arXiv:2211.01052v2 [cs.LG] 21 Nov 2022

OFFLINE RL WITH REALISTIC DATASETS: HETEROSKEDASTICITY AND SUPPORT CONSTRAINTS

Anikait Singh[1,*], **Aviral Kumar**[1,2,*], **Quan Vuong**[2], **Yevgen Chebotar**[2], **Sergey Levine**[1,2]
[1]EECS, UC Berkeley, [2]Google Research
{asap7772, aviralk}@berkeley.edu, quanhovuong@google.com

ABSTRACT

Offline reinforcement learning (RL) learns policies entirely from static datasets, thereby avoiding the challenges associated with online data collection. Practical applications of offline RL will inevitably require learning from datasets where the variability of demonstrated behaviors changes non-uniformly across the state space. For example, at a red light, nearly all human drivers behave similarly by stopping, but when merging onto a highway, some drivers merge quickly, efficiently, and safely, while many hesitate or merge dangerously. Both theoretically and empirically, we show that typical offline RL methods, which are based on distribution constraints fail to learn from data with such non-uniform variability, due to the requirement to stay close to the behavior policy **to the same extent** across the state space. Ideally, the learned policy should be free to choose **per state** how closely to follow the behavior policy to maximize long-term return, as long as the learned policy stays within the support of the behavior policy. To instantiate this principle, we reweight the data distribution in conservative Q-learning (CQL) to obtain an approximate support constraint formulation. The reweighted distribution is a mixture of the current policy and an additional policy trained to mine poor actions that are likely under the behavior policy. Our method, CQL (ReDS), is simple, theoretically motivated, and improves performance across a wide range of offline RL problems in Atari games, navigation, and pixel-based manipulation.

1 INTRODUCTION

Recent advances in offline RL (Levine et al., 2020; Lange et al., 2012) hint at exciting possibilities in learning high-performing policies, entirely from offline datasets, without requiring dangerous (Garcıa & Fernández, 2015) or expensive (Kalashnikov et al., 2018) active interaction with the environment. Analogously to the importance of data diversity in supervised learning (Deng et al., 2009), the practical benefits of offline RL depend heavily on the *coverage* of behavior in the offline datasets (Kumar et al., 2022). Intuitively, the dataset must illustrate the consequences of a diverse range of behaviors, so that an offline RL method can determine what behaviors lead to high returns, ideally returns that are significantly higher than the best single behavior in the dataset.

One easy option to attain this kind of coverage is to combine many realistic sources of data, but doing so can lead to the variety of demonstrated behaviors varying in highly non-uniform ways across the state space, i.e. the dataset is *heteroskedastic*. For example, a driving dataset might show very high variability in driving habits, with some drivers being timid and some more aggressive, but remain remarkably consistent in "critical" states (e.g., human drivers are extremely unlikely to swerve in an empty road or drive off a bridge). A good offline RL algorithm should combine the *best* parts of each behavior in the dataset – e.g., in the above example, the algorithm should produce a policy that is *as good as the best human in each situation*, which would be better than *any* human driver overall. At the same time, the learned policy should not attempt to extrapolate to novel actions in subset of the state space where the distribution of demonstrated behaviors is narrow (e.g., the algorithm should not attempt to drive off a bridge). How effectively can current offline RL methods selectively choose on a *per-state* basis how closely to stick to the behavior policy?

Most existing methods (Kumar et al., 2019; 2020; Kostrikov et al., 2021b;a; Wu et al., 2019; Fujimoto et al., 2018a; Jaques et al., 2019) constraint the learned policy to stay close to the behavior

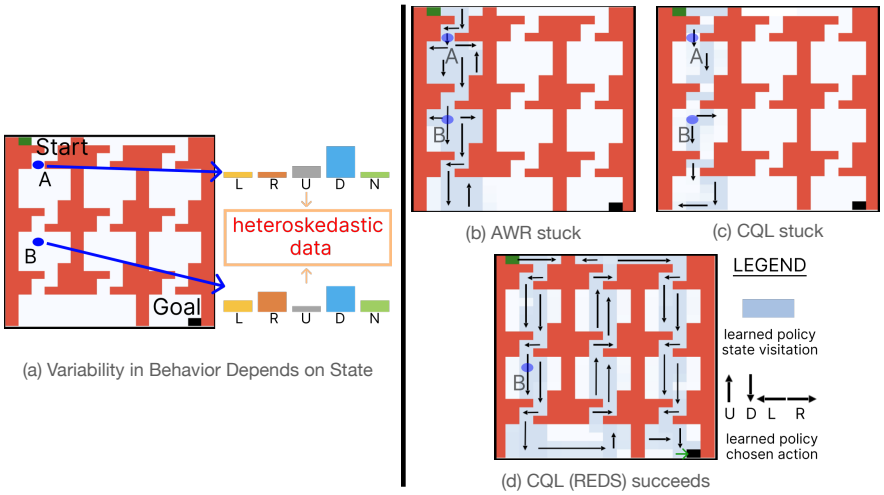

Figure 1: **Failure mode of distributional constraints.** In this navigation task, an offline RL algorithm must find a path from the start state to the goal state as indicated in (a). The offline dataset provided exhibits non-uniform coverage at different state, e.g., in the state marked as "B" located in a wide room has more uniform action distribution, whereas the states in the narrow hallways exhibit a more narrow action distribution. This is akin to how the behavior of human drivers varies in certain locations ("B"), but is very similar in other situations ("A"). To perform well, an algorithm must stay close to the data in the hallways ("A"), but deviate significantly from the data in the rooms ("B"), where the data supports many different behaviors (most are not good). AWR and CQL become stuck because they stay too close to the bad behavior policy in the rooms, e.g. the left and right arrows near State B in Fig (b) and (c). Our method, CQL (ReDS), learns to ignore the bad behavior policy action in state B and prioritizes the good action, indicated by the downward arrow near State B in (d).

policy, so-called "distributional constraints". Using a combination of empirical and theoretical evidence, we first show that distributional constraints are insufficient when the heteroskedasticity of the demonstrated behaviors varies non-uniformly across states, because the strength of the constraint is state-agnostic, and may be overly conservative at some states even when it is not conservative enough at other states. We also devise a measure of heteroskedasticity that enables us to determine if certain offline datasets would be challenging for distributional constraints.

Our second contribution is a simple and theoretically-motivated observation: distribution constraints against a *reweighted* version of the behavior policy give rise to support constraints. That is, the return-maximization optimization process can freely choose state-by-state how much the learned policy should stay close to the behavior policy, so long as the learned policy remains within the data support. We show that it is particularly convenient to instantiate this insight on top of conservative Q-learning (CQL) (Kumar et al., 2020), a recent offline RL method. The new method, CQL (ReDS), only changes minimally the form of regularization, design decisions employed by CQL and inherits existing hyper-parameter values. CQL (ReDS) attains better performance than recent distribution constraints methods on a variety of tasks with more heteroskedastic distributions.

2 Preliminaries on Distributional Constraints For offline RL

The goal in offline RL is find the optimal policy in a Markov decision process (MDP) specified by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, r, \mu_0, \gamma)$. $\mathcal{S}, \mathcal{A}$ denote the state and action spaces. $T(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ and $r(\mathbf{s}, \mathbf{a})$ represent the dynamics and reward function. $\mu_0(s)$ denotes the initial state distribution. $\gamma \in (0, 1)$ denotes the discount factor. The goal is to learn a policy that maximizes the return, denoted by $J(\pi) := \frac{1}{1-\gamma}\mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t)\sim\pi}[\sum_t \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)]$. We must find the best possible policy while only having access to an offline dataset of transitions collected using a behavior policy π_β, $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, r, \mathbf{s}')\}$.

Offline RL via distributional constraints. Most offline RL algorithms regularize the learned policy π from querying the target Q-function on unseen actions (Fujimoto et al., 2018a; Kumar et al., 2019), either implicitly or explicitly. For our theoretical analysis, we will abstract the behavior of distributional constraint offline RL algorithms into a generic formulation following Kumar et al. (2020). As shown in Equation 1, we consider the problem where we must maximize the return of

the learned policy π (in the empirical MDP) $\widehat{J}(\pi)$, while also penalizing the divergence from π_β:

$$\max_{\pi} \;\; \mathbb{E}_{\mathbf{s}\sim\widehat{d}^{\pi}}\left[\widehat{J}(\pi) - \alpha D(\pi,\pi_\beta)(\mathbf{s})\right] \qquad \text{(generic distributional constraint)} \tag{1}$$

where D denotes a divergence between the learned policy π and the behavior policy π_β at state $\mathbf{s}$.

Conservative Q-learning. (Kumar et al., 2020) enforces the distributional constraint on the policy *implicitly*. To see why this is the case, consider the CQL objective, which consists of two terms:

$$\min_{\theta} \; \underbrace{\alpha\left(\mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\pi}\left[Q_\theta(\mathbf{s},\mathbf{a})\right] - \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}\left[Q_\theta(\mathbf{s},\mathbf{a})\right]\right)}_{\mathcal{R}(\theta)} + \frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q_\theta(\mathbf{s},\mathbf{a}) - \mathcal{B}^\pi\bar{Q}(\mathbf{s},\mathbf{a})\right)^2\right], \tag{2}$$

where $\mathcal{B}^\pi\bar{Q}(\mathbf{s},\mathbf{a})$ is the Bellman backup operator applied to a delayed target Q-network, $\bar{Q}$: $\mathcal{B}^\pi\bar{Q}(\mathbf{s},\mathbf{a}) := r(\mathbf{s},\mathbf{a}) + \gamma\mathbb{E}_{\mathbf{a}'\sim\pi(\mathbf{a}'|\mathbf{s}')}[\bar{Q}(\mathbf{s}',\mathbf{a}')]$. The second term (in blue) is the standard TD error (Lillicrap et al., 2015; Fujimoto et al., 2018b; Haarnoja et al., 2018). The first term $\mathcal{R}(\theta)$ (in red) attempts to prevent overestimation in the Q-values for out-of-distribution (OOD) actions by minimizing the Q-values under a distribution $\mu(\mathbf{a}|\mathbf{s})$, which is automatically chosen to pick actions with high Q-values $Q_\theta(\mathbf{s},\mathbf{a})$, and counterbalances by maximizing the Q-values of the actions in the dataset. Kumar et al. (2020) show that Equation 2 gives rise to a pessimistic Q-function that modifies the optimal Q function by the ratios of densities, $\pi(\mathbf{a}|\mathbf{s})/\pi_\beta(\mathbf{a}|\mathbf{s})$ at a given state-action pair $(\mathbf{s},\mathbf{a})$. More formally, the Q-function obtained after one iteration of TD-learning is given by:

$$Q_\theta(\mathbf{s},\mathbf{a}) := \mathcal{B}^\pi\bar{Q}(\mathbf{s},\mathbf{a}) - \alpha\left[\frac{\pi(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1\right]. \tag{3}$$

The Q function is unchanged only if the density of the learned policy π matches that of the behavior policy π_β. Otherwise, for state-action pairs where $\pi(\mathbf{a}|\mathbf{s}) < \pi_\beta(\mathbf{a}|\mathbf{s})$, Eq. 3 increases their Q values and encourages the policy π to assign more mass to the action. Vice versa, if $\pi(\mathbf{a}|\mathbf{s}) > \pi_\beta(\mathbf{a}|\mathbf{s})$, Eq. 3 encourages the policy π to assign smaller density to the action $\mathbf{a}$.

In Eq. 3, α is a constant for every state, and hence the value function learned by CQL is altered by the ratio of action probabilities to the same extent at all possible state-action pairs. As we will discuss in the next section, this can be sub-optimal when the learnt policy should stay close to the behavior policy in some states, but not others. We elaborate on this intuition in the next section.

3 Why Distribution Constraints Fail with Heteroskedastic Data

In statistics, heteroskedasticity is typically used to refer to conditions when the standard deviation in a given random variable varies non-uniformly over time (see for example, Cao et al. (2020)). We call a offline dataset heteroskedastic when the variability of the behavior differs drastically in different regions of the state space: for instance, if for certain regions of the state space, the observed behaviors in the dataset assign the most probability mass to a few actions, but in other regions, the observed behaviors are more uniform. Realistic sources of offline data are often heteroskedastic. This is because datasets are typically generated by multiple policies, each with its own characteristics under a variety of different conditions. As an example, driving datasets are typically collected from multiple humans (Ettinger et al., 2021), and many robotic manipulation datasets are collected by multiple scripted policies (Mandlekar et al., 2021) or teleoperators (Ebert et al., 2021) that aim to solve a variety of tasks, resulting in systematic variability in different regions of the state space.

3.1 A Didactic Example

To understand why distributional constraints are insufficient when learning from heteroskedastic data, we present a didactic example. Motivated by the driving scenario in the introduction, we consider a maze navigation task shown in Fig. 1. The task is to navigate from the position labeled as "Start" to the position labeled as "Goal" using five actions at every possible state (L: $\leftarrow$, R: $\rightarrow$, U: $\uparrow$, D: $\downarrow$, No: No Op), while making sure that the executed actions do not hit the walls of the grid.

Dataset construction. In order to collect a heteroskedastic dataset, we consider a mixture of several behavior policies that attain a uniform occupancy over different states in the maze. However, the dataset action distributions differ significantly in different states. The induced action distribution is heavily biased to move towards the goal in the narrow hallways (e.g., the behavior policy moves upwards at state A)). In contrast, the action distribution is quite diverse in the wider rooms. In these rooms, the behavior policy often selects actions that do not immediately move the agent towards the goal (e.g., the behavior policy at state B), because doing so does not generally hit the walls as

the rooms are wider, and hence the agent is not penalized. On the other hand, the agent must take utmost precaution to not hit the walls in the narrow hallways. More details are in Appendix A.

Representative distributional constraints algorithms such as AWR (Peng et al., 2019; Nair et al., 2020b) and CQL (Kumar et al., 2020) fail to perform the task, as shown in Figure 1. To ensure fair comparison, we tune each method to its best evaluation performance using online rollouts. The visualization in Figure 1 demonstrates that these two algorithms fail to learn reasonable policies because the learned policies match the random behavior of the dataset actions too closely in the wider rooms, and therefore are unable to make progress towards the Goal position. This is a direct consequence of enforcing too strong of a constraint on the learned policy to stay close to the behaviors in the dataset. Therefore, we also evaluated the performance of CQL and AWR in this example, with lower amounts of conservatism. As shown in Appendix A, utilizing a lower amount of conservatism suffers from the opposite failure mode: it is unable to prevent the policies from hitting the walls in the narrow hallways. To summarize, this means that choosing to be conservative prevents the algorithm from making progress in the regions where the behavior in the dataset is more diverse, whereas choosing to not be conservative enough hurts performance in regions where the behaviors in the dataset agree with each other. In contrast, the method we propose in this paper to tackle this precise challenge of heteroskedasticity, indicated as "CQL (ReDS)", effectively traverses the maze and attains a success rate of 80%.

