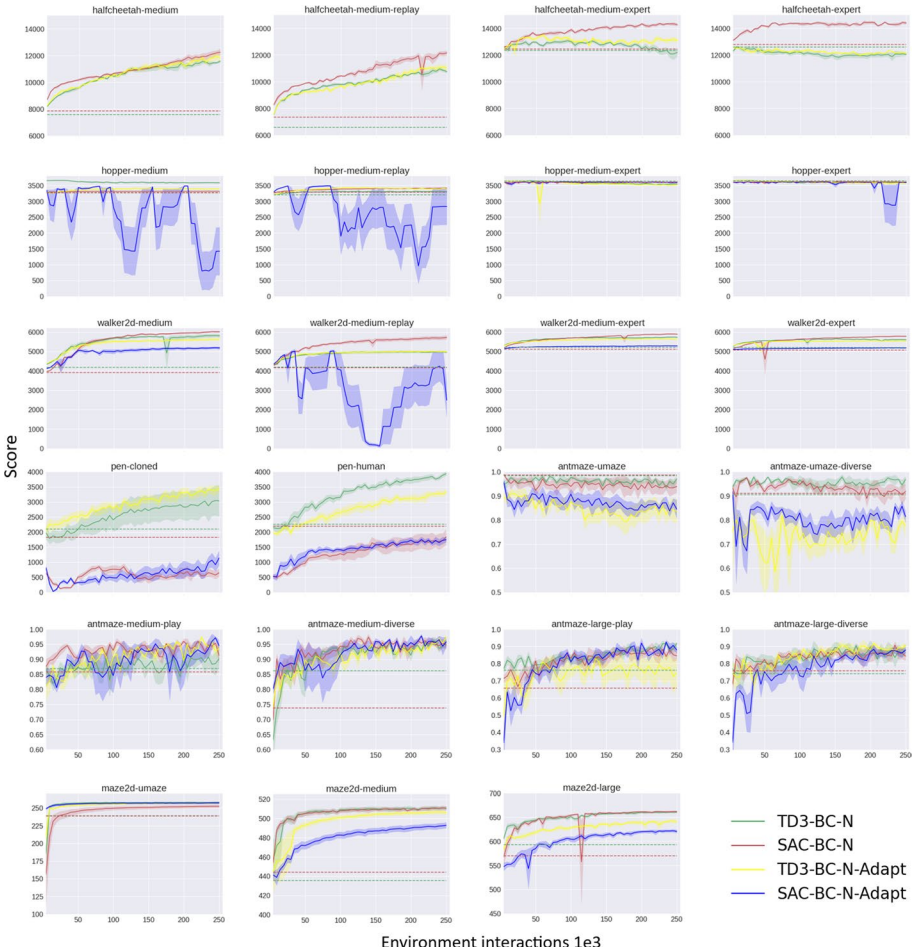

Fig. 9 Online fine-tuning for D4RL tasks. The solid line represents the mean non-normalised score across each of the five agents, shaded area the standard error and dashed line performance prior to fine-tuning. In general, agents are able to improve their policies in a stable manner, with only a few tasks/data sources causing stability issues. Note that learning curves for SAC-BC-N and -Adapt are identical for "halfcheetah" tasks as $\beta = 0$, hence we omit -Adapt versions for clarity

(TD3-BC, IQL) are the most computationally efficient, but least performant. TD3-BC-N and SAC-BC-N on other hand are able to retain the advantages of both approaches while diminishing their individual deficiencies.

5.4.1 Ablation studies

In addition to our main results, we also conduct a number of ablations studies to verify the importance of individual components of our approach, as well implementation decisions. In Ablations 1–3, we use the MuJoCo environments to assess the impact of removing the BC component, ensemble of critics and inflated period of BC, respectively. In Ablation 4, we use the AntMaze and Adroit environments to show the impact

of using dependent targets instead of independent targets during policy evaluation. We conduct these ablations using TD3-BC-N, making no other changes than those outlined above.

We summarise results in Fig. 8, plotting the percentage difference between each ablation score and the main results of Table 1. For Ablations 1–2 we see that removing either BC or the ensemble has a detrimental impact on performance overall. While the performance for some tasks is unaffected by removing the BC component, there are others that suffer catastrophic failure and hence its inclusion is essential. For Ablation 3 we see removing the inflated period of BC has minimal impact on most data sources, but the severe impact on "walker2d-expert" warrants its inclusion. Finally, in Ablation 4 we see the use of independent targets is crucial for the more challenging "medium" and "large" AntMaze environments and is beneficial for Adroit environments.

5.5 Online fine-tuning

Starting with our offline trained agents, we perform online fine-tuning according to the procedures outlined in Algorithms 3 and 4. We populate the replay buffer R with the last 2500 transitions from D and train agents for an additional 250 k environment interactions, with gradient updates commencing after the first 2500 interactions (i.e. $K = 2500$). The offline value of β is used for β_{start} and the number of decay steps S is set as 50 k. The value of β_{end} is set according to environment and procedure, but as with our offline experiments, its value doesn't change according to initial data quality. Values for each data set and procedure are provided in the Appendix. All other parameters remain the same.

For comparison purposes, we also fine-tune agents using a similar procedure to REQD+AdaptiveBC, in which the BC component is adjusted based on online returns. Specifically β is adjusted as:

$$\Delta\beta = K_P(R_{avg} - R_{target}) + K_D \max(0, R_{avg} - R_{current}),$$

where, as per the original paper, $K_P = 3e^{-5}$, $K_D = 1e^{-4}$, $R_{target} = 1.05$, $R_{current}$ is the latest normalised return and $R_{average}$ a running average of normalised returns. Note, in order to calculate normalised scores prior knowledge of random and expert performance is required. Apart from how β is adjusted during online training, all other conditions remain the same. We denote these curves as TD3-BC-N-Adapt and SAC-BC-N-Adapt.

For each task, we plot the corresponding learning curves in Fig. 9, evaluating policies every 5000 environment interactions (10 evaluations for MuJoCo, 100 evaluations for Adroit/AntMaze/Maze2d). The solid line represents the mean (non-normalised) score across each of the five seeds, shaded area the standard error and dashed line performance prior to fine-tuning. For the MuJoCo environments, in the majority of cases agents are able to improve their policies while avoiding severe performance drops during the offline to online transition. For TD3-BC-N, the performance for "hopper/halfcheetah-expert" declines slightly over the course of training and for SAC-BC-N there is sharp decline for "walker2d-expert" within the β decay period. For Adroit, TD3-BC-N manages a reasonable transition and subsequent improvement, but SAC-BC-N is less successful. With the exception of "antmaze-umaze", in AntMaze both TD3-BC-N and SAC-BC-N obtain improved policies in a reasonable stable manner. Finally, for Maze2d we see continued improvement for both methods, with some minor initial deterioration in TD3-BC-N for "maze2d-umaze" and fairly large initial slump in SAC-BC-N for "maze2d-umaze".

Comparing to the -Adapt versions, in general we see both performance and stability are as good or better, despite the fact our approach requires no prior domain knowledge. We note severe stability issues for SAC-BC-N-Adapt for "hopper-medium", "hopper-medium-replay" and "walker2d-medium-replay". This may be a result of the values of K_p and K_d, which were originally set based on mean-squared error, not transferring to log-likelihood.

6 Discussion and conclusion

In this work we have investigated the role of policy constraints as a mechanism for improving the computational efficiency of ensemble-based approaches to offline reinforcement learning. Through empirical evaluation, we have shown how constraints in the form of behavioural cloning can be used to control the level of uncertainty in the estimated value of out-of-distribution actions, allowing these estimates to be sufficiently penalised to prevent overestimation bias. Through this feature, we have been able to match state-of-the-art performance across a number of challenging benchmarks while significantly reducing computational burden, cutting the size of the ensemble to a fraction of that needed when policies are unconstrained. We have also shown how behavioural cloning can be repurposed to promote stable and performant online fine-tuning, by gradually reducing its influence during the offline-to-online transition. These achievements have required only minimal changes to existing approaches, allowing for easy implementation and interpretation.

Our work highlights a number of interesting avenues for future research. Primary among these is the development of methods for selecting the size of the ensemble N and level of behavioural cloning β offline. While we have demonstrated our approach can achieve strong performance using consistent hyperparameters, we have also shown how performance can be further improved by allowing them to vary. Related to this is the development of approaches for automatically tuning β during training, possibly making use of uncertainty metrics described in Sect. 5.3. A theoretical analysis of the impact of β on uncertainty could also prove beneficial in this regard.

While in this work we have used ensembles for uncertainty estimation, other techniques such a multi-head, multi-input/outputs and Monte Carlo dropout can just as easily be used and integrated with BC. Similarly, other forms of policy constraints and/or other divergence metrics can be incorporated into ensemble-based approaches in a relatively straightforward manner. As such, there a number of permutations which could lead to improved performance and/or computational efficiency.

Finally, our fine-tuning procedure may benefit from incorporating elements from methods outlined in Sect. 2, allowing for greater stability during the entire duration of online learning. In addition, our approach may also prove useful in promoting greater data efficiency in online-RL.

Table 2 TD3-BC-N shared hyperparameters and network architecture

Hyperparameter	Value
TD3-BC-N	
Optimiser	Adam
Actor learning rate	3e − 4
Critic learning rate	3e − 4
Batch size	256
Discount factor γ	0.99
Target network update rate τ	0.005
Policy noise ϵ	0.2
Policy noise clipping	$(^{-}0.5, 0.5)$
Critic-to-Actor update ratio	2:1
TD3-BC-N online	
Exploration noise σ	0.1
BC decay stay steps S	50,000
Architecture	
Critic hidden nodes	256
Critic hidden layers	3
Critic hidden activation	ReLU
Critic input	State + Action
Critic output	Q-value
Ensemble size N	10
Actor hidden nodes	256
Actor hidden layers	3
Actor hidden activation	ReLU
Actor input	State
Actor outputs	Action (tanh transformed)

Appendix

SAC-BC-N online fine-tuning procedure

Following on from Sect. 4.3, we outline the online fine-tuning procedure using SAC-BC-N in Algorithm 4.

Table 3 TD3-BC-N task specific BC hyperparameters

Task	Dataset	β	β_{end}
halfcheetah	medium	0.04	$1e^{-12}$
	medium-replay	0.04	$1e^{-12}$
	medium-expert	0.04	$1e^{-12}$
	expert	0.04	$1e^{-12}$
hopper	medium	0.03	0.02
	medium-replay	0.03	0.02
	medium-expert	0.03	0.02
	expert	0.03	0.02
walker2d	medium	0.03	$1e^{-10}$
	medium-replay	0.03	$1e^{-10}$
	medium-expert	0.03	$1e^{-10}$
	expert	0.03	$1e^{-10}$
maze2d-umaze		0.01	$1e^{-12}$
maze2d-medium		0.01	$1e^{-12}$
maze2d-large		0.01	$1e^{-12}$
antmaze-umaze	–	0.1	0.1
	-diverse	0.1	0.1
antmaze-medium	-play	0.02	0.01
	-diverse	0.02	0.01
antmaze-large	-play	0.02	0.005
	-diverse	0.02	0.005
pen	-cloned	10	2
	-human	10	2

Note the BC parameters are fixed within each task, i.e. do not vary based on dataset

Table 4 SAC-BC-N shared hyperparameters and network architecture

Hyperparameter	Value
SAC-BC-N	
Optimiser	Adam
Actor learning rate	3e − 4
Critic learning rate	3e − 4
Batch size	256
Discount factor γ	0.99
Target network update rate τ	0.005
Minimum entropy H	−1 * action dimension
SAC-BC-N online	
BC decay stay steps S	50,000
Architecture	
Critic hidden nodes	256
Critic hidden layers	3
Critic hidden activation	ReLU
Critic input	State + Action
Critic output	Q-value
Ensemble size N	10
Actor hidden nodes	256
Actor hidden layers	3
Actor hidden activation	ReLU
Actor input	State
Actor outputs	Mean/standard deviation of Gaussian

Table 5 SAC-BC-N task specific BC hyperparameters

Task	Dataset	BC form	β	β_{end}
halfcheetah	medium	Log-likelihood	0	0
	medium-replay	Log-likelihood	0	0
	medium-expert	Log-likelihood	0	0
	expert	Log-likelihood	0	0
hopper	medium	Log-likelihood	0.0025	0.001
	medium-replay	Log-likelihood	0.0025	0.001
	medium-expert	Log-likelihood	0.0025	0.001
	expert	Log-likelihood	0.0025	0.001
walker2d	medium	Log-likelihood	0.0025	$1e^{-10}$
	medium-replay	Log-likelihood	0.0025	$1e^{-10}$
	medium-expert	Log-likelihood	0.0025	$1e^{-10}$
	expert	Log-likelihood	0.0025	$1e^{-10}$
maze2d-umaze		MSE	0.005	$1e^{-12}$
maze2d-medium		MSE	0.02	$1e^{-12}$
maze2d-large		MSE	0.02	$1e^{-12}$
antmaze-umaze	–	MSE	0.1	0.05
	-diverse	MSE	0.1	0.05
antmaze-medium	-play	MSE	0.02	0.02
	-diverse	MSE	0.02	0.02
antmaze-large	-play	MSE	0.01	0.005
	-diverse	MSE	0.01	0.005
pen	-cloned	MSE	10	2
	-human	MSE	10	2

Note the BC parameters are fixed within each task, i.e. do not vary based on dataset

Table 6 Performance comparison across Adroit benchmark

Task/data set	CQL	IQL	EDAC	SAC-N	TD3-BC-N
pen-cloned	39.2	37.3	68.2	64.1	67.2
hammer-cloned	2.1	2.1	0.3	0.2	1.5
door-cloned	0.4	1.6	9.6	− 0.3	0.0
relocate-cloned	− 0.1	− 0.2	0	0	0.0
pen-human	37.5	71.5	52.1	9.5	72.8
hammer-human	4.4	1.4	0.8	0.3	0.8
door-human	9.9	4.3	10.7	− 0.3	0
relocate-human	0.2	0.1	0.1	− 0.1	− 0.1
average	11.7	14.8	17.7	9.2	17.8

Figures are normalised scores, with 0 and 100 representing random and expert policies, respectively. As with other methods, our approach only achieves notable performance in the "pen" task

Table 7 Performance comparison across MuJoCo benchmark, allowing β to vary within each task

Task/data set	EDAC	SAC-N	TD3-BC-N (fixed)	TD3-BC-N (variable)	TD3-BC-N β	SAC-BC-N (fixed)	SAC-BC-N (variable)	SAC-BC-N β
halfcheetah								
-medium	65.9	67.5	63.3 ± 0.1	66.9 ± 0.2	0	65.6 ± 0.2	65.6 ± 0.2	0
-medium-replay	61.3	63.9	55.3 ± 0.1	62.0 ± 0.2	0	61.5 ± 0.1	61.5 ± 0.1	0
-medium-expert	106.3	107.1	101.7 ± 0.3	101.7 ± 0.3	0.04	102.6 ± 0.5	102.6 ± 0.5	0
-expert	106.8	105.2	103.8 ± 0.5	103.8 ± 0.5	0.04	105.3 ± 0.1	105.3 ± 0.1	0
-random	28.4	28.0	22.4 ± 0.1	33.5 ± 0.3	0	27.9 ± 0.2	27.9 ± 0.2	0
hopper								
-medium	101.6	100.3	101.5 ± 0.3	103.2 ± 0.0	0.01	101.2 ± 0.1	101.2 ± 0.1	0.0025
-medium-replay	101.0	101.8	99.1 ± 0.1	100.0 ± 0.1	0.01	100.8 ± 0.2	103.5 ± 0.4	0
-medium-expert	110.7	110.1	112.3 ± 0.0	112.3 ± 0.0	0.03	111.3 ± 0.1	111.3 ± 0.1	0.0025
-expert	110.1	110.3	112.7 ± 0.1	112.7 ± 0.1	0.03	111.5 ± 0.1	111.5 ± 0.1	0.0025
-random	25.3	31.3	8.0 ± 0.1	12.9 ± 1.3	0	17.2 ± 1.6	22.3 ± 1.6	0
walker2d								
-medium	92.5	87.9	90.9 ± 0.2	96.6 ± 0.3	0.01	85.3 ± 0.1	92.1 ± 1.9	0.001
-medium-replay	87.1	78.7	91.4 ± 0.4	91.4 ± 0.4	0.03	90.8 ± 0.2	96.6 ± 0.3	0
-medium-expert	114.7	116.7	113.5 ± 0.1	115.7 ± 0.3	0	110.9 ± 0.0	117.5 ± 0.4	0.001
-expert	115.1	107.4	113.2 ± 0.0	113.2 ± 0.0	0.03	110.4 ± 0.0	110.4 ± 0.0	0.0025
-random	16.6	21.7	0.8 ± 0.3	14.8 ± 1.2	1	− 0.2 ± 0.0	7.0 ± 2.2	0.1
mujoco average	82.9	82.5	79.3	82.7		80.1	82.4	

Figures are normalised scores, with 0 and 100 representing random and expert policies, respectively. Allowing β to vary marginally enhances performance