3.2 Characterizing the Challenges with Distributional Constraints

Having seen that distribution constraints can fail in certain scenarios, we now formally characterize when offline RL datasets is heteroskedastic, and why distribution constraints may be ineffective in such scenarios. Similar to how standard analyses utilize concentrability coefficient (Rashidinejad et al., 2021), which upper bounds the ratio of state-action visitation under a policy $d^\pi(\mathbf{s}, \mathbf{a})$ and the dataset distribution μ, i.e., $\max_{\mathbf{s},\mathbf{a}} d^\pi(\mathbf{s}, \mathbf{a})/\mu(\mathbf{s}, \mathbf{a}) \leq C^\pi$, we introduce a new metric called *differential concentrability*, which measures dataset heteroskedasticity (i.e., the variability in the dataset behavior across different states).

Definition 3.1 (Differential concentrability.)**.** *Given a divergence D over the action space, the differential concentrability of a given policy π with respect to the behavioral policy π_β is given by:*

$$C^\pi_{\textit{diff}} = \mathop{\mathbb{E}}_{\mathbf{s}_1,\mathbf{s}_2\sim d^\pi} \left[\left(\sqrt{\frac{D(\pi,\pi_\beta)(\mathbf{s}_1)}{\mu(\mathbf{s}_1)}} - \sqrt{\frac{D(\pi,\pi_\beta)(\mathbf{s}_2)}{\mu(\mathbf{s}_2)}} \right)^2 \right]. \tag{4}$$

Eq. 4 measures the variation in the divergence between a given policy $\pi(\mathbf{a}|\mathbf{s})$ and the behavior policy $\pi_\beta(\mathbf{a}|\mathbf{s})$ weighted inversely by the density of these states in the offline dataset (i.e., $\mu(\mathbf{s})$ in the denominator). For simplicity, let us revisit the navigation example from Section 3.1 and first consider the simpler scenario where $\mu(\mathbf{s}) = \text{Unif}(\mathcal{S})$. For any given policy π, if there are states where π chooses actions that lie on the fringe of the data distribution (e.g., in the wider rooms), as well as states where the policy π chooses actions at the mode of the data distribution (e.g., as in the narrow passages), then C^π_{diff} would be large for the policy π learned using distributional constraints. Crucially, C^π_{diff} would be small even if the learned policy π deviates significantly from the behavior policy π_β, such that $D(\pi,\pi_\beta)(\mathbf{s})$ is large, but $|D(\pi,\pi_\beta)(\mathbf{s}_1) - D(\pi,\pi_\beta)(\mathbf{s}_2)|$ is small, indicating the dataset is not heteroskedastic. We will show in Section 5 that arbitrary policy checkpoints π learned by offline RL algorithms generally attain a low value of C^π_{diff} on offline datasets from non-heteroskedastic sources, such as those covered in the D4RL (Fu et al., 2020) benchmark.

We now use the definition of differential concentrability above to bound both the improvement and deprovement of π w.r.t. π_β for distribution constraint algorithms using the framework of safe policy improvement (Laroche et al., 2017; Kumar et al., 2020). We show that when C^π_{diff} is large, then distributional constraints (Eq. 1) may not improve significantly over π_β, even for the best value for the weight α (proof in Appendix B):

Theorem 3.1 (Informal; Limited policy improvement via distributional constraints.)**.** *W.h.p. $\geq 1-\delta$, for any prescribed level of safety ζ, the maximum possible policy improvement over choices of α, $\max_\alpha \; [J(\pi_\alpha) - J(\pi_\beta)] \leq \zeta^+$, where ζ^+ is given by:*

$$\zeta^+ := \max_\alpha \quad \frac{h^*(\alpha)}{(1-\gamma)^2} \quad \textit{s.t.} \quad \frac{c_1\sqrt{\log\frac{|\mathcal{S}||\mathcal{A}|}{\delta}}}{(1-\gamma)^2}\frac{\sqrt{C^{\pi_\alpha}_{\textit{diff}}}}{|\mathcal{D}|} - \frac{\alpha}{1-\gamma}\mathbb{E}_{\mathbf{s}\sim\widehat{d}^{\pi_\alpha}}[D(\pi_\alpha,\pi_\beta)(\mathbf{s})] \leq \zeta, \tag{5}$$

where h^* is a monotonically decreasing function of α, and $h(0) = \mathcal{O}(1)$.

Theorem 3.1 quantifies the fundamental tradeoff with distribution constraints: to satisfy a given ζ-safety constraint in problems with larger C^{π}_{diff}, we would need a larger α. Since the maximum policy improvement ζ^{+} is upper bounded by $h^{*}(\alpha)$, the policy may not necessarily improve over the behavior policy if α is large. On the flip side, if we choose to fix the value of α to be small in hopes to attain more improvement in problems where C^{π}_{diff} is high, we would end up compromising on the safety guarantee as ζ needs to be large for a small α and large C^{π}_{diff}. Thus, in this case, the policy may not improve over the behavior policy reliably.

Note that larger C^{π}_{diff} need not imply large $\mathbb{E}_{\mathbf{s}\sim\widehat{d}^{\pi}}[D(\pi,\pi_{\beta})(\mathbf{s})]$ because the latter does not involve $\mu(\mathbf{s})$. C^{π}_{diff} also measures the dispersion of $D(\pi,\pi_{\beta})(\mathbf{s})$, while the latter performs a mean over states. In addition, Theorem 3.1 characterizes the *maximum possible* improvement with an *oracle* selection of α, though is not feasible in practice. Thus, when C^{π}_{diff} is large, distribution constraint algorithms could either not safely improve over π_{β} or would attain only a limited, bounded improvement.

Remark 1 [Heteroskedasticity vs. distributional shift]. Note that larger C^{π}_{diff} need not necessarily imply large $\mathbb{E}_{\mathbf{s}\sim\widehat{d}^{\pi}}[D(\pi,\pi_{\beta})(\mathbf{s})]$ because the latter does not involve $\mu(\mathbf{s})$. C^{π}_{diff} also measures the dispersion of $D(\pi,\pi_{\beta})(\mathbf{s})$, while the latter performs a mean over states.

Remark 2 [Oracle choice of α]. Note that Theorem 3.1 characterizes the *maximum possible* improvement with an *oracle* selection of α, though is not feasible in practice. Thus, when C^{π}_{diff} is large, distribution constraint algorithms could either not safely improve over π_{β} or would attain only a limited, bounded improvement, for any given α.

Remark 3 [Comparison with prior results]. Complementing prior works (Kumar et al., 2019; Levine et al., 2020) that discuss failure modes of distributional constraints with high-entropy behavior policies, Theorem 3.1 quantifies when this would be the case: this happens when C^{π}_{diff} is large. Our analysis also indicates that when C^{π}_{diff} is small, we might be able to observe reasonable performance with distributional constraints and this potentially serves as the justification for the superior empirical results of distributional constraints, compared to support constraints on the D4RL (Fu et al., 2020) tasks, which as we show are not heteroskedastic.

4 Support Constraints As Reweighted Distribution Constraints

Thus far, we have seen that distribution constraints can be ineffective with heteroskedastic datasets. If we can impose the distribution constraints such that the constraint strength can be modulated per state, then in principle, we can alleviate the issue raised in Theorem 3.1.

Our key insight is that by reweighting the action distribution in the dataset before utilizing a distribution constraint, we can obtain a method that enforces a per-state distribution constraint, which corresponds to an approximate *support* constraint. This will push down the values of actions that are outside the behavior policy support, but otherwise not impose a severe penalty for in-support actions, thus enabling the policy to deviate from the behavior policy by different amounts at different states. Consider the generic distribution constraint shown in Eq. 1. Rather than having a distribution constraint between π and π_{β}, if we can impose a constraint between π and a *reweighted* version of π_{β}, where the reweighting is state-dependent, then we can obtain an approximate support constraint. Let the reweighted distribution be π^{re}. Intuitively, if $\pi(\cdot|\mathbf{s})$ is within the support of the $\pi_{\beta}(\cdot|\mathbf{s})$, then one can find a reweighting $\pi^{re}(\cdot|\mathbf{s})$ such that $D(\pi,\pi^{re})(\mathbf{s}) = 0$, whereas if $\pi(\cdot|\mathbf{s})$ is not within the support of $\pi^{re}(\cdot|\mathbf{s})$, then $D(\pi,\pi^{re})(\mathbf{s})$ still penalizes π when π chooses out-of-support actions, since no reweighting π^{re} can put non-zero probability on out-of-support actions. This allows us to handle the failure mode from Section 3: at states with wide behavior policy, even with a large α, π is not anymore constrained to the behavior policy distribution, whereas at other "critical" states, where the behavior policy is narrow, a large enough α will constrain $\pi(\cdot|\mathbf{s})$ to stay close to $\pi_{\beta}(\cdot|\mathbf{s})$. We call this approach **Re**weighting **D**istribution constraints to **S**upport (ReDS).

4.1 Instantiating the Principle Behind ReDS

How can we derive a reweighting of the behavior policy to instantiate an approximate support constraint? One option is reweight π_{β} into π^{re}, and enforce the constraints between π and π^{re} explicitly, by minimizing $D(\pi,\pi^{re})$, or implicitly, by pushing up the Q-values under π^{re} instead of π_{β}, analogously to CQL. Both choices are problematic, because the error in estimating the behavior policy can propagate, leading to poor downstream RL performance, as demonstrated in prior work (Nair et al., 2020b; Ghasemipour et al., 2021). For the case of CQL, this issue might be especially severe if

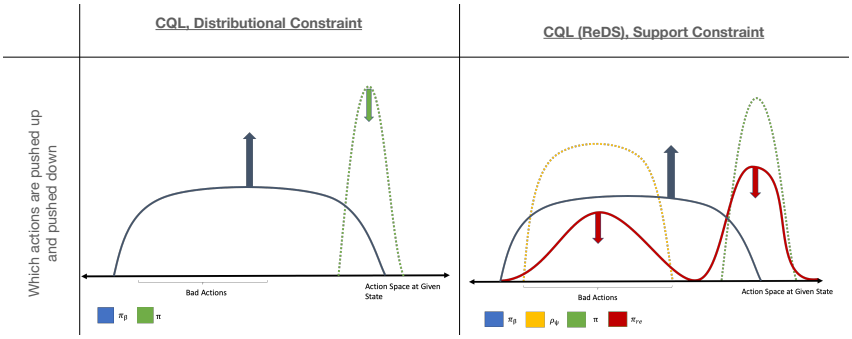

Figure 2: **Comparison between support and distributional constraints: Left:** CQL pushes down the Q-function under the policy π, while pushing up the function under the behavior policy π_β. This means that the Q-values for bad actions can go up **Right:** In contrast, ReDS re-weights the data distribution to push down the values of bad actions, alleviating this shortcoming.

we push up the Q-values under π^{re}, because then the errors in estimating the behavior policy might lead to severe over-estimation.

Instead, we devise an alternative formulation for ReDS that re-weights the learned policy such that applying a distributional constraint on this re-weighting imposes a support constraint. In this formulation, we instead *push down* the Q-values under π^{re}. To still enable approximate support constraints between π and π_β, we define π^{re} as a mixture distribution of the learned policy π and the reweighted behavior policy as shown below. We then demonstrate how π^{re} can modify CQL to enable approximate support constraints while reusing all existing algorithmic components.

$$\pi^{re}(\cdot|\mathbf{s}) := \frac{1}{2}\pi(\cdot|\mathbf{s}) + \frac{1}{2}\left[\pi_\beta(\cdot|\mathbf{s}) \cdot g\left(\pi(\cdot|\mathbf{s})\right)\right], \tag{6}$$

where $g(\cdot)$ is a monotonically decreasing function. As Fig. 2 depicts, the second term in Eq. 6 increases the probability of actions that are likely under the behavior policy, but are less likely under the learned policy (due to g being a decreasing function). We will show in Lemma 4.1 that utilizing π^{re} in CQL enforces a support constraint on π. Thus, the learned policy π can be further away from π_β, allowing π to assign more probability to good actions that are within the behavior policy support, even if they have lower probabilities under π_β. Section 4.2 illustrates theoretically why pushing down the Q-values under Eq. 6 approximates a support constraint in terms of how it modifies the resulting Q-values. For an illustration, please see Figure 3.

We must pick g to devise a complete offline algorithm. A practical choice that reuses existing components in CQL is to subsume π_β and g into one distribution, denoted by ρ (additional justification in the next section). More concretely, π^{re} now mixes π and ρ (the equal weight of 1/2 is an arbitrary choice and any non-zero weight on ρ can be used) as seen in Eq. 7. Utilizing a mixture of π and a reweighted $\pi_\beta \cdot g$ is crucial, allowing for two important properties. Firstly, the inclusion of π in the mixture ensures that the learned policy π is penalized for deviating outside the support of the behavior policy, π_β. Second, the inclusion of the re-weighted behavior policy $\pi_\beta \cdot g(\mathbf{a}|\mathbf{s})$ allows π to assign more density to good actions that are within the support of π_β, even if these actions have lower densities under π_β. This ensures that π is constrained to lie within the support of π_β, but is not constrained to the behavior policy distribution.

$$\pi^{re}(\cdot|\mathbf{s}) := \frac{1}{2}\pi(\cdot|\mathbf{s}) + \frac{1}{2}\rho(\cdot|\mathbf{s}) \tag{7}$$

ReDS modifies the regularizer $\mathcal{R}(\theta)$ in CQL (Eq.3) to utilize the mixture distribution ρ from Eq. 7, leading to the following modified regularizer:

$$\mathcal{R}(\theta; \rho) = \left(\frac{1}{2}\mathop{\mathbb{E}}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\pi}\left[Q_\theta(\mathbf{s},\mathbf{a})\right] + \frac{1}{2}\mathop{\mathbb{E}}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\rho}\left[Q_\theta(\mathbf{s},\mathbf{a})\right] - \mathop{\mathbb{E}}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}\left[Q_\theta(\mathbf{s},\mathbf{a})\right]\right) \tag{8}$$

We will show that Equation 8 imposes an approximate support constraint in Lemma 4.1. Subsuming π_β and g into ρ means the optimal ρ should satisfy $\rho(\cdot|\mathbf{s}) = \pi_\beta(\cdot|\mathbf{s}) \cdot g\left(\pi(\cdot|\mathbf{s})\right)$. Under the MaxEnt RL framework (Levine, 2018), policy probabilities are the exponentiated advantages: $\pi(\mathbf{a}|\mathbf{s}) = \exp(A(\mathbf{s},\mathbf{a})) = \exp(Q(\mathbf{s},\mathbf{a}) - V(\mathbf{s}))$. As such, we can utilize the advantage function (derived from Q-function) directly to train ρ, instead of using the policy. That is, $\rho(\cdot|\mathbf{s}) =$