Table 8 Additional performance comparison across D4RL benchmark

Task/data set	PBRL	RORL	SAC-RND	TD3-BC-N	SAC-BC-N
halfcheetah-v2					
-medium	57.9	66.8	66.6	63.3 ± 0.1	65.6 ± 0.2
-medium-replay	45.1	61.9	54.9	55.3 ± 0.1	61.5 ± 0.1
-medium-expert	92.3	107.8	107.6	101.7 ± 0.3	102.6 ± 0.5
-expert	92.4	105.2	105.8	103.8 ± 0.5	105.3 ± 0.1
hopper-v2					
-medium	75.3	104.8	97.8	101.5 ± 0.3	101.2 ± 0.1
-medium-replay	100.6	102.8	100.5	99.1 ± 0.1	100.8 ± 0.2
-medium-expert	110.8	112.7	109.8	112.3 ± 0.0	111.3 ± 0.1
-expert	110.5	112.8	109.7	112.7 ± 0.1	11.5 ± 0.1
walker2d-v2					
-medium	89.6	102.4	91.6	90.9 ± 0.2	85.3 ± 0.1
-medium-replay	77.7	90.4	88.7	91.4 ± 0.4	90.8 ± 0.2
-medium-expert	110.1	121.2	105.0	113.5 ± 0.1	110.9 ± 0.0
-expert	108.3	115.4	114.3	113.2 ± 0.0	110.4 ± 0.0
mujoco average	89.2	100.4	96.0	96.6	96.4
(exc. expert)	84.4	96.8	91.4	92.2	92.2
maze2d-v1					
-umaze	–	–	–	155.9 ± 1.4	155.8 ± 2.9
-medium	–	–	–	159.9 ± 2.1	163.1 ± 1.3
-large	–	–	–	219.4 ± 2.0	210.8 ± 2.0
maze2d-average	–	–	–	178.4	176.6
antmaze-v0					
-umaze	–	96.7	97.2	98.3 ± 0.7	98.6 ± 0.5
-umaze-diverse	–	90.7	83.5	90.6 ± 1.3	91.2 ± 1.3
-medium-play	–	76.3	65.5	87.0 ± 1.5	85.8 ± 1.6
-medium-diverse	–	69.3	88.5	86.2 ± 1.5	73.8 ± 2.0
-large-play	–	16.3	67.2	76.2 ± 1.9	65.8 ± 2.1
-large-diverse	–	41.0	57.6	74.2 ± 2.0	75.8 ± 1.9
antmaze average	–	65.1	76.6	85.5	81.8
adroit-v1					
-pen-cloned	74.9	35.7	–	67.2 ± 2.9	58.0 ± 2.8
-pen-human	35.4	33.7	–	72.8 ± 2.7	70.3 ± 2.9
adroit average	55.2	34.7	–	70.0	64.2

Figures are normalised scores, with 0 and 100 representing random and expert policies, respectively. For TD3-BC-N and SAC-BC-N we report the mean normalised score ± one standard error across 50 evaluations for MuJoCo tasks (10 evaluations over 5 seeds) and 500 evaluations for Maze2d, AntMaze and Adroit tasks (100 evaluations over 5 seeds)

Table 9 Computation time calculation details

Algorithm	Runtime (s/epoch*)	Total gradient steps (M)	Total runtime (h)	GPU memory (GB)
SAC-10	60	3	5.0	1.2
SAC-20	61	3	5.1	1.3
SAC-100	142	3	11.8	1.6
SAC-200	251	3	20.9	2.0
SAC-500	588	3	49.0	3.5
EDAC-10	76	3	6.3	1.2
EDAC-20	86	3	7.2	1.3
EDAC-50	136	3	11.3	1.5
MSG-4	65	2	3.6	1.2
MSG-64	131	2	7.3	1.5
CQL	123	1	3.4	1.3
IQL	54	1	1.5	1.2
TD3-BC	26	1	0.7	1.2
SAC-BC-10	62	1	1.7	1.2
TD3-BC-10	31	1	0.9	1.2

*1 epoch=10,000 gradient steps.

Algorithm 4 Online fine-tuning (SAC-BC-N)

Require: Ensemble size N, discount factor γ, minimum entropy $\mathcal{H}$, target network update rate τ, data set D and decay parameters $\beta_{start}, \beta_{end}, S$
Initialise pre-trained critic parameters θ_i, policy parameters ϕ and corresponding target parameters θ'_i.
Initialise environment and replay buffer R
Populate R with a proportion of transitions from D.
for $k = 0$ to K **do**
  Act in environment with exploration, $a \sim \pi_\phi(s)$
  Store resulting transition (s, a, r, s') in R
end for
Set decay rate κ_β as per equation (17)
Set $\beta = \beta_{start}$
for $j = 0$ to J **do**
  Act in environment with exploration, $a \sim \pi_\phi(s)$
  Store resulting transition (s, a, r, s') in R
  Sample minibatch of transitions (s, a, r, s') from R
  Update Q-function parameters θ_i using equation (11) or (12)
  Update policy parameters ϕ using equation (13) or (14)
  Update entropy parameter α using equation (15)
  Update target network parameters θ'_i using equation (16)
  Update BC coefficient $\beta = \max(\beta_{end}, \kappa_\beta \beta)$
end for

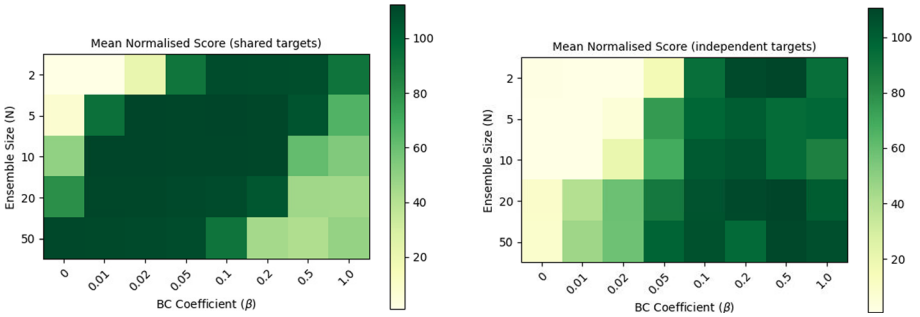

Fig. 10 Performance as a function of N and β. Lower values of β require larger values of N and smaller values of N require higher values of β. Shared targets (left)—If both N and β are large, the uncertainty in Q-value estimates for OOD actions is too high, and thus the penalty applied too severe, leading the agent to prefer actions similar to those of the data. Independent targets (right)—the decline in performance for large N and β is not observed but this may be a result of both values needing to be higher in general, and hence for even larger N and β this outcomes may also be observed

Further implementation details

As per previous works, we perform the following data transformations:

- Normalise states as per TD3-BC
- Transform AntMaze rewards according to $4(r - 0.5)$ as per MSG/CQL
- Normalise Adroit rewards as per SAC-N/EDAC

TD3-BC-N hyperparameters and network architecture

Following on from Sect. 5, we provide details of shared hyperparameters and network architecture in Table 2, and details of task specific hyperparameters for BC in Table 3.

SAC-BC-N hyperparameters and network architecture

Following on from Sect. 5, we provide details of shared hyperparameters and network architecture in Table 4, and details of task specific hyperparameters for BC in Table 5.

Hardware

The large scale experiment featured in Sect. 5.2 was conducted on a machine with Intel Xeon E5-2698 v4 CPU, 512 GB RAM and 8x Tesla V100-SXM2 32 GB GPUs

Experiments featured in Sects. 5.4 and 5.5 were conducted on a machine with Intel Core i9 9900 K CPU, 64 GB RAM and 2× NVIDIA GeForce RTX 2080Ti 11 GB TURBO GPUs.

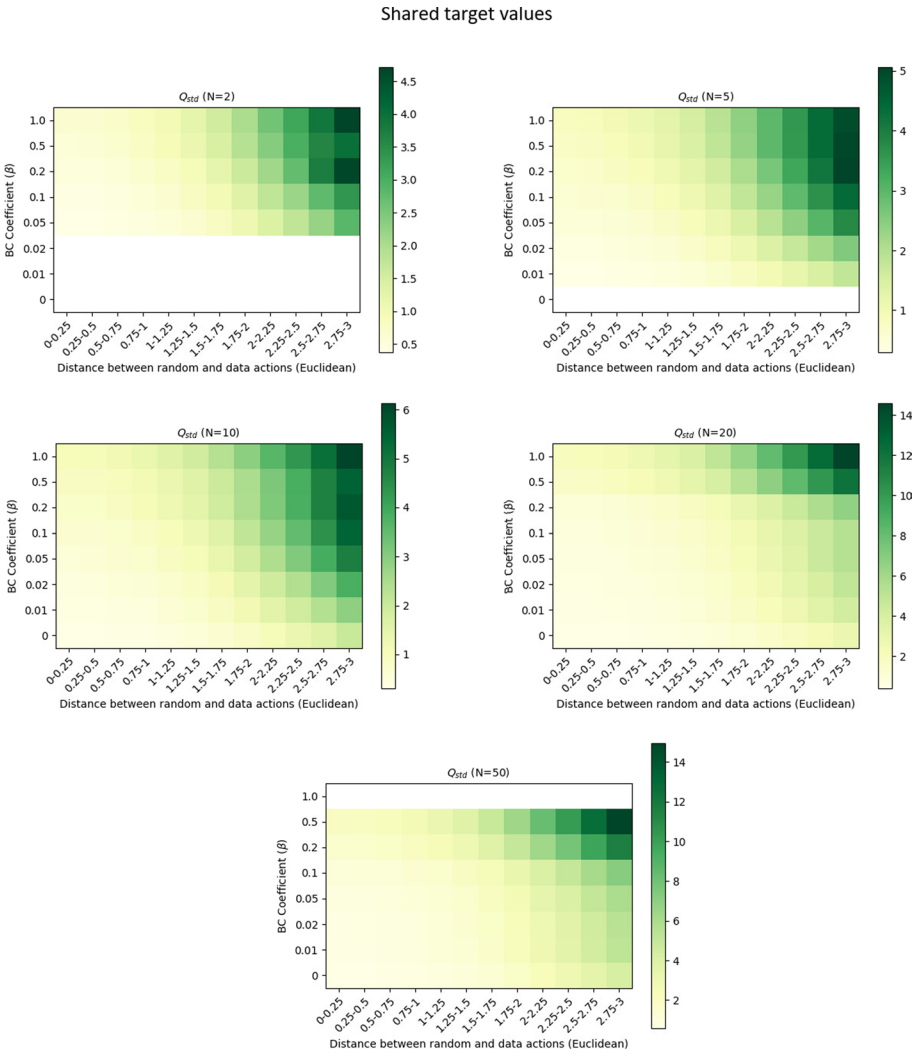

Fig. 11 Standard deviation as a function of distance, N and β (shared target values). As the distance between random and data actions increases so too does the level of uncertainty, becoming more pronounced as β and N get larger. White space is used to represent erroneous values due to unreliable Q-values estimates resulting from divergent critic loss during training

Additional experimental results

Following on from Sect. 5.1, in Table 6 we provide results for the full set of tasks from the Adroit domain using TD3-BC-N ($N = 10, \beta = 10$). As with other approaches, we are only able to attain notable performance on the "pen" task.

Following on from Sect. 5.4, in Table 7 we provide results for MuJoCo tasks allowing the value of β to vary within each task, observing a slight increase in performance. For

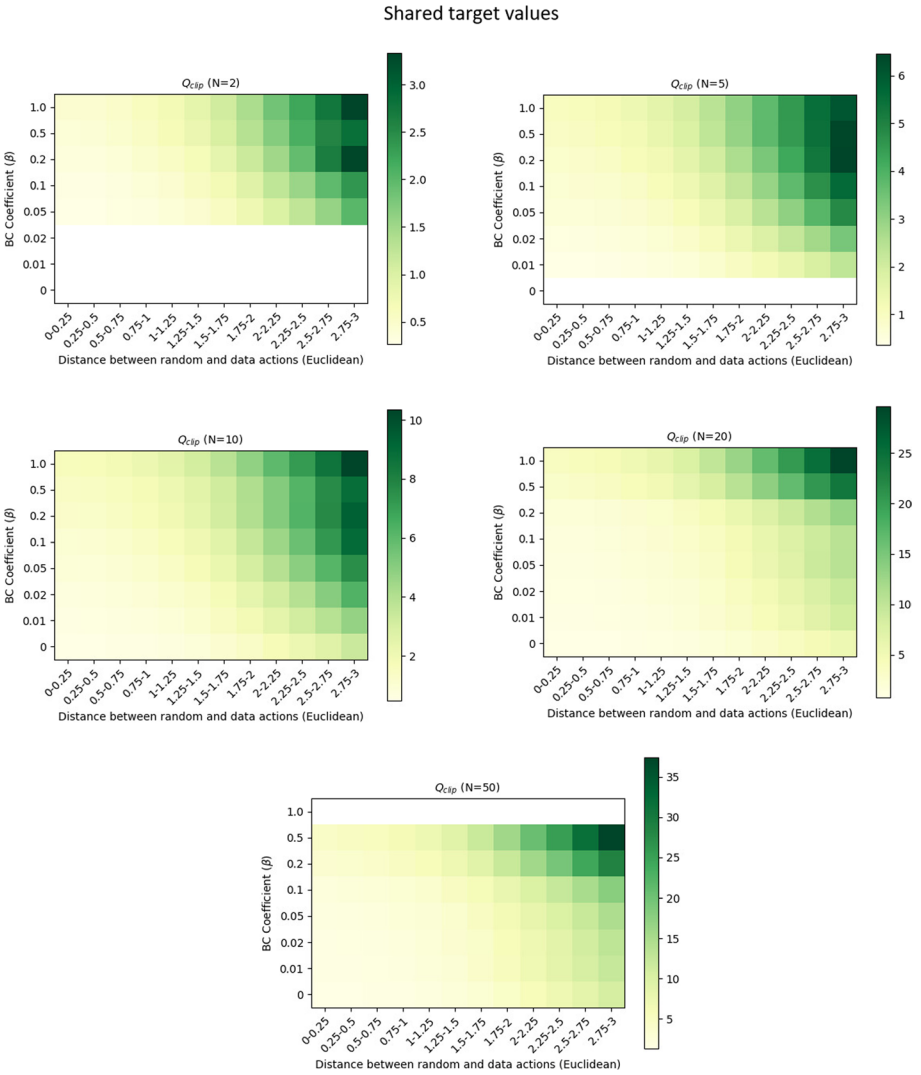

Fig. 12 Clip penalty as a function of distance, N and β (shared target values). As the distance between random and data actions increases so too does the level of uncertainty, becoming more pronounced as β and N get larger. White space is used to represent erroneous values due to unreliable Q-values estimates resulting from divergent critic loss during training

completeness, we also include results for "random" datasets, which we omit form our main results due to not being as applicable to real-world scenarios.

Following on from Sect. 5.4, in Table 8 we provide results for additional baselines mentioned in Sect. 2. Due to discrepancies between -v0 and -v2 MuJoCo datasets, we only report those using the latest versions, namely PBRL, RORL and SAC-RND. In general, we see TD3-BC-N and SAC-BC-N are competitive across all tasks and data sets, without the requirement for hyperparameter tuning within each task. We note that RORL performs

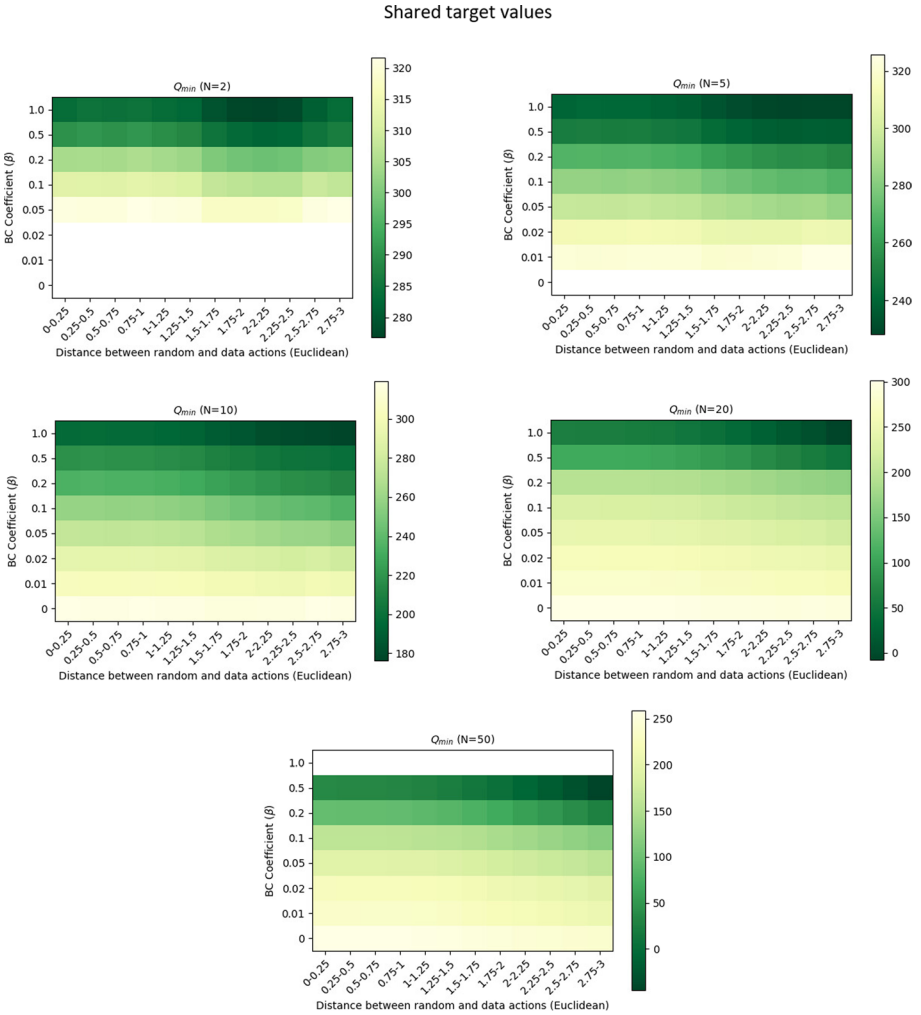

Fig. 13 Q_{min} as a function of distance, N and β (shared target values). As the distance between random and data actions increases, Q_{min} decreases, with this decrease more pronounced as β and N get larger. White space is used to represent erroneous values due to unreliable Q-values estimates resulting from divergent critic loss during training

particularly well on MuJoCo benchmarks, however we also note these results follow extensive hyperparameter tuning within each task.

In terms of computation time, based on author provided details PBRL and RORL take roughly 1.7× and 0.9× the computation time of CQL per iteration, respectively. PBRL requires 1 M gradient updates and RORL 3 M gradient updates. SAC-RND takes roughly the same amount of computation time as SAC-10 per iteration, requiring 3 M gradient updates in total. Thus, PBRL, RORL and SAC-RND are all notably less computationally efficient than TD3-BC-N and SAC-BC-N.