$\pi_\beta(\cdot|\mathbf{s}) \cdot g(\exp(A(\mathbf{s}, \mathbf{a})))$. As shown in prior work on reward-weighted and advantage-weighted policy learning (e.g., AWR (Peng et al., 2019)), we can solve for this distribution using a weighted maximum log-likelihood objective. We choose $g(x) = 1/x$, such that $g(\exp(A(\mathbf{s}, \mathbf{a}))) = \exp(-A(\mathbf{s}, \mathbf{a}))$ (which is a decreasing function). As is common with such weighted regression methods (Peng et al., 2019), we add a temperature τ into the exponent, yielding the following objective for ρ:

$$\rho_\psi(\cdot|\mathbf{s}) = \arg\max_{\rho_\psi} \; \mathbb{E}_{\mathbf{s}\sim\mathcal{D}, \mathbf{a}\sim\pi_\beta(\cdot|\mathbf{s})} \left[\log \rho_\psi(\mathbf{a}|\mathbf{s}) \cdot \exp\left(-A_\theta(\mathbf{s}, \mathbf{a})/\tau\right)\right]. \tag{9}$$

The crucial difference between this objective and standard advantage-weighted updates is the difference of the sign. While algorithms such as AWR aim to find an action that attains a high advantage while being close to the behavior policy, and hence, uses a positive advantage, we utilize the *negative* advantage to mine for poor actions that are still quite likely under the behavior policy. The final objective for the Q-function combines the regularizer in Eq. 8 with a standard TD objective:

$$\min_\theta \quad J_Q(\theta) = \mathcal{R}(\theta; \rho) + \frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}} \left[\left(Q_\theta(\mathbf{s}, \mathbf{a}) - \mathcal{B}^\pi \bar{Q}(\mathbf{s}, \mathbf{a})\right)^2\right] \tag{10}$$

Practical implementation. Our algorithm, CQL (ReDS) is illustrated in Algorithm 1. A complete implementation of the functions in python-style is provided in Appendix C.1. CQL (ReDS) first performs one gradient descent step on ρ_ψ using Eq. 9 (Lines 3), then performs one gradient step on the Q-function using the objective in Eq. 10 (Line 4), then performs one gradient descent step on the policy π_ϕ (Line 5), and repeats.

Algorithm 1 CQL (ReDS) pseudo-code

1: Initialize Q-function, Q_θ, a policy, π_ϕ, and distribution ρ_ψ
2: **for** step t in $\{1, \ldots, \text{N}\}$ **do**
3: Update ρ_ψ with the objective in Eq. 9:
$\psi_t := \psi_{t-1} + \eta_\rho \mathop{\mathbb{E}}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}[\log \rho_\psi(\mathbf{a}|\mathbf{s}) \cdot \exp(-A_\theta(\mathbf{s}, \mathbf{a})/\tau)]$
4: Train the Q-function using $J_Q(\theta)$ in Eq. 10:
$\theta_t := \theta_{t-1} - \eta_Q \nabla_\theta J_Q(\theta)$
5: Improve policy π_ϕ with SAC-style update:
$\phi_t := \phi_{t-1} + \eta_\pi \mathbb{E}_{\mathbf{s}\sim\mathcal{D}, \mathbf{a}\sim\pi_\phi(\cdot|\mathbf{s})}[Q_\theta(\mathbf{s}, \mathbf{a}) - \log \pi_\phi(\mathbf{a}|\mathbf{s})]$

4.2 Theoretical Analysis of CQL (ReDS)

Next, we analyze CQL (ReDS), showing how learning using the regularizer in Eq. 8 modifies the Q-values, providing justification behind our choice of the distribution ρ in the previous section.

Lemma 4.1 (Per-state modification of Q-values.)**.** *Let $g(\mathbf{a}|\mathbf{s})$ be a shorthand representation for $g(\mathbf{a}|\mathbf{s}) = g\left(\tau \cdot \pi(\mathbf{a}|\mathbf{s})\right)$. In the tabular setting, the Q-function obtained after one iteration using the objective in Eq. 10 is given by:*

$$Q_\theta(\mathbf{s}, \mathbf{a}) := \mathcal{B}^\pi \bar{Q}(\mathbf{s}, \mathbf{a}) - \alpha \frac{\pi(\mathbf{a}|\mathbf{s}) + \pi_\beta(\mathbf{a}|\mathbf{s}) g(\mathbf{a}|\mathbf{s}) - 2\pi_\beta(\mathbf{a}|\mathbf{s})}{2\pi_\beta(\mathbf{a}|\mathbf{s})} \tag{11}$$

where $\mathcal{B}^\pi \bar{Q}(\mathbf{s}, \mathbf{a})$ is the Bellman backup operator applied to a delayed target Q-network.

Eq. 11 illustrates why the modified regularizer in Eq. 8 leads to a "soft" support constraint whose strength is modulated per-state. Since g is a monotonically decreasing function of π, for state-action pairs where $\pi(\mathbf{a}|\mathbf{s})$ has high values, $g(\mathbf{a}|\mathbf{s})$ is low and therefore the Q-value $Q(\mathbf{s}, \mathbf{a})$ for such state-action pairs are underestimated less. Vice versa, for state-action pairs where $\pi(\mathbf{a}|\mathbf{s})$ attains low values, $g(\mathbf{a}|\mathbf{s})$ is high to counter-acts the low $\pi(\mathbf{a}|\mathbf{s})$ values. Also, since $\pi_\beta(\mathbf{a}|\mathbf{s})$ appears in the denominator, for out-of-support actions, where $\pi_\beta(\mathbf{a}|\mathbf{s}) = 0$, $\pi(\mathbf{a}|\mathbf{s})$ must also assign 0 probability to the actions for the Q values to be well defined. An illustration of this idea is shown in Figure 3. We can use this insight to further derive the objective optimized by ReDS.

Lemma 4.2 (CQL (ReDS) objective.)**.** *Assume that for all policies $\pi \in \Pi, \forall(\mathbf{s}, \mathbf{a}), \pi(\mathbf{a}|\mathbf{s}) > 0$. Then, CQL (ReDS) solves the following optimization problem:*

$$\max_{\pi\in\Pi} \; \widehat{J}(\pi) - \frac{\alpha}{2(1-\gamma)} \mathbb{E}_{\mathbf{s}\sim\widehat{d}^\pi} \left[D(\pi, \pi_\beta)(\mathbf{s}) + \cdot \mathop{\mathbb{E}}_{\mathbf{a}\sim\pi(\cdot|\mathbf{s})} \left[g\left(\tau \cdot \pi(\mathbf{a}|\mathbf{s})\right) \mathbb{I}\left\{\pi_\beta(\mathbf{a}|\mathbf{s}) > 0\right\}\right]\right]. \tag{12}$$

$\widehat{J}(\pi)$ corresponds to the empirical return of the learned policy, i.e., the return of the policy under the learned Q-function. The objective in Lemma 4.2 can be intuitively interpreted as follows. The first term, $D(\pi, \pi_\beta)(\mathbf{s})$, is a standard distributional constraint, also present in naïve CQL, and it aims to penalize the learned policy π if it deviates too far away from π_β. ReDS adds an additional second

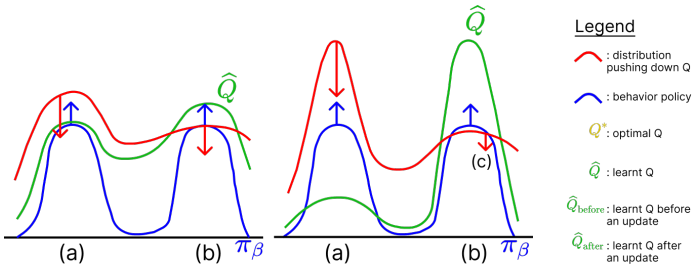

Figure 3: **An illustration of how ReDS modulates the strength of the distributional constraint per state. (i) Left:** let (a) and (b) denote the two modes of the behavior policy $\pi_\beta(\cdot|\mathbf{s})$ at a state $\mathbf{s}$. Consider in this setting that the learned Q-values at these modes are also roughly similar. In this case, the reweighted distribution π^{re} (in red) puts a roughly identical weight on pushing down Q-values on actions appearing in each of these modes. The resulting outcome is that the ReDS constraint behaves similarly to a distributional constraint. **(ii) Right:** In this case, let's consider a setting where actions appearing in one mode of the behavior policy attain very low Q-values (mode (a)). The re-weighted distribution π^{re} pushes down the actions appearing in the mode with lower Q-values with a larger weight than the mode with higher Q-values. This counteracts the effect of the push-up term in CQL that attempts to push up the Q-values for all actions under the behavior policy more strongly in region (a). In effect, this ensures that the Q-value for any action in the high-density regions of the behavior policy is not pushed up blindly, but the Q-values for only good, in-support actions (b) is pushed up. The ReDS constraint, therefore allows the learned policy to selectively pick out the better mode in this case.

term that effectively encourages π to be "sharp" within the support of the behavior policy, enabling it to put its mass on actions that lead to a high $\widehat{J}(\pi)$.

This second term allows us control the strength of the distributional constraint per state: at states where the support of the policy is narrow, i.e., the volume of actions such that $\pi_\beta(\mathbf{a}|\mathbf{s}) \geq \varepsilon$ is small, the penalty in Equation 12 reverts to a standard distributional constraint by penalizing divergence from the behavioral policy via $D(\pi, \pi_\beta)(\mathbf{s})$ as the second term is non-existent. At states where the policy π_β is broad, the second term counteracts the effect of the distributional constraint within the support of the behavior policy, by π to concentrate its density on only good actions within the support of π_β with the same multiplier α. Thus even when we need to set α to be large in order to stay close to $\pi_\beta(\cdot|\mathbf{s})$ at certain states (e.g., in the narrow hallways in the didactic example in Sec. 3.1), $D(\pi, \pi_\beta)(\mathbf{s})$ is not heavily constrained at other states.

5 Experimental Evaluation

The goal of our experiments is to understand how CQL (ReDS) compares to distributional constraint methods when learning from heteroskedastic offline datasets. In order to perform our experiments, we construct new heteroskedastic datasets that pose challenges representative of what we would expect to see in real-world problems. We first introduce tasks and heteroskedastic datasets that we evaluate on, and then present our results compared to prior state-of-the-art methods. We also evaluate ReDS on some of the standard D4RL (Fu et al., 2020) datasets which are not heteroskedastic in and find that the addition of ReDS, as expected, does not help, or hurt on those tasks.

5.1 Comparison on D4RL Benchmark

Our motivation for studying heteroskedastic datasets is that heteroskedasticity likely exists in real-world domains such as driving, manipulation where datasets are collected by multiple policies that agree and disagree at different states. While standard benchmarks (D4RL (Fu et al., 2020) and RLUnplugged (Gulcehre et al., 2020)) include offline datasets generated by mixture policies, e.g. the "medium-expert" generated by two policies with different performance, these policies are themselves trained via RL methods such as SAC, that constrain the entropy of the action distribution at each state to be uniform. To measure their heteroskedasticity, we utilize a tractable approximation to C^{π}_{diff} used in our theoretical analysis: the standard deviation in the value of $D(\pi, \pi_\beta)(\mathbf{s})$ across states in the dataset, using a fixed policy π obtained by running CQL. We did not use C^{π}_{diff} directly, as it requires estimating the state density of the data distribution, which is challenging in continuous spaces. Observe in Table 4a that the standard deviation is relatively low for the D4RL antmaze datasets. This corroborates our intuition that D4RL datasets are significantly less heteroskedastic.

Dataset	std	max
noisy (Ours)	**18**	**253**
biased (Ours)	**9**	**31**
diverse (D4RL)	2	11
play (D4RL)	2	13

(a) the new antmaze datasets (Ours) are significantly more heteroskedastic compared to the standard D4RL datasets. We measure heteroskedasticity using the std and max of $D(\pi, \pi_\beta)(\mathbf{s})$ across states in the offline dataset.

Task & Dataset	EDAC	BEAR	CQL	IQL	Ours
medium-noisy	0	0	55	44	**73**
medium-biased	0	0	**73**	48	**74**
large-noisy	0	0	42	39	**53**
large-biased	0	0	**50**	41	45

(b) CQL (ReDS) (Ours) outperforms prior offline RL methods including state-of-the-art methods (IQL) , and prior support constraint methods(BEAR, EDAC) on three out of four scenarios when learning from heteroskedastic datasets in the antmaze task. The improvement over prior methods is larger when learning from the `noisy` datasets, which are more heteroskedactic, as shown in Table 4a, compared to the `biased` datasets.

Table 1: Our method out-performs prior methods in the antmaze navigation task when learning from heteroskedastic datasets, indicating the importance of support constraints relative to distributional constraints.

Dataset	BC	10%BC	DT	AWAC	Onestep RL	TD3+BC	COMBO	CQL	IQL	(Ours)
halfcheetah-medium-replay	36.6	40.6	36.6	40.5	38.1	44.6	55.1	45.5	44.2	52.3
hopper-medium-replay	18.1	75.9	82.7	37.2	97.5	60.9	89.5	95.0	94.7	101.5
walker2d-medium-replay	26.0	62.5	66.6	27.0	49.5	81.8	56.0	77.2	73.9	85.0
halfcheetah-medium-expert	55.2	92.9	86.8	42.8	93.4	90.7	90.0	91.6	86.7	89.5
hopper-medium-expert	52.5	110.9	107.6	55.8	103.3	98.0	111.1	105.4	91.5	110.0
walker2d-medium-expert	107.5	109.0	108.1	74.5	113.0	110.1	103.3	108.8	109.6	112.0
locomotion total	295.9	491.8	488.4	277.8	494.8	486.1	505	523.5	500.6	550.3

Table 2: Performance comparison with recent offline RL algorithms on the D4RL benchmark.

5.2 Comparisons on Constructed Heteroskedastic datasets

Heteroskedastic datasets. To stress-test our method and prior distributional constraint approaches, we collect new datasets for the medium and large mazes used in the antmaze navigation tasks from D4RL: `noisy` datasets, where the behavior policy action variance differs in different regions of the maze, representative of user variability in navigation, and `biased` datasets, where the behavior policy admits a systematic bias towards certain behaviors in different regions of the maze, representative of bias towards certain routes in navigation problems. Table 4a illustrates that these datasets are significantly more heteroskedastic compared to the D4RL datasets.

Using these more heteroskedastic datasets, we compare CQL (ReDS) with CQL and IQL (Kostrikov et al., 2021a), which are recent popular methods, and two prior methods, BEAR (Kumar et al., 2019) and EDAC (An et al., 2021), that also enforce support constraints. For each algorithm, including ours, we utilize hyperparameters directly from the counterpart tasks in D4RL. Due to lack of an effective method for offline policy selection (see Fu et al. (2021)), we utilize oracle checkpoint selection for every method. We compute the mean and standard deviation of the performance across 3 seeds. Table 4b illustrates that the largest gap between CQL (ReDS) and prior methods appears on the `noisy` datasets, which are particularly more heteroskedastic as measured in Table 4a.

We also compare CQL (ReDS) with recent offline RL algorithms on D4RL, including DT (Chen et al., 2021), AWAC (Nair et al., 2020a), onestep RL (Brandfonbrener et al., 2021), TD3+BC (Fujimoto & Gu, 2021b) and COMBO (Yu et al., 2021). Table 2 shows that CQL (ReDS) obtains similar performance as existing distributional constraint methods and outperform BC-based baselines. This is expected given that the D4RL datasets exhibit significantly smaller degree of heteroscedasticity, as previously explained. Also, a large fraction of the datasets is trajectories with high returns. BC using the top 10% trajectories with the highest episode returns already has strong performance (10%BC).

The previous results compares CQL (ReDS) to baselines in tasks where the MDP states are low-dimensional vectors. This section compares CQL (ReDS) to baselines in vision-based tasks.

Task	CQL	CQL (ReDS)	std $D(\pi, \pi_\beta)(\mathbf{s})$	max $D(\pi, \pi_\beta)(\mathbf{s})$
Pick & Place	6.5 ± 0.4	$\mathbf{15.1 \pm 0.4}$	48.7	307.4
Bin Sort (Easy)	$\mathbf{31.2 \pm 0.3}$	$\mathbf{31.4 \pm 0.3}$	7.9	81.6
Bin Sort (Hard)	6.1 ± 0.2	$\mathbf{23.1 \pm 0.7}$	59.6	988.3

Table 3: **CQL (ReDS) vs CQL** on the manipulation tasks. CQL (ReDS) outperforms CQL significantly when learning from more heteroskedastic datasets, as measured by our estimates of C^{π}_{diff}: the standard-deviation and the maximum of $D(\pi, \pi_\beta)(\mathbf{s})$ across states.

Visual robotic manipulation. We consider two types of manipulation tasks. In the "Pick & Place" task, the algorithm controls a WidowX robot to grasp an object and place it into a tray located at a test location, directly from raw $128 \times 128 \times 3$ images and sparse 0/1 reward signal. The dataset consists of behavior from suboptimal grasping and placing policies, and the positions of the tray in the offline dataset very rarely match the target test location. The placing policies exhibit significant variability, implying that these datasets are heteroskedastic under our definition. We also consider "Bin Sort" task (see Figure 4), where a WidowX robot is controlled to sort two objects into two separate bins. Here heteroskedacity is introduced by when sorting the objects into the desirable bins. Similar to the Pick & Place Domain, the placing policy exhibits significant variability, and shows an the object being placed in the incorrect bin (e.g., recyclable trash thrown into the non-recyclable bin). However, the grasping policy is more expert-like grasping the object with low variability. More details can be found in Appendix D.

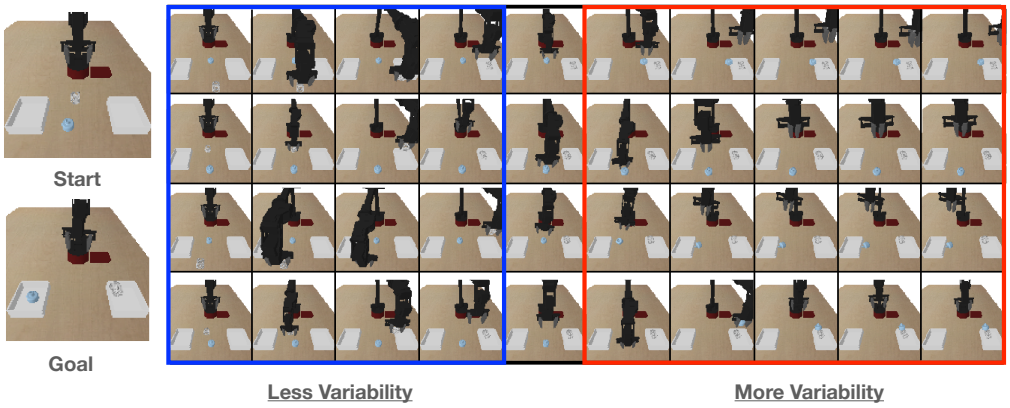

Figure 4: **Examples trajectories from the heteroskedastic bin-sort data.** In this visual manipulation task, an offline RL algorithm must sort the objects in front of it into two bins. The offline dataset provided exhibits non-uniform coverage at different states. In the first half of the trajectory, the states exhibit a more narrow action distribution. However, the second half of the trajectory has more uniform action distribution.

Table 3 presents the results on these tasks. We utilize oracle policy selection analogous to the antmaze experiments from Table 4b. Table 3 shows that CQL (ReDS) outperforms CQL attaining a success rate of about 15.1% for the visual pick and place task, whereas CQL only attains 6.5% success. While these performance numbers might appear low in an absolute sense, note that both CQL and ReDS do improve over the behavior policy, which only attains a success rate of 4%. Thus offline RL does work on this task, and utilizing ReDS in conjunction with the standard distributional constraint in CQL does result in a boost in performance with this heteroskedastic dataset. For the "Bin Sorting", our method outperforms CQL by **3.5x** when learning from more heteroskedastic datasets. This indicates that the effectiveness of our method in settings with higher heteroskedasticity.

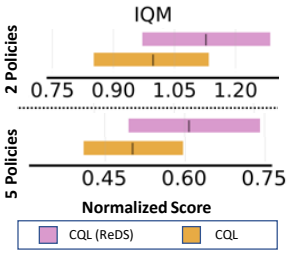

Figure 5: **Performance of CQL (ReDS) vs CQL** on the IQM normalized score and the mean normalized score over ten Atari games, for the case of **two** (top) and **five** (bottom) policies.

Atari games. We collect data on 10 Atari games from multiple policies that behave differently at certain states, while having similar actions otherwise. We consider a case of **two** such policies, and a harder scenario of **five**. We evaluate the performance of CQL (ReDS) on the Atari games using the evaluation metrics from prior works (Agarwal et al., 2020; Kumar et al., 2021), denoted IQM.

Figure 5 shows that in both testing scenarios: with the mixture of two policies (top figure) and the mixture of five policies (bottom figure), CQL (ReDS) outperforms CQL in aggregate. **To summarize**, our results indicate that incorporating CQL (ReDS) outperform common distributional constraints algorithms when learning from heteroskedastic datasets.

6 Related Work

Offline Q-learning methods utilize mechanisms to prevent backing up unseen actions (Levine et al., 2020), by applying an explicit behavior constraint that forces the learned policy to be "close" to the

behavior policy (Jaques et al., 2019; Wu et al., 2019; Peng et al., 2019; Siegel et al., 2020; Wu et al., 2019; Kumar et al., 2019; Kostrikov et al., 2021b;a; Wang et al., 2020; Fujimoto & Gu, 2021a), or by learning a conservative value function (Kumar et al., 2020; Xie et al., 2021; Nachum et al., 2019; Yu et al., 2021; 2020; Rezaeifar et al., 2021; Jin et al., 2020; Wu et al., 2019). Most of these offline RL methods utilize a distributional constraint, explicit (e.g., TD3+BC (Fujimoto & Gu, 2021a)) or implicit (e.g., CQL (Kumar et al., 2020)), and our empirical analysis of representative algorithms from either family indicates that these methods struggle with heteroskedastic data, especially those methods that use an explicit constraint. Model-based methods (Kidambi et al., 2020; Yu et al., 2020; Argenson & Dulac-Arnold, 2020; Swazinna et al., 2020; Rafailov et al., 2021; Lee et al., 2021; Yu et al., 2021) train value functions using dynamics models, which is orthogonal to our method.

Some prior works have also made a case for utilizing support constraints instead of distribution constraints, often via didactic examples (Kumar et al., 2019; Kumar, 2019; Levine et al., 2020), and devised algorithms that impose support constraints in theory, by utilizing the maximum mean discrepancy metric (Kumar et al., 2019) or an asymmetric f-divergences (Wu et al., 2019) for the policy constraint (Wu et al., 2019). Empirical results on D4RL (Fu et al., 2020) and the analysis by Wu et al. (2019) suggest that support constraints are not needed, as strong distribution constraint algorithms often have strong performance. As we discussed in Sections 3.2 (Theorem 3.1 indicates that this distributional constraints may not fail when C^{π}_{diff} is small, *provided these algorithms are well-tuned*.) and 4, these benchmark datasets are not heteroskedastic, as they are collected from policies that are equally wide at all states and centered on good actions (e.g., Antmaze domains in (Fu et al., 2020), control suite tasks in Gulcehre et al. (2020)) and hence, do not need to modulate the distributional constraint strength. To benchmark with heteroskedastic data, we developed some novel tasks which may be of independent interest beyond this work, and find that our method ReDS can work well in these cases.

7 Discussion

In this work, we studied the behavior of distributional constraint offline RL algorithms when learning from heteroskedastic datasets, a property we are likely encounter in the real world. Naïve distributional constraint algorithms can be highly ineffective in such settings both in theory and practice, as they fail to modulate the constraint strength per-state. We propose ReDS, a method to convert distributional constraints into support-based constraints via reweighting, and validate it in CQL. A limitation of ReDS is that it requires estimating the distribution ρ_ψ to enforce a support constraint, which brings about its some additional compute overhead. Devising approaches for enforcing support constraints that do not require extra machinery is an interesting direction for future work. Understanding if support constraints are less sensitive to hyperparameters or are more amenable to cross-validation procedures is also interesting. Extending ReDS to other Q-learning based, and even model-based offline RL algorithms is also an interesting direction for future work.

Acknowledgements

We would like to thank anonymous reviewers and others from the RAIL lab at UC Berkeley and IRIS lab at Stanford for their comments and support. We thank Rishabh Agarwal for informative comments and suggestions. AK is supported by the Apple Scholars in AI/ML PhD Fellowship. This research is supported by the Office of Naval Research, Army Research Labs and the National Science Foundation. We thank Google Cloud for donating TPU resources via the TRC program that enabled us to timely obtain results in this paper.

REFERENCES

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 22–31. JMLR. org, 2017.

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2020.

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-Based Offline Reinforcement Learning with Diversified Q-Ensemble. *arXiv e-prints*, art. arXiv:2110.01548, October 2021.

Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning. *arXiv preprint arXiv:2008.05556*, 2020.

David Brandfonbrener, William F. Whitney, Rajesh Ranganath, and Joan Bruna. Offline RL without off-policy evaluation. *CoRR*, abs/2106.08909, 2021. URL https://arxiv.org/abs/2106.08909.

Kaidi Cao, Yining Chen, Junwei Lu, Nikos Arechiga, Adrien Gaidon, and Tengyu Ma. Heteroskedastic and imbalanced deep learning with adaptive regularization. *arXiv preprint arXiv:2006.15766*, 2020.

Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G. Bellemare. Dopamine: A Research Framework for Deep Reinforcement Learning. 2018. URL http://arxiv.org/abs/1812.06110.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*, 2021.

Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R Qi, Yin Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9710–9719, 2021.

Justin Fu, Aviral Kumar, Matthew Soh, and Sergey Levine. Diagnosing bottlenecks in deep Q-learning algorithms. *arXiv preprint arXiv:1902.10250*, 2019.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Justin Fu, Mohammad Norouzi, Ofir Nachum, George Tucker, ziyu wang, Alexander Novikov, Mengjiao Yang, Michael R Zhang, Yutian Chen, Aviral Kumar, Cosmin Paduraru, Sergey Levine, and Thomas Paine. Benchmarks for deep off-policy evaluation. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=kWSeGEeHvF8.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *arXiv preprint arXiv:2106.06860*, 2021a.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *CoRR*, abs/2106.06860, 2021b. URL https://arxiv.org/abs/2106.06860.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. *arXiv preprint arXiv:1812.02900*, 2018a.

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

Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl. In *International Conference on Machine Learning*, pp. 3682–3691. PMLR, 2021.

Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Tom Le Paine, Sergio Gómez Colmenarejo, Konrad Zolna, Rishabh Agarwal, Josh Merel, Daniel Mankowitz, Cosmin Paduraru, et al. Rl unplugged: Benchmarks for offline reinforcement learning. 2020.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. Technical report, 2018.

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline rl? *arXiv preprint arXiv:2012.15085*, 2020.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pp. 651–673, 2018.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *arXiv preprint arXiv:2005.05951*, 2020.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021a.

Ilya Kostrikov, Jonathan Tompson, Rob Fergus, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. *arXiv preprint arXiv:2103.08050*, 2021b.

Aviral Kumar. Data-driven deep reinforcement learning. https://bair.berkeley.edu/blog/2019/12/05/bear/, 2019. BAIR Blog.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, pp. 11761–11771, 2019.

Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. *arXiv e-prints*, art. arXiv:1906.00949, June 2019.

Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. 2019. URL http://arxiv.org/abs/1906.00949.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=O9bnihsFfXU.

Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. Should i run offline reinforcement learning or behavioral cloning? In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=AP1MKT37rJ.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pp. 45–73. Springer, 2012.

Romain Laroche, Paul Trichelair, and Rémi Tachet des Combes. Safe policy improvement with baseline bootstrapping. *arXiv preprint arXiv:1712.06924*, 2017.

Byung-Jun Lee, Jongmin Lee, and Kee-Eung Kim. Representation balancing offline model-based reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=QpNz8r_Ri2Y.

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

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

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

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Fei-Fei Li, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=JrsfBJtDFdI.

Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019.

Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. *CoRR*, abs/2006.09359, 2020a. URL https://arxiv.org/abs/2006.09359.

Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020b.

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

Rafael Rafailov, Tianhe Yu, A. Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. *Learning for Decision Making and Control (L4DC)*, 2021.

Paria Rashidinejad, Banghua Zhu, Cong Ma, Jiantao Jiao, and Stuart Russell. Bridging offline reinforcement learning and imitation learning: A tale of pessimism. *arXiv preprint arXiv:2103.12021*, 2021.

Shideh Rezaeifar, Robert Dadashi, Nino Vieillard, Léonard Hussenot, Olivier Bachem, Olivier Pietquin, and Matthieu Geist. Offline reinforcement learning as anti-exploration. *arXiv preprint arXiv:2106.06431*, 2021.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897, 2015.

Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

Avi Singh, Albert Yu, Jonathan Yang, Jesse Zhang, Aviral Kumar, and Sergey Levine. Cog: Connecting new skills to past experience with offline reinforcement learning. *arXiv preprint arXiv:2010.14500*, 2020.

Phillip Swazinna, Steffen Udluft, and Thomas Runkler. Overcoming model bias for robust offline deep reinforcement learning. *arXiv preprint arXiv:2008.05533*, 2020.