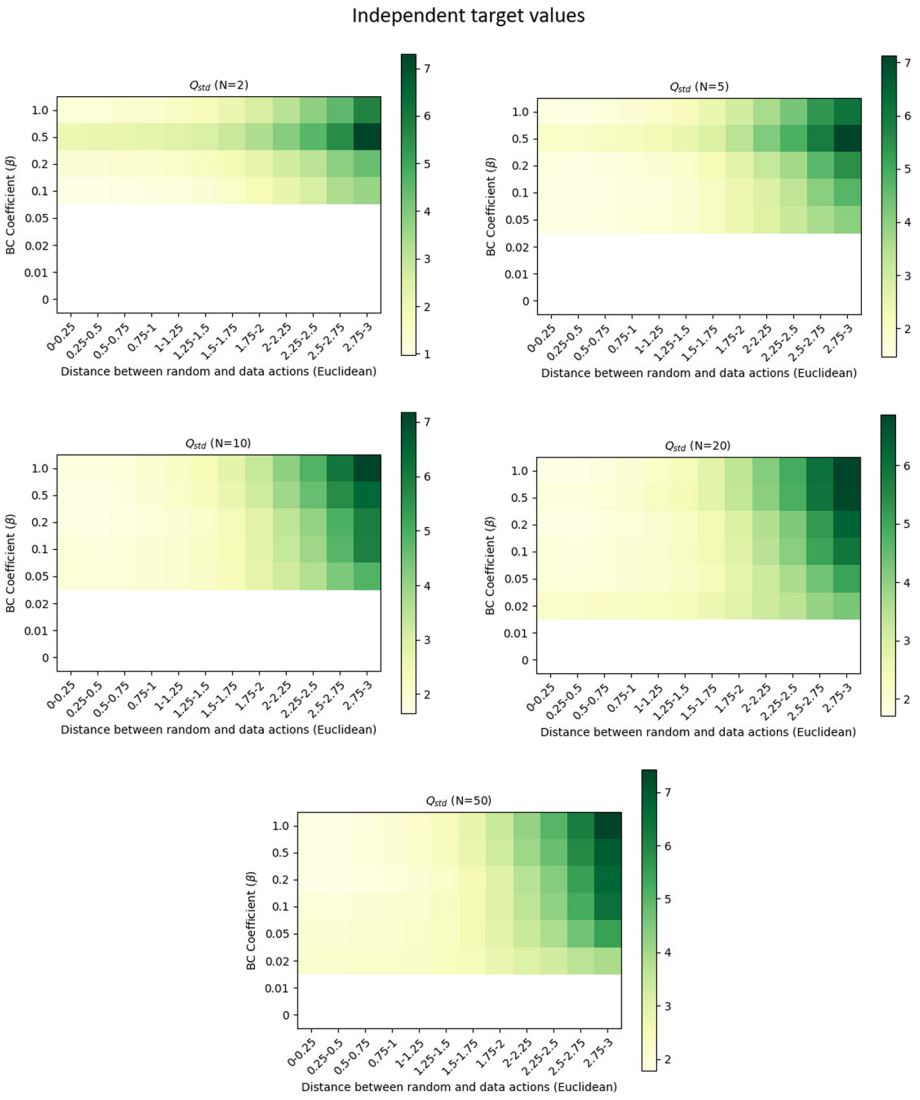

Fig. 14 Standard deviation as a function of distance, N and β (independent target values). As the distance between random and data actions increases so too does the level of uncertainty, becoming more pronounced as β and N get larger. White space is used to represent erroneous values due to unreliable Q-values estimates resulting from divergent critic loss during training

Further details regarding computational efficiency experiments

To ensure a fair comparison of computational efficiency, we implement our own versions of baselines (available in our code repository) based on author published source code and the CORL repository (Tarasov et al., 2022), and run them on the same hardware/software configuration. In terms of hardware we use a machine with a Intel Core i9 9900 K CPU,

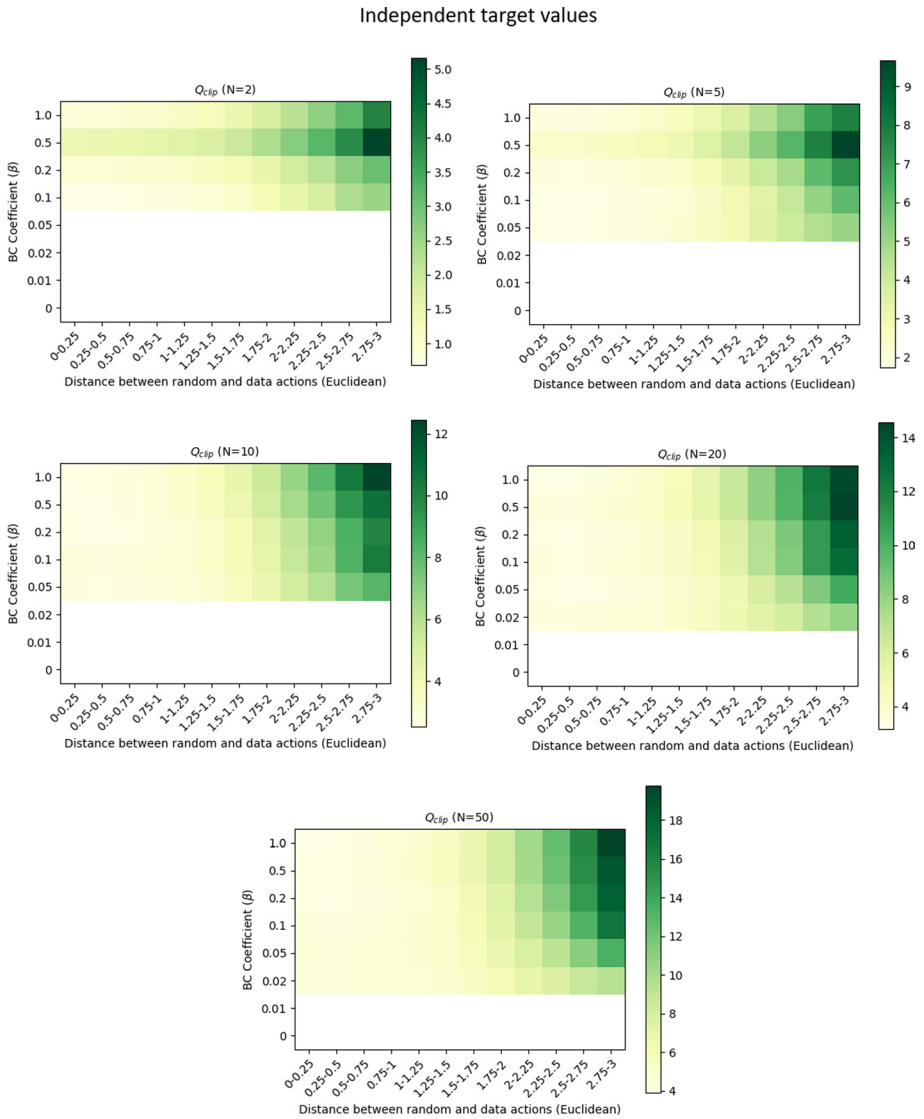

Fig. 15 Clip penalty as a function of distance, N and β (shared independent values). As the distance between random and data actions increases so too does the level of uncertainty, becoming more pronounced as β and N get larger. White space is used to represent erroneous values due to unreliable Q-values estimates resulting from divergent critic loss during training

64 GB RAM and 2× NVIDIA GeForce RTX 2080Ti 11 GB TURBO GPUs. In terms of software we use PyTorch (version 1.9.1+cu102).

The ensemble architecture for TD3-BC-N, SAC-BC-N, SAC-N, EDAC and MSG is exactly the same. Each Q-network comprises a 3-layer MLP with ReLU activation functions and 256 nodes, taking as input a state-action pair and outputting a Q-value. For TD3-BC-N the policy network comprises a 3-layer MLP with ReLU activation functions and

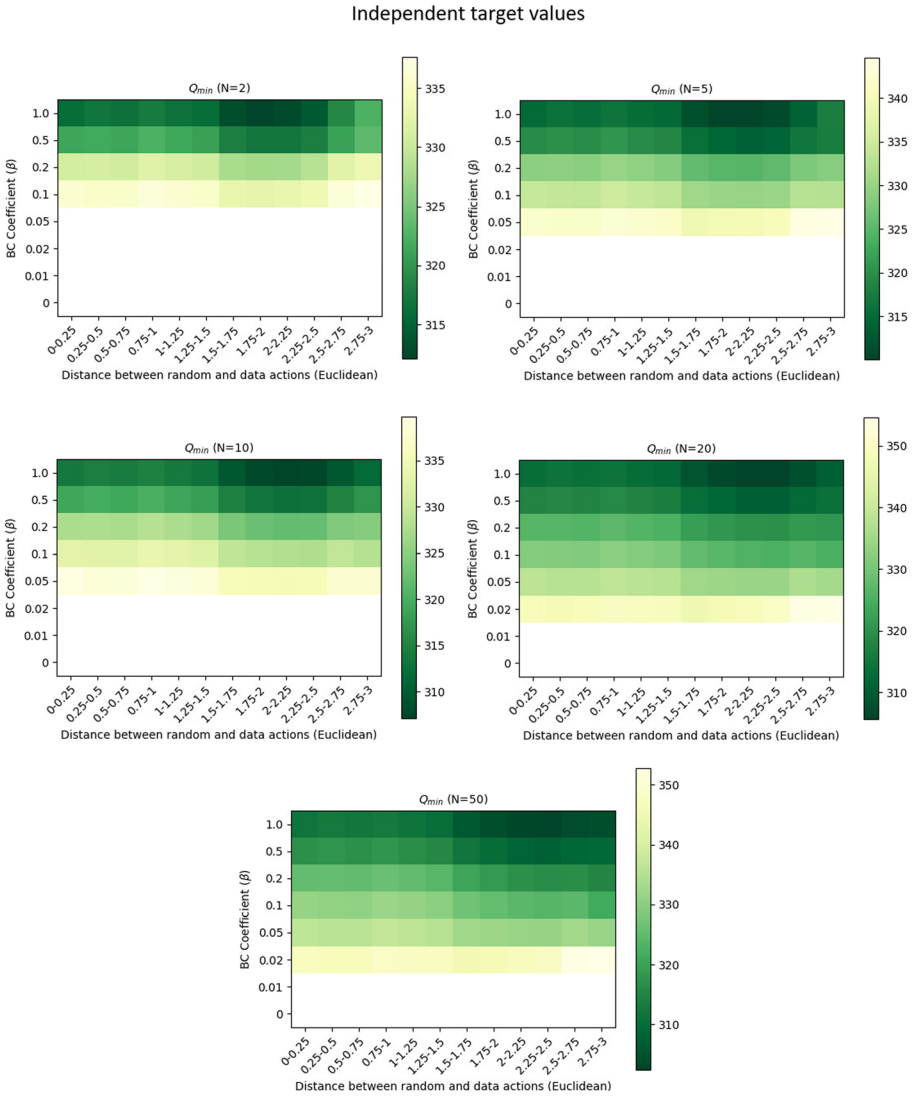

Fig. 16 Q_{min} as a function of distance, N and β (independent target values). As the distance between random and data actions increases, Q_{min} decreases, with this decrease more pronounced as β and N get larger. White space is used to represent erroneous values due to unreliable Q-values estimates resulting from divergent critic loss during training

256 nodes, taking as input a state and outputting an action bound to $[-1, 1]$ via tanh transformation. For SAC-BC-N, SAC-N, EDAC and MSG the policy network comprises the same architecture but instead outputs the mean and standard deviation of a Gaussian distribution which is also bound to $[-1, 1]$ via tanh transformation.

For CQL, we use a dual critic, with each Q-network comprising a 3-layer MLP with ReLU activation functions and 256 nodes, taking as input a state-action pair and outputting a Q-value. The policy network comprises a 3-layer MLP with ReLU activation functions

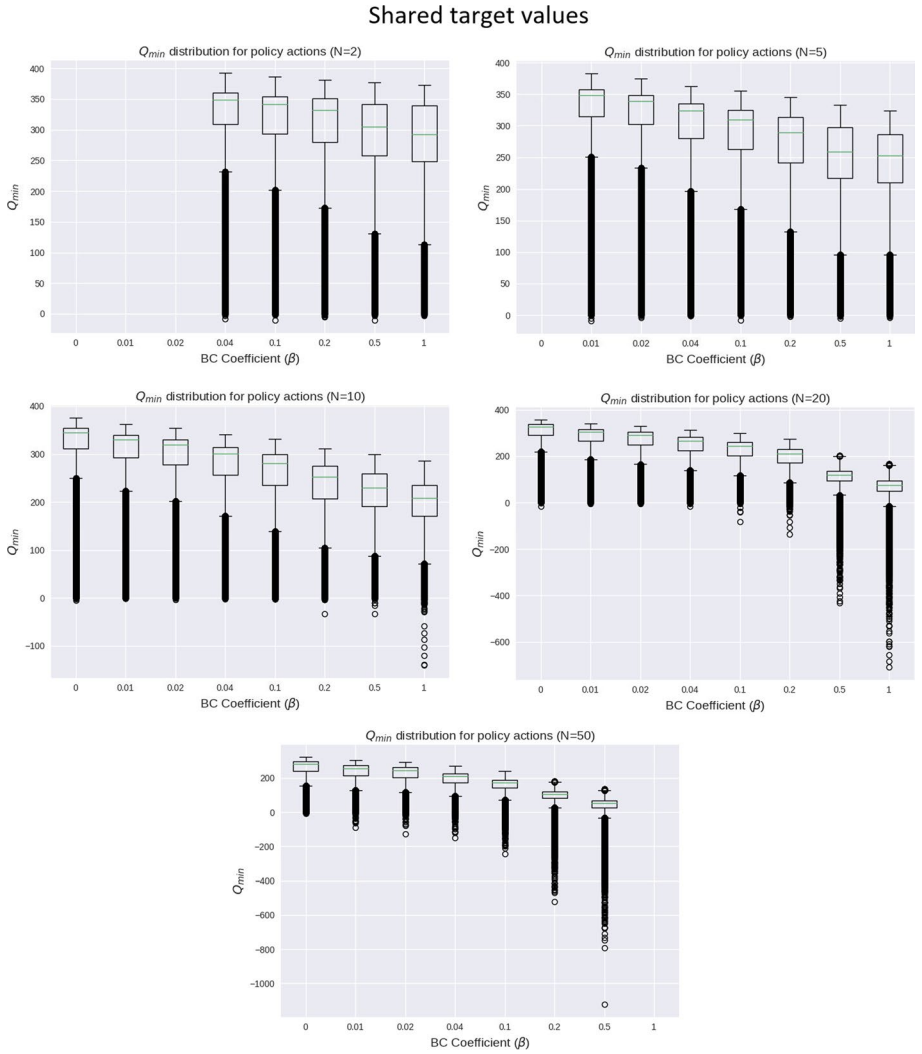

Fig. 17 Distribution of Q_{min} for policy actions (shared target values). In general, the higher the value of β the lower the values of Q_{min}, as Q-value estimates are penalised more heavily. This is particularly noticeable when N and β are large, contributing to declining performance as observed in Fig. 10

and 256 nodes outputting the mean and standard deviation of a Gaussian distribution which is bound to $[-1, 1]$ via tanh transformation.

For IQL, we use a dual critic, with each Q-network comprising a 2-layer MLP with ReLU activation functions and 256 nodes, taking as input a state-action pair and outputting a Q-value. We use a single state-value network comprising a 2-layer MLP with ReLU activation functions and 256 nodes, taking as input a state and outputting a state-value. The policy network comprises a 2-layer MLP with ReLU activation functions and 256 nodes outputting a tanh transformed mean and standard deviation of a Gaussian distribution.

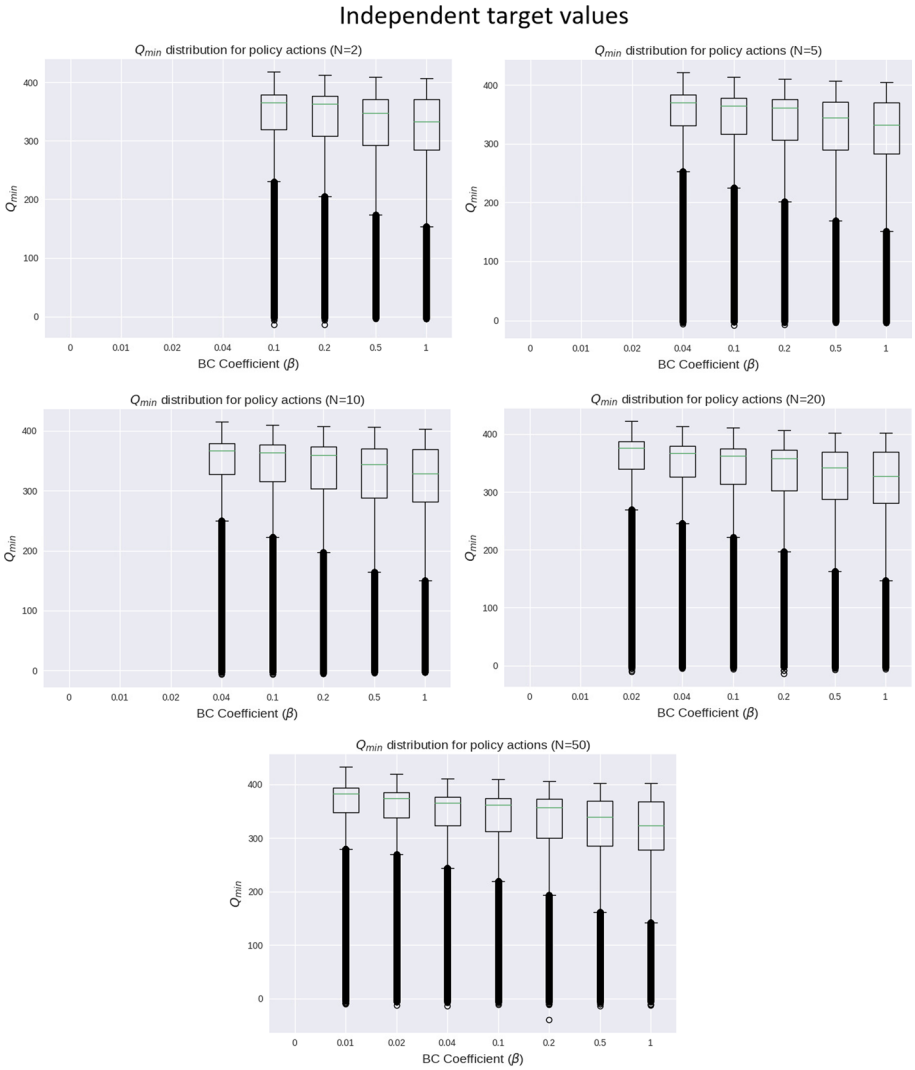

Fig. 18 Distribution of Q_{min} for policy actions (independent target values). In general, the higher the value of β the lower the values of Q_{min}, as Q-value estimates are penalised more heavily. For this range of N and β the distributions do not exhibit extreme estimates as in Fig. 17, consistent with performance as observed in Fig. 10. However, this may be the case for higher N and β