Ziyu Wang, Alexander Novikov, Konrad Żołna, Jost Tobias Springenberg, Scott Reed, Bobak Shahriari, Noah Siegel, Josh Merel, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. *arXiv preprint arXiv:2006.15134*, 2020.

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

Tengyang Xie, Ching-An Cheng, Nan Jiang, Paul Mineiro, and Alekh Agarwal. Bellman-consistent pessimism for offline reinforcement learning. *Advances in neural information processing systems*, 34, 2021.

Denis Yarats, David Brandfonbrener, Hao Liu, Michael Laskin, Pieter Abbeel, Alessandro Lazaric, and Lerrel Pinto. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. *arXiv preprint arXiv:2201.13425*, 2022.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *arXiv preprint arXiv:2005.13239*, 2020.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *arXiv preprint arXiv:2102.08363*, 2021.

Appendices

A Details of the Didactic Navigation Example from Section 3.1

Figure 6: **Results of running AWR on the gridworld maze for a variety of temperature values** annotated with state-occupancy values. Observe that AWR is unable to reach the goal across a wide range of temperatures for the AWR hyperparameter. Even though for some of these hyperparameters, such as $\tau = 0.1$, it is able to traverse quickly into the third region, it does spend a larger fraction of its state visitation in the narrow hallways in this case (e.g., the final narrow hallway it is able to reach to) indicating that it gets stuck. Increasing the temperature to $\tau = 1.0$ does not solve it either, since now it does not even reach this hallway with as high of an occupancy.

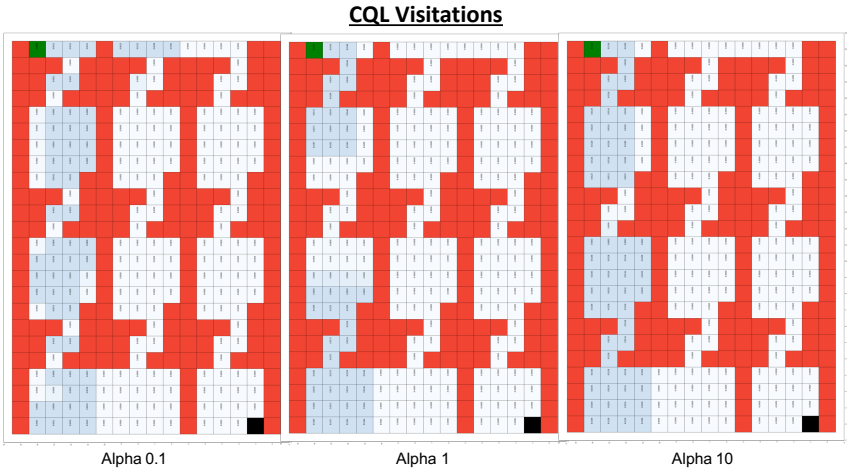

Figure 7: **Results of running CQL on the gridworld maze for a variety of temperature values.** Observe that CQL is unable to reach the goal and gets stuck for all the α values we studied.

In this section, we present some details regarding the navigation example we considered in Section 3.1. To create this example, we modified the gridworld code from Fu et al. (2019) (code taken from: https://github.com/justinjfu/diagnosing_qlearning) to create the corresponding gridworld maze. We utilize a 24×16 gridworld as shown in the figures below, larger than the 8×8 or 16×16 gridworlds studied in this repository in the past. We utilize a smooth representation of the observation space. This is constructed by first sampling random Gaussian feature vectors from $\mathbb{R}^{50}$ for each grid cell (each grid cell is a state). Smoothing of these features vectors is then done locally, following the protocol for "grid-*-smoothobs" in Fu et al. (2019). This observation type presents a challenge for Q-learning algorithms that may often generalize incorrectly. This is because aliasing occurs with the predictions of nearby states that exhibit very different dynamics

but not so different observation vectors. The agents gets a sparse binary 0/1 reward: +1 is attained only when the agent reaches the goal (marked in black) from the start location (marked in green).

The behavior policy in each part of the maze is based on a mixture of different policies. In the wider rooms of the maze, one of the policies is a uniform policy, that uniformly chooses every action at a state. The second policy is a biased policy, that drives the agent away from the goal. In the narrow hallways, the behavior policy deterministically drives the agent towards the goal. For wide rooms, in contrast, a bias exists for actions taken in the direction away from the goal. This bias was set to be 0.8. This means that for rooms where you need to exit the wide passage by going down, the action of going up was selected 80% of the time by the behavior policy and each of the other action was randomly sampled with 20% probability. The opposite decision was taken for rooms where the goal was towards the top of the wide passage where 80% of the time the down action was selected.

Result visualizations. We now present some visualizations of the policies learned by various methods: AWR, CQL and CQL (ReDS). Since the reward values are binary and sparse, return curves for AWR and CQL are not as informative, since they attain a 0 return in any episode, even if they make some progress. Therefore, we present the results in the form of state-visitation density plots under rollouts from the learned policy. Observe in Figures 6 and 7 that neither AWR or CQL are able to actually successfully traverse the maze, and get stuck in it. Note that τ and α control the strength of the distributional constraint in the methods AWR and CQL respsectively. Varying the temperature hyperparameter τ for AWR and α for CQL does modify the density in the narrow hallways, but utilizing too small of a parameter leads the agent to more frequent crashes in the narrow hallways. This makes the agent spend a higher fraction of its visitation in one such narrow region, whereas a higher τ or α does not even reach the third narrow hallway with a high-enough visitation.

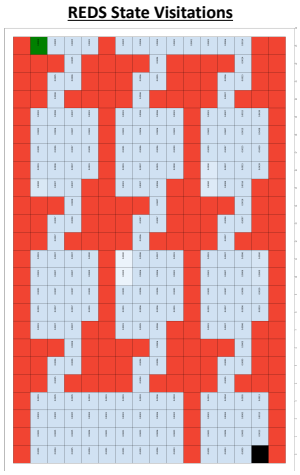

Figure 8: **Results of running CQL (ReDS) on the gridworld.** Note that CQL (ReDS) does actually succeed at solving the task.

On the other hand, the method we propose in this paper, ReDS, when combined with CQL is able to successfully traverse this maze, as shown in Figure 8.

Analysis. These experiments are consistent with our expectations. When distributional constraints are faced with highly heteroskedastic action distributions at subsequent states, they failed to perform good actions at these consecutive states. A strong distributional constraint leads to the agent being too close to the behavior policy, whereas a weaker constraint fails to identify good actions at states where the behavior policy is narrow.

A.1 Why and When Do Distributional Constraints Fail in scenarios with Heteroskedastic Data?

In this section, we shall discuss why distributional constraints are especially worse than support constraints in scenarios with heteroskedastic data. Let's first consider a simple single-state bandit problem. In this simple one-state problem, we must find a single optimal action by using offline data. At first, it might appear that when faced with a wide behavior policy, distributional constraints

would clearly fail since they would not deviate far away from the behavior policy. However, note that by carefully choosing the strength of the distributional constraint, we can, in principle, control the strength of the distributional constraint quite effectively. To see this concretely, consider applying CQL (Equation 2) to a single-state bandit problem. The Q-values on actions not observed in the training dataset will be clearly pushed down to $-\infty$. In addition, choosing a non-zero α will allow us to precisely control how close the action taken by the learned policy is to the best in-support action vs how close the learned policy is to the behavior policy. Therefore, for an optimally chosen value of α, distributional constraint methods should already work well in the single-state setting.

However, the precise challenge with distributional constraints arises in the multi-state setting, where finding a single value of α that can work well at all states might be can be exceedingly challenging in practice. This is the setting with heteroskedastic data that our paper considers, including in our didactic example above. In such a setting, a strong distributional constraint is able to take a desirable action when the behavior policy is narrow (for example in the narrow room). However, the action distribution has wider support in the wider room and the strong distributional constraint would not take a desirable action in this setting due to the high entropy of the behavioral distribution in this setting. In contrast, with a weaker distributional constraint, a bad out-of-distribution action may be taken where the behavior policy is narrow which can lead to instability in the policy. Note that these challenges are not, however, present in the single-state scenario. Our method CQL (ReDS) can tackle the problem in this setting by modulating the strength of the constraint per state. This allows the agent to stay close to the behavior distribution in the narrow room as well as learning to output the most desirable in-support action in the wider room.

A.2 Heteroskedastic Data vs Exploratory Data

There is a distinction between exploratory data (Yarats et al., 2022) and heteroskedasticity. In the context of offline RL, exploratory data typically refers to being able to cover **all** possible state-action pairs, as uniformly as possible. On the other hand, heteroskedasticity is used to refer to **disproportionate** coverage: when the action distribution in the offline data is more uniform at certain states, but very narrow at others. Thus, a heteroskedastic dataset does not have high coverage in the sense of observing all possible state-action pairs.

Mathematically, while the notion of coverage is typically quantified by the concentrability coefficient which denotes the **worst case** density ratio between the distribution of the training dataset and the distribution of the learned policy, as shown below:

$$C^{\pi} = \max_{\mathbf{s},\mathbf{a}} \; \frac{d^{\pi}(\mathbf{s},\mathbf{a})}{\mu(\mathbf{s},\mathbf{a})},$$

where $\mu(\mathbf{s},\mathbf{a})$ denotes the state-action distribution induced by the dataset, while $d^{\pi}(\mathbf{s},\mathbf{a})$ denotes the state-action visitation distribution of a policy π. $\max_{\pi} C^{\pi}$ is generally expected to be smaller when $\mu(\mathbf{s},\mathbf{a})$ is more uniform. In contrast, we quantify the notion of heteroskedasticity by the variation in density ratios (in this case, action density ratios, $\frac{\pi(\mathbf{a}|\mathbf{s})}{\pi_{\beta}(\mathbf{a}|\mathbf{s})}$) across the state space in Equation 4 (reproduced below for convenience).

$$C^{\pi}_{\text{diff}} = \mathop{\mathbb{E}}_{\mathbf{s}_1,\mathbf{s}_2\sim d^{\pi}} \left[\left(\sqrt{\frac{D(\pi,\pi_{\beta})(\mathbf{s}_1)}{\mu(\mathbf{s}_1)}} - \sqrt{\frac{D(\pi,\pi_{\beta})(\mathbf{s}_2)}{\mu(\mathbf{s}_2)}} \right)^2 \right].$$

Even if the dataset has high coverage (i.e., it has a low concentrability coefficient, C^{π}), it can attain a higher C^{π}_{diff}, because of higher variation in the density ratios across state spaces.

Out of all methods considered in Yarats et al. (2022), we note that the methods classified as "Data" (APT, ProtoRL) attempt to maximize state coverage, the methods classified as "Knowledge" (ICM, Disagreement, RND) attempt to find surprising state-action pairs and visit them, and the methods classified as "Competence" (SMM, DIAYN, APS) that maximize the mutual information between the state distribution attained by the learned policy and the learned skills also implicitly optimize for state coverage. In effect, all of these methods aim to visit all state-action pairs (or all states), which is akin to the standard definition of concentrability, whereas heteroskedastic data corresponds distinctly to disproportionate action coverage at different states.

B PROOFS

In this appendix, we will provide proofs for the various theoretical results in the main paper: Theorem 3.1. We will first discuss some preliminaries and notation, then present the proof for Theorem 3.1, and finally the remaining results.

B.1 NOTATION AND PRELIMINARIES

Let $\pi_\beta(\mathbf{a}|\mathbf{s})$ denote the behavior policy. Note that the dataset, $\mathcal{D}_i$ is generated from the marginal state-action distribution of π_β, i.e., $\mathcal{D} \sim d^{\pi_\beta}(\mathbf{s})\pi_\beta(\mathbf{a}|\mathbf{s})$. Define $\widehat{d}^\pi$ as the state marginal distribution introduced by π under the empirical MDP defined by the transitions in the dataset. Let $D_{\text{CQL}}(p, q)$ denote the following distance between two distributions $p(\mathbf{x})$ and $q(\mathbf{x})$ with equal support $\mathcal{X}$:

$$D_{\text{CQL}}(p, q) := \sum_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x}) \left(\frac{p(\mathbf{x})}{q(\mathbf{x})} - 1\right).$$

We drop the subsrcipt "CQL" from D_{CQL} for clarity. (Kumar et al., 2020) showed that when optimizing the generic distributional constraint algorithm shown in Equation 1, the resulting policy π^* attains a high probability safe-policy improvement guarantee, i.e., $J(\pi^*) \geq J(\pi_\beta) - \zeta$, where ζ is:

$$\zeta = \mathcal{O}\left(\frac{1}{(1-\gamma)^2}\right) \mathbb{E}_{\mathbf{s}\sim\widehat{d}^{\pi^*}} \left[\sqrt{\frac{D(\pi^*, \pi_\beta)(\mathbf{s}) + 1}{|\mathcal{D}(\mathbf{s})|}}\right] + \frac{\alpha}{1-\gamma} \mathbb{E}_{\mathbf{s}\sim\widehat{d}^{\pi}}[D(\pi^*, \pi_\beta)(\mathbf{s})]. \quad (13)$$

We can further express $|\mathcal{D}(\mathbf{s})| = |\mathcal{D}||\mu(\mathbf{s})|$. The first term in Equation 13 corresponds to the decrease in performance due to sampling error and this term is high when the learned policy π^* visits low density states under the dataset distribution (i.e., $\mu(\mathbf{s})$ is small) and when the divergence from the behavior policy π_β is higher under these states. We will use this safe policy improvement guarantee in our proofs.

B.2 FORMAL RESTATEMENT OF THEOREM 3.1

Theorem B.1 (Formal version of Theorem 3.1)**.** *For any prescribed level of safety ζ, i.e., the learned policy satisfies $J(\pi) - J(\pi_\beta) \geq -\zeta$, the maximum possible policy improvement over choices of α:*

$$\max_\alpha \left[J(\pi_\alpha) - J(\pi_\beta)\right] \leq \zeta^+,$$

where ζ^+ is given by:

$$\zeta^+ := \max_\alpha \quad h^*(\alpha) \cdot \frac{1}{(1-\gamma)^2} \qquad \textit{s.t.} \quad \frac{c_1\sqrt{\log\frac{|\mathcal{S}||\mathcal{A}|}{\delta}}}{(1-\gamma)^2}\frac{\sqrt{C^{\pi_\alpha}_{\textit{diff}}}}{|\mathcal{D}|} - \zeta \leq \underbrace{\alpha \cdot \left(\frac{\mathbb{E}_{\mathbf{s}\sim\widehat{d}^{\pi_\alpha}}[D(\pi_\alpha, \pi_\beta)(\mathbf{s})]}{1-\gamma}\right)}_{:=g(\alpha)} \quad (14)$$

where h^* is a monotonically decreasing function of α, and $h(0) = const.$

Intuition behind the theorem: We will show in the proof of this theorem that $g(\alpha)$ is a monotonically increasing function of α and $g(0) = 0$. This means that if the value of C^π_{diff} is large for all policies π, the value of $\alpha = g^{-1}\left(\sqrt{C^\pi_{\text{diff}}}c' - \zeta\right)$ (where c' is the constant and subsumes the term containing δ) is also large. Further, note that $h^*(\alpha)$ is decreasing in α. This means that the larger the value of α, the smaller the value of $h^*(\alpha)$, and hence, smaller the value of ζ^+. This means that as C^π_{diff} increases, α needs to take larger values to satisfy the safety constraint, and this reduces the value of maximal improvement, ζ^+.