For TD3-BC, we use a dual critic, with each Q-network comprising a 2-layer MLP with ReLU activation functions and 256 nodes, taking as input a state-action pair and outputting a Q-value. The policy network comprises a 2-layer MLP with ReLU activation functions and 256 nodes, taking as input a state and outputting an action bound to $[-1, 1]$ via tanh transformation

For all algorithms we use the Adam optimiser (Kingma & Ba, 2014) and a batch size of 256.

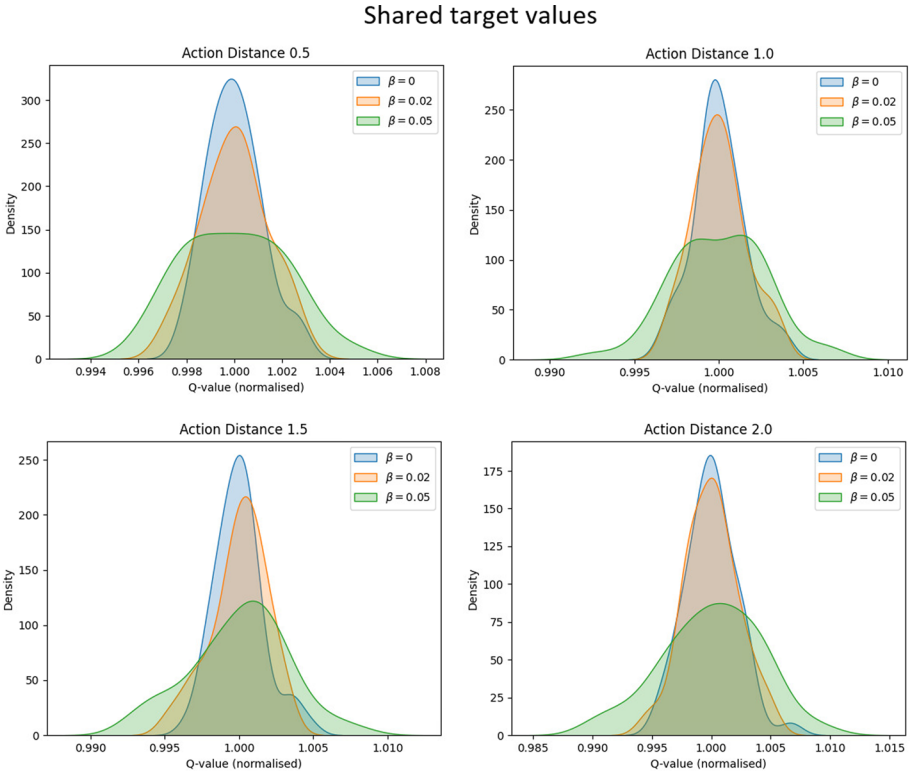

Fig. 19 Examples density estimates of Q-functions (shared target values, $N = 50$). Q-values are normalised to allow easier comparison of uncertainty. As β increases so too does the variance in Q-value estimates

For each algorithm, we record the training time for 10,000 gradient steps and scale by the total number of gradient steps to arrive at the total computation time. We detail these calculations in Table 9.

Additional plots

Following on from Sect. 5.3, we provide the complete set of plots from our case study using the "hopper-medium-expert" dataset. Figure 10 summarises performance for shared and independent target values, with Figs. 11, 12, 13, 14, 15 and 16 showing Q_{std}, Q_{clip} and Q_{min} for shared and independent target values, respectively.

We also provide plots examining the distribution of Q_{min} for policy actions in Figs. 17 and 18, and examples density estimates of Q-value distributions for individual state-action pairs in Figs. 19 and 20. To allow for better estimates of density, we use ensembles of size $N = 50$, and to allow easier comparisons of uncertainty we normalise Q-values by dividing by the mean of the absolute value across the ensemble (similar to Sect. 4).

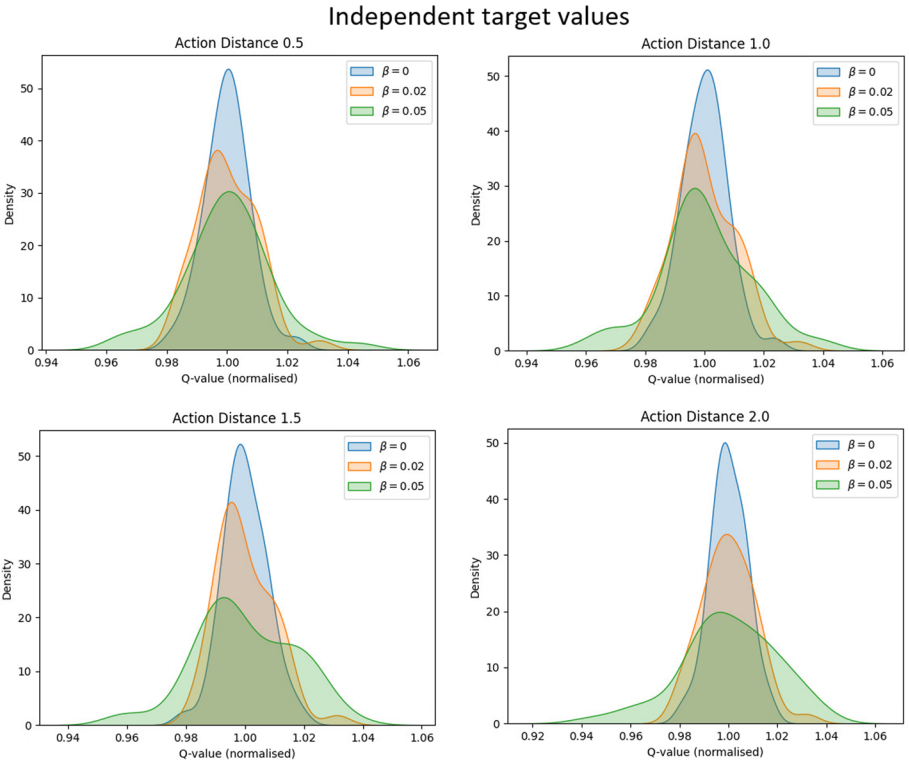

Fig. 20 Examples density estimates of Q-functions (independent target values, $N = 50$). Q-values are normalised to allow easier comparison of uncertainty. As β increases so too does the variance in Q-value estimates

Acknowledgements AB acknowledges support from University of Warwick and University of Birmingham NHS Foundation Trust. GM acknowledges support from a UKRI AI Turing Acceleration Fellowship (EPSRC EP/V024868/1). The authors acknowledge Weights & Biases (https://www.wandb.com/) as the online platform used for experiment tracking and visualizations to develop insights for this paper.

Author Contributions Authors' contributions follow the authors' order convention.

Funding AB acknowledges support from University of Warwick and University of Birmingham NHS Foundation Trust. GM acknowledges support from a UKRI AI Turing Acceleration Fellowship (EPSRC EP/V024868/1).

Availability of data and materials Benchmark data sets are open source.

Declarations

Conflict of interest No competing or financial interests to disclose.

Consent to participate The authors give their consent to participate.

Consent for publication The authors give their consent for publication.

Code availability Code base for implementation is made freely available at https://github.com/AlexBeesonWarwick/OfflineRLConstrainedEnsemble.

Ethical approval Not applicable.

References

Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U. R., et al. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion, 76*, 243–297.

An, G., Moon, S., Kim, J.-H., & Song, H. O. (2021). Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in Neural Information Processing Systems, 34*, 7436–7447.

Argenson, A., & Dulac-Arnold, G. (2020) Model-based offline planning. arXiv preprint arXiv:2008.05556

Bai, C., Wang, L., Yang, Z., Deng, Z.-H., Garg, A., Liu, P., & Wang, Z. (2022). Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning. In *International conference on learning representations.*

Bain, M., & Sammut, C. (1995). A framework for behavioural cloning. *Machine Intelligence, 15*, 103–129.