Conversely, to attain a larger improvement ζ^+, if we choose a smaller α for a problem where C^π_{diff} is large for all policies, then, we must give up on the safety guarantee, and ζ would be large. This means that the learned policy π_α might substantially degrade beyond the behavior policy.

B.3 PROOF OF THEOREM 3.1

In order to prove this result, we we will utilize a basic algebraic inequality mentioned below in Lemma B.1.

Lemma B.1 (Algebraic variation)**.** *Given any N positive real numbers, $x_1, x_2, \cdots, x_N$:*

$$\left(\sum_{i=1}^{N} \sqrt{x_i}\right)^2 \geq \sum_{i=1}^{N} x_i \geq \frac{1}{(N-1)} \sum_{i<j} (\sqrt{x_i} - \sqrt{x_j})^2. \tag{15}$$

Proof. For every x_i, define $y_i = \sqrt{x_i}$. Then, the difference between the two sides in the equation above is given by:

$$\sum_i y_i^2 - \frac{1}{(N-1)} \sum_{i<j} (y_i^2 + y_j^2 - 2y_i y_j) = \sum_i y_i^2 - \frac{N-1}{N-1} y_i^2 + \frac{1}{N-1} \sum_{i<j} 2y_i y_j \tag{16}$$

$$= \frac{1}{N-1} \sum_{i<j} 2y_i y_j \geq 0, \tag{17}$$

where the first step follows by rearranging y_i and y_j, and the final step follows by noting that $y_i \geq 0$ for all i. The other inequality follows trivially by noting that $\sqrt{x_i}$ are positive, and applying the standard formula for sum of squares. □

We will also require a Lemma that allows us to upper bound the performance difference $J(\pi) - J(\pi_\beta)$ in terms of the metric $D_{\text{CQL}}(\pi, \pi_\beta)$ that appears in the safe policy improvement guarantee in Equation 13.

Lemma B.2 (Tight upper bound on policy improvement.)**.** *Assume that the reward function $r(\mathbf{s}, \mathbf{a})$ of the MDP is bounded such that $\forall \mathbf{s}, \mathbf{a}, r(\mathbf{s}, \mathbf{a}) \in [-R_{\max}, R_{\max}]$. For any two policies π and π_β, we have the following:*

$$J(\pi) - J(\pi_\beta) \lesssim \mathcal{O}\left(\frac{1}{(1-\gamma)^2}\right) \cdot \mathbb{E}_{\mathbf{s}\sim d^\pi}\left[D(\pi(\cdot|\mathbf{s}), \pi_\beta(\cdot|\mathbf{s}))\right] \cdot R_{\max}. \tag{18}$$

Proof. The core of the proof of this lemma relies on the fact that for any given function $\nu(\mathbf{x})$ over some space $\mathbf{x}$, we can upper bound, $\Delta_\nu(p, q) := \mathbb{E}_{\mathbf{x}\sim p}[\nu(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim q}[\nu(\mathbf{x})]$ in terms of $D(p, q)$. To show this, we expand this expression:

$$\Delta_\nu(p, q) := \sum_{\mathbf{x}} (p(\mathbf{x}) - q(\mathbf{x})) \cdot \nu(\mathbf{x}) \tag{19}$$

$$= \sum_{\mathbf{x}} q(\mathbf{x}) \cdot \left(\frac{p(\mathbf{x})}{q(\mathbf{x})} - 1\right) \cdot \nu(\mathbf{x}) \tag{20}$$

$$= \sum_{\mathbf{x}: \frac{p(\mathbf{x})}{q(\mathbf{x})} \geq 1, \nu(\mathbf{x}) \geq 0} q(\mathbf{x}) \left(\frac{p(\mathbf{x})}{q(\mathbf{x})} - 1\right) \nu(\mathbf{x}) + \sum_{\mathbf{x}: \frac{p(\mathbf{x})}{q(\mathbf{x})} < 1, \nu(\mathbf{x}) \geq 0} q(\mathbf{x}) \left(\frac{p(\mathbf{x})}{q(\mathbf{x})} - 1\right) \nu(\mathbf{x})$$

$$+ \sum_{\mathbf{x}: \frac{p(\mathbf{x})}{q(\mathbf{x})} \geq 1, \nu(\mathbf{x}) \leq 0} q(\mathbf{x}) \left(\frac{p(\mathbf{x})}{q(\mathbf{x})} - 1\right) \nu(\mathbf{x}) + \sum_{\mathbf{x}: \frac{p(\mathbf{x})}{q(\mathbf{x})} < 1, \nu(\mathbf{x}) \leq 0} q(\mathbf{x}) \left(\frac{p(\mathbf{x})}{q(\mathbf{x})} - 1\right) \nu(\mathbf{x}).$$

Each of the four terms above can be bounded independently as follows: for the first two terms, we multiply by $\frac{p(\mathbf{x})}{q(\mathbf{x})}$, the third term is clearly negative, and the final term is upper bounded by

multiplying by $1 + \frac{p(\mathbf{x})}{q(\mathbf{x})}$:

$$\sum_{\mathbf{x}: \frac{p(\mathbf{x})}{q(\mathbf{x})} \geq 1, \nu(\mathbf{x}) \geq 0} q(\mathbf{x}) \left(\frac{p(\mathbf{x})}{q(\mathbf{x})} - 1 \right) \nu(\mathbf{x}) \leq \sum_{\mathbf{x}: \frac{p(\mathbf{x})}{q(\mathbf{x})} \geq 1, \nu(\mathbf{x}) \geq 0} q(\mathbf{x}) \frac{p(\mathbf{x})}{q(\mathbf{x})} \left(\frac{p(\mathbf{x})}{q(\mathbf{x})} - 1 \right) \nu(\mathbf{x}) \tag{21}$$

$$\sum_{\mathbf{x}: \frac{p(\mathbf{x})}{q(\mathbf{x})} < 1, \nu(\mathbf{x}) \geq 0} q(\mathbf{x}) \left(\frac{p(\mathbf{x})}{q(\mathbf{x})} - 1 \right) \nu(\mathbf{x}) \leq \sum_{\mathbf{x}: \frac{p(\mathbf{x})}{q(\mathbf{x})} \leq 1, \nu(\mathbf{x}) \geq 0} q(\mathbf{x}) \frac{p(\mathbf{x})}{q(\mathbf{x})} \left(\frac{p(\mathbf{x})}{q(\mathbf{x})} - 1 \right) \nu(\mathbf{x}) \tag{22}$$

$$\sum_{\mathbf{x}: \frac{p(\mathbf{x})}{q(\mathbf{x})} \geq 1, \nu(\mathbf{x}) \leq 0} q(\mathbf{x}) \left(\frac{p(\mathbf{x})}{q(\mathbf{x})} - 1 \right) \nu(\mathbf{x}) \leq 0 \tag{23}$$

$$\sum_{\mathbf{x}: \frac{p(\mathbf{x})}{q(\mathbf{x})} < 1, \nu(\mathbf{x}) \leq 0} q(\mathbf{x}) \left(\frac{p(\mathbf{x})}{q(\mathbf{x})} - 1 \right) \nu(\mathbf{x}) \leq \sum_{\mathbf{x}: \frac{p(\mathbf{x})}{q(\mathbf{x})} < 1, \nu(\mathbf{x}) \leq 0} q(\mathbf{x}) \left(\frac{p(\mathbf{x})}{q(\mathbf{x})} - 1 \right) \left(1 + \frac{p(\mathbf{x})}{q(\mathbf{x})} \right) \nu(\mathbf{x}). \tag{24}$$

Finally, for each of these terms, we can now upper bound $\nu(\mathbf{x})$ by its maximum absolute value, $|\nu(\mathbf{x})| \leq \nu_0$, and combine the terms to get the following bound on $\Delta_\nu(p, q)$:

$$\Delta_\nu(p, q) \leq \nu_0 \sum_{\mathbf{x}: \nu(\mathbf{x}) > 0} p(\mathbf{x}) \left(\frac{p(\mathbf{x})}{q(\mathbf{x})} - 1 \right) + 0 + \nu_0 \sum_{\mathbf{x}: \frac{p(\mathbf{x})}{q(\mathbf{x})} < 1, \nu(\mathbf{x}) < 0} \left(\frac{p^2(\mathbf{x})}{q(\mathbf{x})} - q(\mathbf{x}) \right) \tag{25}$$

$$\leq \nu_0 \left[\sum_{\mathbf{x}} \frac{p^2(\mathbf{x})}{q(\mathbf{x})} - 1 \right], \tag{26}$$

where Equation 26 follows from using the fact that for the case when $p(\mathbf{x})/q(\mathbf{x}) > 1$ but $\nu(\mathbf{x}) \leq 0$, $p(\mathbf{x}) \left(\frac{p(\mathbf{x})}{q(\mathbf{x})} - 1 \right) > 0$, and hence it upper bounds the RHS of Equation 23. For the last case, where $\mathbf{x} : \frac{p(\mathbf{x})}{q(\mathbf{x})} < 1, \nu(\mathbf{x}) < 0$, we note that $\sum_{\mathbf{x}: \frac{p(\mathbf{x})}{q(\mathbf{x})} < 1, \nu(\mathbf{x}) < 0} q(\mathbf{x}) \geq \sum_{\mathbf{x}: \frac{p(\mathbf{x})}{q(\mathbf{x})} < 1, \nu(\mathbf{x}) < 0} p(\mathbf{x})$, and hence the upper bound on this term in Equation 26 follows. To complete the argument note that D_{CQL} exactly takes the form obtained in the final equation, and hence:

$$\Delta_\nu(p, q) \leq \nu_0 \cdot D(p, q).$$

We can now use this result to bound the return differences, by using standard results for bounding the performance difference between policies (Achiam et al., 2017; Schulman et al., 2015) in terms of $\frac{1}{(1-\gamma)^2} \times D(\pi, \pi_\beta)$. At the core of these results is a bound on $\mathbb{E}_{\mathbf{a} \sim \pi(\cdot|\mathbf{s})}[f(\mathbf{s}, \mathbf{a})] - \mathbb{E}_{\mathbf{a} \sim \pi_\beta(\cdot|\mathbf{s})}[f(\mathbf{s}, \mathbf{a})]$, and hence the result proven above directly applies. This proves the required result. □

B.4 Additional Technical Lemmas

In this section, we will provide proofs of two technical lemmas, that allow us to conclude the proof of Theorem 3.1. For these lemmas, we will consider a generic optimization problem,

$$\max_{\mathbf{x}} \; f(\mathbf{x}) + \alpha \, g(\mathbf{x}), \tag{27}$$

where $g(\mathbf{x}) > 0$ for any $\mathbf{x}$, and α can only take non-negative values.

Lemma B.3 (Value of $g(\mathbf{x})$ as a function of α)**.** *Let $\mathbf{x}^*_\alpha$ be the value of $\mathbf{x}$ that maximizes Equation 27 for a given fixed value of α. Then the following statements hold:*

1. *For any $\alpha \geq \beta \geq 0$, $g(\mathbf{x}^*_\alpha) \geq g(\mathbf{x}^*_\beta)$.*
2. *For any $\alpha \geq \beta \geq 0$, $\alpha g(\mathbf{x}^*_\alpha) \geq \beta g(\mathbf{x}^*_\beta)$.*

Proof. For any given α, $\mathbf{x}^*_\alpha$ satisfies the following inequality:

$$\forall \mathbf{x}', \quad f(\mathbf{x}^*_\alpha) + \alpha g(\mathbf{x}^*_\alpha) \geq f(\mathbf{x}') + \alpha g(\mathbf{x}'). \tag{28}$$

Using the above relation, we can write down two inequalities relating $\mathbf{x}^*_\alpha$ and $\mathbf{x}^*_\beta$:

$$f(\mathbf{x}^*_\alpha) + \alpha g(\mathbf{x}^*_\alpha) \ \geq f(\mathbf{x}^*_\beta) + \alpha g(\mathbf{x}^*_\beta) \tag{29}$$

$$f(\mathbf{x}^*_\beta) + \beta g(\mathbf{x}^*_\beta) \ \geq f(\mathbf{x}^*_\alpha) + \beta g(\mathbf{x}^*_\alpha) \tag{30}$$

Now, adding the two inequalities above, and cancelling the terms $f(\mathbf{x}^*_\alpha) + f(\mathbf{x}^*_\beta)$ from both sides, we obtain:

$$(\alpha - \beta)\, g(\mathbf{x}^*_\alpha) \geq (\alpha - \beta)\, g(\mathbf{x}^*_\beta). \tag{31}$$

Since $\alpha - \beta$ is non-negative, it is either equal to 0, in which case $g(\mathbf{x}^*_\alpha) = g(\mathbf{x}^*_\beta)$ or it is positive, in which case, $g(\mathbf{x}^*_\alpha) \geq g(\mathbf{x}^*_\beta)$. Combining these two cases, we get the desired result in (1). For proving the second part (2), note that we can write the difference of the two sides as:

$$\alpha g(\mathbf{x}^*_\alpha) - \beta g(\mathbf{x}^*_\beta) = \alpha g(\mathbf{x}^*_\alpha) - \alpha g(\mathbf{x}^*_\beta) + \alpha g(\mathbf{x}^*_\beta) - \beta g(\mathbf{x}^*_\beta) \tag{32}$$

$$= \alpha \left[g(\mathbf{x}^*_\alpha) - g(\mathbf{x}^*_\beta) \right] + (\alpha - \beta) g(\mathbf{x}^*_\beta) \tag{33}$$

$$\geq 0 + 0 = 0, \tag{34}$$

where the last inequality follows from the fact that $g(\mathbf{x}) \geq 0$ and $\alpha > \beta$. □

We will now provide a proof for Theorem 3.1.

Theorem B.2 (Theorem 3.1 restated)**.** *W.h.p. $\geq 1 - \delta$, for any prescribed level of safety ζ, the maximum possible policy improvement over choices of α, $J(\pi_\alpha) - J(\pi_\beta) \leq \zeta^+$, where ζ^+ is given by:*

$$\zeta^+ := \max_{\alpha} \quad h^*(\alpha) \cdot \frac{1}{(1-\gamma)^2} \qquad \textit{s.t.} \ \ \frac{c_1}{(1-\gamma)^2} \sqrt{\frac{C^{\pi_\alpha}_{\textit{diff}}}{|\mathcal{D}|}} - \frac{\alpha}{1-\gamma} \mathbb{E}_{\mathbf{s} \sim \widehat{d}^{\pi_\alpha}} \left[D(\pi_\alpha, \pi_\beta)(\mathbf{s}) \right] \leq \zeta, \tag{35}$$

where h^* is a monotonically decreasing function of α, and $h(0) = \mathcal{O}(1)$.

Proof of Theorem B.2. To prove this theorem, we will apply Lemma B.1 on $x_i = \frac{D(\pi_\alpha(\cdot|\mathbf{s}_i)||\pi_\beta(\cdot|\mathbf{s}_i))}{\mu(\mathbf{s}_i)}$ and combine it with a the safe policy improvement guarantee for behavior regularization methods that admit updates of the form shown in Equation 1.

First, we note by applying Lemma B.1 in its expectation form that:

$$\left(\mathbb{E}_{\mathbf{s} \sim \widehat{d}^{\pi_\alpha}} \left[\sqrt{\frac{D(\pi_\alpha(\cdot|\mathbf{s})||\pi_\beta(\cdot|\mathbf{s}))}{\mu(\mathbf{s})}} \right] \right)^2 \geq \mathbb{E}_{\mathbf{s}_1 \sim \widehat{d}^{\pi_\alpha}, \mathbf{s}_2 \sim \widehat{d}^{\pi_\alpha}} \left[\sqrt{\frac{D(\pi_\alpha(\cdot|\mathbf{s}_1)||\pi_\beta(\cdot|\mathbf{s}_1))}{\mu(\mathbf{s}_1)}} - \sqrt{\frac{D(\pi_\alpha(\cdot|\mathbf{s}_2)||\pi_\beta(\cdot|\mathbf{s}_2))}{\mu(\mathbf{s}_2)}} \right]^2,$$

where the term on the RHS of the above equation corresponds to C^{π}_{diff}.

Now we can plug this into the safe-policy improvement guarantee to obtain the resulting result as follows. Note that the first term in the bound in Equation 13 can be lower bounded using the differential concentrability as discussed above, and therefore, we get the following lower bound on ζ:

$$\zeta \geq \mathcal{O}\left(\frac{1}{(1-\gamma)^2} \right) \sqrt{\frac{C^{\pi_\alpha}_{\text{diff}}}{|\mathcal{D}|}} + \alpha \mathbb{E}_{\mathbf{s} \sim \widehat{d}^{\pi_\alpha}} \left[D(\pi_\alpha, \pi_\beta)(\mathbf{s}) \right], \tag{36}$$

which is exactly the same as the expression for the constraint in Theorem 3.1.

Next we provide an upper bound on the maximal improvement that can be possible, in terms of $D(\pi_\alpha, \pi_\beta)$. For this, we will utilize Lemma B.2, and we can directly upper bound $J(\pi_\alpha) - J(\pi_\beta)$ as follows:

$$J(\pi_\alpha) - J(\pi_\beta) \lesssim \frac{1}{(1-\gamma)^2} \mathbb{E}_{\mathbf{s} \sim \widehat{d}^{\pi_\alpha}} \left[D(\pi_\alpha, \pi_\beta)(\mathbf{s}) \right] \cdot R_{\max}. \tag{37}$$

Finally, we express this upper bound in terms of α.

Now, note that the RHS in Equation 37 depends on $\mathbb{E}_{\mathbf{s} \sim \widehat{d}^{\pi_\alpha}} \left[D(\pi_\alpha, \pi_\beta)(\mathbf{s}) \right]$, which is directly the term that a generic distributional constraint algorithm minimizes (Equation 1). We wish to understand how this term evolves a function of α.

Now we will invoke Lemma B.3 to understand the behavior of the term above when solving the optimization problem in Equation 1. To do so, consider any α, α' and note that $f(\mathbf{x}) = \widehat{J}(\pi)$ and $g(\mathbf{x}) = -\mathbb{E}_{\mathbf{s}\sim\widehat{d}^{\pi}}[D(\pi, \pi_\beta)(\mathbf{s})]$. Now applying Lemma B.3, Part (1), we note that:

$$\mathbb{E}_{\mathbf{s}\sim\widehat{d}^{\pi_\alpha}(\mathbf{s})}[D(\pi_\alpha, \pi_\beta)(\mathbf{s})] \leq \mathbb{E}_{\mathbf{s}\sim\widehat{d}^{\pi_{\alpha'}}(\mathbf{s})}[D(\pi_{\alpha'}, \pi_\beta)(\mathbf{s})], \tag{38}$$

for $\alpha' \leq \alpha$. This means that we can upper bound this quantity by a function $h^*(\alpha)$ that is monotonically decreasing in α.

Therefore, the maximal improvement is upper bounded by: $h^*(\alpha)\mathcal{O}\left(\frac{1}{(1-\gamma)^2}\right)$, which completes the proof of this theorem. □

B.5 Proof of Lemma 4.1

Lemma B.4 ((Lemma 4.1 restated) Per-state modification.)**.** *Let g represents $g\left(\tau \cdot \pi(\cdot|\mathbf{s})\right)$. The Q-function obtained after one TD-learning iteration using the objective in Eq. 10 is:*

$$Q_\theta(\mathbf{s}, \mathbf{a}) := \mathcal{B}^\pi \bar{Q}(\mathbf{s}, \mathbf{a}) - \alpha \frac{\pi(\mathbf{a}|\mathbf{s}) + \pi_\beta(\mathbf{a}|\mathbf{s})g - 2\pi_\beta(\mathbf{a}|\mathbf{s})}{2\pi_\beta(\mathbf{a}|\mathbf{s})} \tag{39}$$

where $\mathcal{B}^\pi \bar{Q}(\mathbf{s}, \mathbf{a})$ is the Bellman backup operator applied to a delayed target Q-network.

Recall from Section 2 that the objective of CQL consists of two terms

$$\min_\theta \; \alpha \underbrace{\left(\mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\pi}\left[Q_\theta(\mathbf{s}, \mathbf{a})\right] - \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}\left[Q_\theta(\mathbf{s}, \mathbf{a})\right]\right)}_{\mathcal{R}(\theta)} + \frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q_\theta(\mathbf{s}, \mathbf{a}) - \mathcal{B}^\pi \bar{Q}(\mathbf{s}, \mathbf{a})\right)^2\right], \tag{40}$$

where $\mathcal{B}^\pi \bar{Q}(\mathbf{s}, \mathbf{a})$ is the Bellman backup operator applied to a delayed target Q-network. In tabular setting, the Q-function obtained after one iteration of TD-learning using the objective function in Eq. 40 is given by:

$$Q_\theta(\mathbf{s}, \mathbf{a}) := \mathcal{B}^\pi \bar{Q}(\mathbf{s}, \mathbf{a}) - \alpha\left[\frac{\pi(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1\right]. \tag{41}$$

In CQL, the result in Eq. 41 is obtained by setting the derivative of the objective in Eq. 40 with respect to the Q-values to 0, and solve for $Q_\theta(\mathbf{s}, \mathbf{a})$ (Kumar et al., 2020). The objective for ReDS is given by:

$$\min_\theta \quad J_Q(\theta) = \mathcal{R}(\theta; \rho) + \frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q_\theta(\mathbf{s}, \mathbf{a}) - \mathcal{B}^\pi \bar{Q}(\mathbf{s}, \mathbf{a})\right)^2\right] \tag{42}$$

Notice that the main difference between the original CQL objective in Eq. 40 and the new objective in Eq. 42 is the distribution with which we push down Q values. The objective in Eq. 40 pushes Q values down under the learned policy π, whereas the objective in Eq. 42 pushes Q values down under a mixture of π and ρ, i.e. $\frac{1}{2}\pi + \frac{1}{2}\rho$. Since ρ is parameterized by a neural network whose input does not contain the Q values, its gradient with respect to the Q values is 0. Additionally, since the mixture $\frac{1}{2}\pi + \frac{1}{2}\rho$ plays the same role that π plays in the objective function in Eq. 40, we therefore can obtain the solution for the Q-values after updating the Q-function using the objective function in Eq. 42 simply by replacing π in Eq. 41 with the mixture. That is, in tabular setting, after updating the Q-function using the objective in Eq.42, the Q-values are:

$$
\begin{aligned}
Q_\theta(\mathbf{s},\mathbf{a}) &= \mathcal{B}^\pi \bar{Q}(\mathbf{s},\mathbf{a}) - \alpha\left[\frac{\frac{1}{2}\pi(\mathbf{a}|\mathbf{s}) + \frac{1}{2}\rho(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1\right]. \\
&= \mathcal{B}^\pi \bar{Q}(\mathbf{s},\mathbf{a}) - \alpha\left[\frac{\pi(\mathbf{a}|\mathbf{s}) + \rho(\mathbf{a}|\mathbf{s})}{2\pi_\beta(\mathbf{a}|\mathbf{s})} - 1\right]. \\
&= \mathcal{B}^\pi \bar{Q}(\mathbf{s},\mathbf{a}) - \alpha\left[\frac{\pi(\mathbf{a}|\mathbf{s}) + \rho(\mathbf{a}|\mathbf{s}) - 2\pi_\beta(\mathbf{a}|\mathbf{s})}{2\pi_\beta(\mathbf{a}|\mathbf{s})}\right]. \\
&= \mathcal{B}^\pi \bar{Q}(\mathbf{s},\mathbf{a}) - \alpha\left[\frac{\pi(\mathbf{a}|\mathbf{s}) + \pi_\beta(\mathbf{a}|\mathbf{s})g(.) - 2\pi_\beta(\mathbf{a}|\mathbf{s})}{2\pi_\beta(\mathbf{a}|\mathbf{s})}\right].
\end{aligned}
$$

since ρ subsumes $\pi_\beta \cdot g$, giving us the desired result.

B.6 Proof of Lemma 4.2

Intuition: Comparison of CQL and CQL (ReDS) objectives We will first intuitively compare the objectives in CQL and CQL (ReDS) to understand where the difference arises from. We write down the CQL (ReDS) objective below:

$$
\max_{\pi\in\Pi}\ \widehat{J}(\pi) - \frac{\alpha}{2(1-\gamma)}\mathbb{E}_{\mathbf{s}\sim\widehat{d}^\pi}\left[D(\pi,\pi_\beta)(\mathbf{s}) + \cdot \mathop{\mathbb{E}}_{\mathbf{a}\sim\pi(\cdot|\mathbf{s})}\left[g\left(\tau\cdot\pi(\mathbf{a}|\mathbf{s})\right)\mathbb{I}\left\{\pi_\beta(\mathbf{a}|\mathbf{s}) > 0\right\}\right]\right]. \tag{43}
$$

and now the CQL objective:

$$
\max_{\pi\in\Pi}\ \widehat{J}(\pi) - \frac{\alpha}{(1-\gamma)}\mathbb{E}_{\mathbf{s}\sim\widehat{d}^\pi}\left[D(\pi,\pi_\beta)(\mathbf{s})\right]. \tag{44}
$$

Observe that the only difference between CQL and ReDS stems from the fact that while the regularizer in CQL only optimizes the policy to stay close to the behavior policy, the regularizer in ReDS minimizes an additional term $\sum_{\mathbf{a}}\pi(\mathbf{a}|\mathbf{s})g(\tau\cdot\pi(\mathbf{a}|\mathbf{s}))$. This term attempts to make the behavior policy more "sharp", as g is a monotonically decreasing function of its argument, by preventing π to be less sharp in states where π_β is broad. This enables π to find the action in the dataset support that maximizes the learned return $\widehat{J}(\pi)$, even when π_β is broad.

To prove Lemma 4.2, we will consider the following abstract update form for the policy evaluation version of CQL (ReDS), that obtains the next Q-function iterate Q_{k+1}:

$$
\min_{Q}\ \alpha\left(\mathop{\mathbb{E}}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\pi^{re}}\left[Q(\mathbf{s},\mathbf{a})\right] - \mathop{\mathbb{E}}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\pi_\beta}\left[Q(\mathbf{s},\mathbf{a})\right]\right) + \frac{1}{2}\mathop{\mathbb{E}}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q(\mathbf{s},\mathbf{a}) - \mathcal{B}^\pi Q_k(\mathbf{s},\mathbf{a})\right)^2\right], \tag{45}
$$

Lemma B.5 (CQL (ReDS) restated more completely.)**.** *CQL (ReDS) solves the following optimization problem, when α is large enough:*

$$
\max_{\pi}\ \widehat{J}(\pi) - \frac{\alpha}{2(1-\gamma)}\mathbb{E}_{\mathbf{s}\sim\widehat{d}^\pi}\left[D(\pi,\pi_\beta)(\mathbf{s}) + \mathbb{E}_{\mathbf{a}\sim\pi(\cdot|\mathbf{s})}\left[g\left(\tau\cdot\pi(\mathbf{a}|\mathbf{s})\right)\mathbb{I}\left\{\pi_\beta(\mathbf{a}|\mathbf{s}) \geq 0\right\}\right]\right].
$$

Proof. For proving Lemma 4.2, we follow an argument similar to the proof of Theorem 3.1 from Kumar et al. (2020). By differentiating the above objective w.r.t. Q, we note that the ReDS + CQL objective above exhibits the following effective Bellman backup

$$
\forall \mathbf{s},\mathbf{a}\in\mathcal{D}, Q_{k+1}(\mathbf{s},\mathbf{a}) := \left(\mathcal{B}^\pi Q_k\right)(\mathbf{s},\mathbf{a}) - \alpha\left(\frac{\pi^{re}(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1\right). \tag{46}
$$

This backup is equivalent to running pessimistic RL with a reward bonus equal to $-\alpha\left(\frac{\pi^{re}(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1\right)$, and therefore, the policy obtained by maximizing the resulting Q-function can be expressed as:

$$
\max_{\pi}\ \widehat{J}(\pi) - \alpha\frac{1}{1-\gamma}\mathbb{E}_{\mathbf{s}\sim\widehat{d}^\pi}\left[\mathbb{E}_{\mathbf{a}\sim\pi(\cdot|\mathbf{s})}\left[\left(\frac{\pi^{re}(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1\right)\right]\right] \tag{47}
$$

$$
\equiv \max_{\pi}\ \widehat{J}(\pi) - \alpha\frac{1}{2(1-\gamma)}\mathbb{E}_{\mathbf{s}\sim\widehat{d}^\pi}\left[D(\pi,\pi_\beta)(\mathbf{s})\right] - \frac{\alpha}{2(1-\gamma)}\mathbb{E}_{\mathbf{s},\mathbf{a}\sim\widehat{d}^\pi}\left[\mathbb{I}\{\pi_\beta(\mathbf{a}|\mathbf{s}) > 0\}g\left(\tau\cdot\pi(\mathbf{a}|\mathbf{s})\right)\right]. \tag{48}
$$

We retain the notion of support $\mathbb{I}\{\pi_\beta(\mathbf{a}|\mathbf{s}) > 0\}$ because $\pi_\beta(\mathbf{a}|\mathbf{s})$ in the denominator only cancels out for actions within the support of the behavior policy, as for all other actions, this term the term would be ill-defined as $\pi_\beta(a|s) = 0$ appears in the denominator. Hence this term in the above equation cannot sum up over actions $\mathbf{a}$ not in the behavior policy in the second term (these actions will be pushed down to have $-\infty$ Q-values). If we just cancelled the π_β in the denominator, note that $g(\tau \cdot \pi(a|s))$ is not guaranteed to be 0 on actions $\mathbf{a}$, where $\pi_\beta(a|s) = 0$. Hence, we must retain the indicator function to explain this result properly. Instead, for the case of standard CQL, the divergence $D(\pi, \pi_\beta)(s)$ term prevents π from putting a non-zero density on actions where $\pi_\beta(a|s)$ in CQL, or else $D(\pi, \pi_\beta)(\mathbf{s}) = \infty$, meaning that the Q-value for such a π would be $-\infty$. □

C Implementation details of CQL (ReDS)

In this section, we will provide implementation details about our algorithm, CQL (ReDS). The pseudo-code in Algorithm 1 illustrates the different update steps of our algorithms. In addition, we provided a detailed python-like algorithm description for ease of implementation. This can be found below in Section C.1.