Ball, P. J., Smith, L., Kostrikov, I., & Levine, S. (2023). Efficient online reinforcement learning with offline data. arXiv preprint arXiv:2302.02948

Beeson, A., & Montana, G. (2022). Improving TD3-BC: Relaxed policy constraint for offline learning and stable online fine-tuning. arXiv preprint arXiv:2211.11802

Brandfonbrener, D., Whitney, W., Ranganath, R., & Bruna, J. (2021). Offline RL without off-policy evaluation. *Advances in Neural Information Processing Systems, 34*, 4933–4946.

Buckman, J., Gelada, C., & Bellemare, M. G. (2020). The importance of pessimism in fixed-dataset policy optimization. arXiv preprint arXiv:2009.06799

Charpentier, B., Senanayake, R., Kochenderfer, M., Günnemann, S. (2022). Disentangling epistemic and aleatoric uncertainty in reinforcement learning. arXiv preprint arXiv:2206.01558

Chen, R. Y., Sidor, S., Abbeel, P., & Schulman, J. (2017). UCB exploration via q-ensembles. arXiv preprint arXiv:1706.01502

Ciosek, K., Vuong, Q., Loftin, R., & Hofmann, K. (2019). Better exploration with optimistic actor critic. Advances in Neural Information Processing Systems **32**

Eriksson, H., Basu, D., Alibeigi, M., Dimitrakakis, C. (2022). Sentinel: Taming uncertainty with ensemble based distributional reinforcement learning. In *Uncertainty in artificial intelligence, PMLR*, pp. 631– 640.

Fu, J., Kumar, A., Nachum, O., Tucker, G., & Levine, S. (2020). D4RL: Datasets for deep data-driven reinforcement learning. arXiv preprint arXiv:2004.07219

Fujimoto, S., & Gu, S. S. (2021). A minimalist approach to offline reinforcement learning. arXiv preprint arXiv:2106.06860

Fujimoto, S., Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *International conference on machine learning, PMLR*, pp. 1587– 1596.

Fujimoto, S., Meger, D., Precup, D. (2019). Off-policy deep reinforcement learning without exploration. In *International conference on machine learning, PMLR* pp. 2052– 2062.

Ghasemipour, S. K. S., Gu, S. S., & Nachum, O. (2022). Why so pessimistic? estimating uncertainties for offline RL through ensembles, and why their independence matters. arXiv preprint arXiv:2205.13703

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning, PMLR*, pp. 1861– 1870.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., & Abbeel, P., et al. (2018). Soft actor-critic algorithms and applications. arXiv preprint arXiv:1812.05905

Havasi, M., Jenatton, R., Fort, S., Liu, J. Z., Snoek, J., Lakshminarayanan, B., Dai, A. M., & Tran, D. (2020). Training independent subnetworks for robust prediction. arXiv preprint arXiv:2010.06610

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., & Silver, D. (2018) Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence.*

Hüllermeier, E., & Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning, 110*, 457–506.

Janner, M., Du, Y., Tenenbaum, J. B., & Levine, S. (2022). Planning with diffusion for flexible behavior synthesis. arXiv preprint arXiv:2205.09991

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., & Vanhoucke, V., et al. (2018) Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. arXiv preprint arXiv:1806.10293

Kidambi, R., Rajeswaran, A., Netrapalli, P., & Joachims, T. (2020). Morel: Model-based offline reinforcement learning. arXiv preprint arXiv:2005.05951

Kingma, D. P., & Ba, J. (2014). ADAM: A method for stochastic optimization. arXiv preprint arXiv:1412.6980

Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A. A. A., Yogamani, S., & Pérez, P. (2022). Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems, 23*(6), 4909–4926.

Komorowski, M., Celi, L. A., Badawi, O., Gordon, A. C., & Faisal, A. A. (2018). The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care. *Nature Medicine, 24*(11), 1716–1720.

Kostrikov, I., Fergus, R., Tompson, J., & Nachum, O. (2021). Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning, PMLR*, pp. 5774– 5783.

Kostrikov, I., Nair, A., & Levine, S. (2021). Offline reinforcement learning with implicit q-learning. arXiv preprint arXiv:2110.06169

Kumar, A., Fu, J., Tucker, G., & Levine, S. (2019). Stabilizing off-policy q-learning via bootstrapping error reduction. arXiv preprint arXiv:1906.00949

Kumar, A., Zhou, A., Tucker, G., & Levine, S. (2020). Conservative q-learning for offline reinforcement learning. arXiv preprint arXiv:2006.04779

Lange, S., Gabel, T., & Riedmiller, M. (2012). *Batch reinforcement learning* (pp. 45–73). Berlin: Springer.

Lee, K., Laskin, M., Srinivas, A., Abbeel, P. (2021). Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International conference on machine learning, PMLR,* pp. 6131– 6141.

Lee, S., Purushwalkam, S., Cogswell, M., Crandall, D., & Batra, D. (2015). Why m heads are better than one: Training a diverse ensemble of deep networks. arXiv preprint arXiv:1511.06314

Lee, S., Seo, Y., Lee, K., Abbeel, P., & Shin, J. (2020). Addressing distribution shift in online reinforcement learning with offline datasets

Levine, S., Kumar, A., Tucker, G., & Fu, J. (2020). Offline reinforcement learning: Tutorial, review, and perspectives on open problems. arXiv preprint arXiv:2005.01643

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971

Liu, S., See, K. C., Ngiam, K. Y., Celi, L. A., Sun, X., & Feng, M. (2020). Reinforcement learning for clinical decision support in critical care: Comprehensive review. *Journal of Medical Internet Research, 22*(7), 18477.

Mahmood, A. R., Korenkevych, D., Vasan, G., Ma, W., & Bergstra, J. (2018). Benchmarking reinforcement learning algorithms on real-world robots. In *Conference on robot learning, PMLR*, pp. 561– 591.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M.(2013) Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602

Nair, A., Gupta, A., Dalal, M., & Levine, S. (2020). AWAC: Accelerating online reinforcement learning with offline datasets. arXiv preprint arXiv:2006.09359

Nair, A., Zhu, B., Narayanan, G., Solowjow, E., & Levine, S. (2022). Learning on the job: Self-rewarding offline-to-online finetuning for industrial insertion of novel connectors from vision. arXiv preprint arXiv:2210.15206

Nikulin, A., Kurenkov, V., Tarasov, D., & Kolesnikov, S. (2023). Anti-exploration by random network distillation. arXiv preprint arXiv:2301.13616

Royston, J., et al. (1982). Expected normal order statistics (exact and approximate). *Journal of the Royal Statistical Society Series C (Applied Statistics), 31*(2), 161–165.
Sohn, K., Lee, H., & Yan, X. (2015). Learning structured output representation using deep conditional generative models. *Advances in Neural Information Processing Systems, 28*, 3483–3491.
Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. Cambridge: MIT press.
Tarasov, D., Nikulin, A., Akimov, D., Kurenkov, V., & Kolesnikov, S. (2022). CORL: Research-oriented deep offline reinforcement learning library. arXiv preprint arXiv:2210.07105
Tesauro, G., et al. (1995). Temporal difference learning and td-gammon. *Communications of the ACM, 38*(3), 58–68.
Todorov, E., Erez, T., & Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International conference on intelligent robots and systems*, pp. 5026– 5033.
Wu, Y., Tucker, G., & Nachum, O. (2019). Behavior regularized offline reinforcement learning. arXiv preprint arXiv:1911.11361
Xie, T., Jiang, N., Wang, H., Xiong, C., & Bai, Y. (2021). Policy finetuning: Bridging sample-efficient offline and online reinforcement learning. *Advances in Neural Information Processing Systems, 34*, 27395–27407.
Yang, R., Bai, C., Ma, X., Wang, Z., Zhang, C., & Han, L. (2022). Rorl: Robust offline reinforcement learning via conservative smoothing. In *Advances in neural information processing systems.*
Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., & Finn, C. (2021). Combo: Conservative offline model-based policy optimization. arXiv preprint arXiv:2102.08363
Yu, C., Liu, J., Nemati, S., & Yin, G. (2021). Reinforcement learning in healthcare: A survey. *ACM Computing Surveys (CSUR), 55*(1), 1–36.
Zhang, H., Xu, W., & Yu, H. (2023). Policy expansion for bridging offline-to-online reinforcement learning. arXiv preprint arXiv:2302.00935
Zhao, Y., Boney, R., Ilin, A., Kannala, J., & Pajarinen, J. (2021). Adaptive behavior cloning regularization for stable offline-to-online reinforcement learning
Zhou, W., Bajracharya, S., & Held, D.(2020). PLAS: Latent action space for offline reinforcement learning. arXiv preprint arXiv:2011.07213
Zhou, X., Liu, H., Pourpanah, F., Zeng, T., & Wang, X. (2022). A survey on epistemic (model) uncertainty in supervised learning: Recent advances and applications. *Neurocomputing, 489*, 449–465.
Zhu, D., Wang, Y., Schmidhuber, J., & Elhoseiny, M. (2023). Guiding online reinforcement learning with action-free offline pretraining. arXiv preprint arXiv:2301.12876