Most of the components of Algorithm 1 are straightforward and follow the same convention, training update and, as we will discuss, hyperparameters as the CQL algorithm. This includes training the policy π_ϕ, and for the most part training the critic Q_θ. The main difference in the update for CQL (ReDS) is utilizing the mixture of π and ρ in the CQL regularizer. For obtaining ρ_ψ, we utilize a standard advantage-weighted training update, following the papers (Kostrikov et al., 2021a; Nair et al., 2020b; Peng et al., 2019). Following these prior works, we also clip the argument to the exponent between a minimum range and a maximum range to be numerically stable:

$$\mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}[\log \rho_\psi(\mathbf{a}|\mathbf{s}) \cdot \exp\left[\text{clip}\left(-A^\pi_\theta(\mathbf{s},\mathbf{a})/\tau, \sigma_{\min}, \sigma_{\max}\right)\right]. \tag{49}$$

In our experiments, we chose $\sigma_{\min} = -10$ and $\sigma_{\max} = 5$ across all the tasks and domains we study. These details are standard in training advantage-weighted algorithms.

C.1 Detailed algorithm description for CQL (ReDS)

Algorithm 1 provides the pseudo-code for CQL (ReDS). We provide the detailed description of how each update step in Algorithm 1 is implemented using Python syntax based on the PyTorch Framework in this section. We include 3 code listings below, illustrating the update steps for the parametric Q-functions, the policy and the learnt distribution ρ.

Listing 1: Training Q networks given a batch of data, corresponding to step 3 in Algorithm 1

```
q_data = critic(batch['observations'], batch['actions'])

next_dist = actor(batch['next_observations'])
next_pi_actions, next_log_pis = next_dist.sample()

target_qval = target_critic(batch['observations'],
                            next_pi_actions)
target_qval = batch['rewards'] + \
    self.gamma * (1 - batch['dones']) * target_qval

td_loss = mse_loss(q_data, target_qval)

# importance sampling term
num_samples = 4

# assume env is normalized between [-1, 1]
random_actions = uniform_sample((num_samples,
    batch['actions'].shape[-1]), min=-1, max=1)
random_pi = 0.5 ** batch['actions'].shape[-1]

dist = actor(batch['observations'])
pi_actions, log_pis = dist.sample(num_samples)

rho_dist = rho(batch['observations'])
rho_actions, log_probs_rho = rho_dist.sample(num_samples)

q_rand_is = critic(batch['observations'],
                   random_actions) - random_pi
q_pi_is = critic(batch['observations'],
                 pi_actions) - log_pis
q_rho_is = critic(batch['observations'],
                  rho_actions) - log_probs_rho

cat_q = concatenate(q_rand_is, q_pi_is, new_axis=True)
cat_q = logsumexp(cat_q, axis=-1)

cat_q_rho = logsumexp(q_rho_is, axis=-1)

# average between rho and pi
push_down_term_reds = 0.5 * (cat_q + cat_q_rho)

reds_loss = td_loss + \\
    ((push_down_term_reds - q_data).mean() * cql_alpha)

critic_optimizer.zero_grad()
reds_loss.backward()
critic_optimizer.step()
```

Listing 2: Training the policy (or the actor) given a batch of data (step 4 in Algorithm 1)

```
# Identical to CQL
# return distribution of actions
dist = actor(batch['observations'])

# sample actions with associated log probabilities
pi_actions, log_pis = dist.sample()

# calculate q value of actor actions
qpi = critic(batch['observations'], actions)
qpi = qpi.min(axis=0)

# same objective as CQL (kumar et al.)
actor_loss = (log_pis * self.alpha - qpi).mean()

# optimize loss
actor_optimizer.zero_grad()
actor_loss.backward()
actor_optimizer.step()
```

Listing 3: Training the ρ_ψ distribution given a batch of data (step 5 in Algorithm 1)

```
# AWR style update to find rho

# sample policy actions for advantage calculation
dist = actor(batch['observations'])
pi_actions, log_pis = dist.sample()

# calculate advantage
qdata = critic(batch['observations'], batch['actions'])
value = critic(batch['observations'], actions)
advantage = (qdata - value.min(0)).mean()

# awr style clipping
clipped_advantage = clip(advantage/self.temperature, \
    min=-10, max=5)

# find log rho(a|s)
rho_dist = _rho(batch['observations'])
log_prob_rho = rho_dist.log_prob(batch['actions'])

# Advantage Weighted Log Probabilities is the loss for rho
rho_loss = -(exp(-clipped_advantage) * log_prob_rho)
rho_loss = rho_loss.mean()

rho_optimizer.zero_grad()
rho_loss.backward()
rho_optimizer.step()
```

D TASK AND DATASET DESCRIPTIONS

In this section, we will describe the various tasks we introduce in this paper. We also provide qualitative descriptions of these tasks here.

Heteroskedastic antmaze navigation. We introduce four new antmaze datasets which exhibit two different dataset distributions each for the medium and large mazes from D4RL (Fu et al., 2020). We reuse the layouts of the mazes directly from D4RL, but modify the data collection protocol. For the `noisy` datasets, given an observation from the environment, we first compute the action that would have been taken by the D4RL behavior policy, and then add Gaussian noise to the action. While this alone is not much harder, crucially, note that the variance of this added Gaussian noise differs depending on the location of the Ant in the 2D Maze. In addition there is a small bias added to the D4RL behavior policy, but this bias is dominated by noise. We present the noise standard deviations (indicated "Noise") and the bias added (indicated "Bias") for this dataset as a function of different location intervals in the maze in the left part of the Figure 9 below.

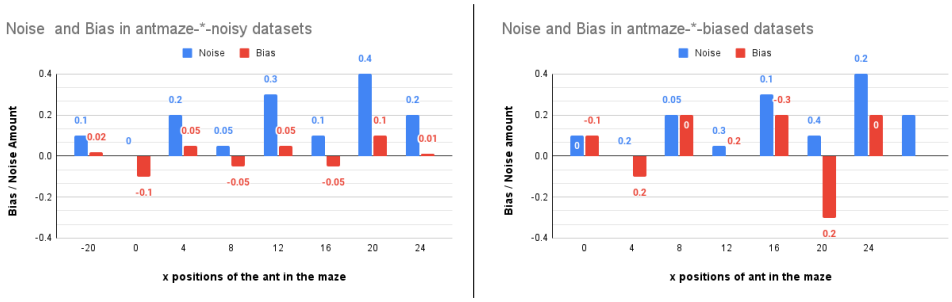

Figure 9: The distribution of noise and bias in the heteroskedastic antmaze datasets as a function of the x-position of the ant in the maze. While the noisy datasets primarily add noise, the biased datasets also add significant bias beyond the noise. The value of a given bar is the variance of the noise / bias added in the region between the x-position for that bar, and the next one.

For the `biased` datasets, in addition to adding location-dependent Gaussian noise to the action computed by D4RL behavior policies, we add a strong bias to the action (see Figure 9 (right)). Crucially note that the direction of this bias (i.e., the sign) changes based on the location of the Ant in the 2D maze, which mimics the scenario studied in our didactic navigation example in Section 3.1. In summary, because in some 2D regions of the maze, the values of the noise and bias added to the actions are larger, while in other 2D regions, they are smaller, the new offline datasets contain more heteroskedastic data distribution, where an optimal learned policy must deviate away from the data distribution much more in certain regions, whereas much lesser in other regions, which would correspond to an increase in the differential concentrability. This is demonstrated quantitatively in Table 4a. Thus, we expect that learning well on these tasks modulating the strength of the distributional constraints per state.

Visual robotic pick and place. We introduce a pick and place dataset, which exhibits a unique dataset distribution for a robotic pick and place manipulation task, building on the framework from Singh et al. (2020). As shown in Figure 11, the robotic setup is a 6-DOF WidowX robot in front of a green bowl with 2 objects: a target object (the ball in this case) and a distractor object (the can). The objective is to place the target object into the bin. The reward function is a sparse, binary indicator of success, where a +1 reward is given when the object is placed in the bin. This task must be done from $128 \times 128 \times 3$ raw visual observations, without access to either the robot state, or the state of the objects, which can change as the objects can roll on the surface.

Visual robotic bin sorting. We introduce a bin sorting tasks, which are also built on the framework from Singh et al. (2020). As shown in Figure 11, the robotic setup is a 6-DOF WidowX robot in front of two identical white bins with 2 objects to sort. The objective is to sort each object into its respective bins the target object in to the bin. The reward function is a sparse, binary indicator of success, where a "+1" reward is given when both objects are placed in their correct bins. This task must be done from $128 \times 128 \times 3$ raw visual observations, without access to either the robot state, or the state of the objects, which can change as the objects can roll on the surface.

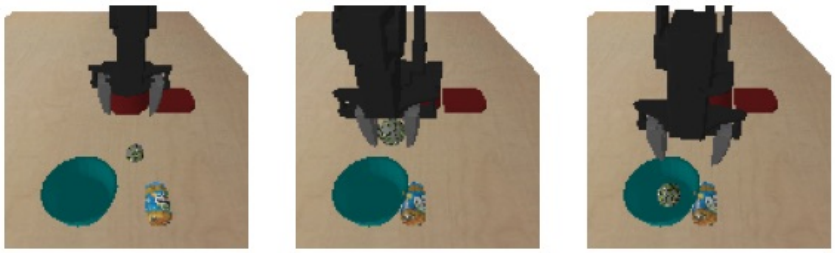

Figure 10: **Visualizing a sample trajectory for the visual pick-place robotic manipulation task.** Here is an example successful trajectory in the dataset collected using a scripted policy. Th robot reaches for the target object (the green ball), lifts it, and places it inside of the green container.

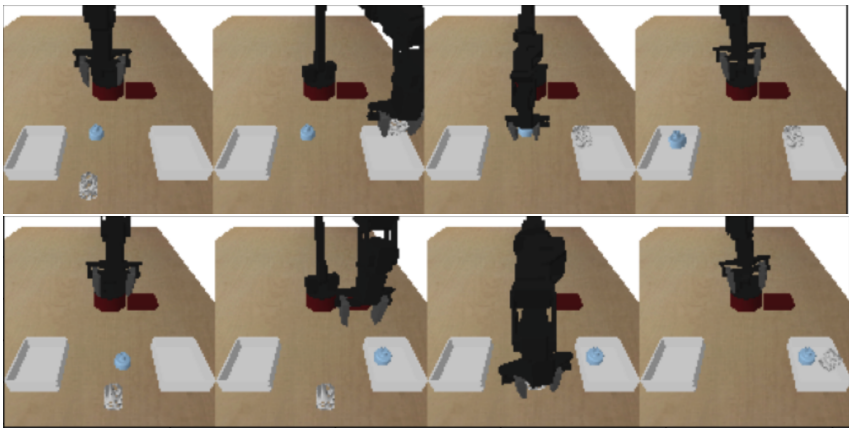

Figure 11: **Visualizing sample trajectories for the visual bin sorting robotic manipulation task.** Here are two sample trajectories for the binsorting domain. **Top:** A successful trajectory in the dataset. Here the robot places the white cylinder in the right bin and the blue ball in the left bin, succesfully sorting the objects into their respective bins. **Bottom:** A unsuccessful trajectory in the dataset. Here the robot places both objects in the same bin. This is unsuccessful as an object was placed in the incorrect bin thereby not sorting them in a correct manner.

To collect a heteroskedastic dataset, we run data collection using hardcoded scripted control policies, whose success rate and variance can be controlled. Each trajectory in the dataset was collected in the following manner. For the first phase, the robot reaches towards the object without any bias and grasps it with a reasonable success rate. Here, though, the noise and stochasticity in the scripted data collection and the inaccuracies in the scripted policy make it not succeed for every trial. During the second phase, where the robot places the object in the bin, there was a bias towards placing the object in a position in the workspace that does not correspond to the target location of the bin for which the robot can attain a reward, and which the robot will observe during evaluation. For our experiments, this bias was 85%. This forces the data distribution to be heteroskedastic: for the picking segment of this task, the behavior policy is centered around the desired optimal behavior i.e., grasping the object successfully, whereas for the placing segment, the behavior is biased towards carrying the object to the incorrect regions, requiring significant deviations from the behavior policy to succeed. An algorithm is now required to have a non-uniform amount of closeness to the behavior policy.

Atari game playing. For the Atari tasks we consider in the paper, we devised a heteroskedastic data composition based on the DQN Replay dataset (Agarwal et al., 2020) which comprises of transitions found in the replay buffer of a run of an online DQN. Since this dataset consist of all the policies that the DQN agent produced over the course of training, and since Atari typically uses ϵ-greedy exploration, where the value of ϵ decays over time, different trajectories of this dataset are generated from different behavior policies, that all have different levels stochasticity. Naturally, since the value of ϵ decays over training and the performance of online DQN increases, the trajectories with higher return generally correlate with having lower stochasticity.

Given this information, we attempted to subsample a dataset that is heteroskedastic. For this purpose, we first divide the full replay buffer into N equal chunks, where chunk 0 consists to experience observed earliest in training, while the chunk $N-1$ consists of experience seen near the final parts of training. Then, we subsample 20% of the trajectories from each of these chunks independently to obtain an intermediate dataset that comes from multiplies policies, observed at different times while training online DQN. Then, for any given trajectory τ of length L in the replay chunk i, we only retain the transitions occurring between time steps $\lfloor \frac{(L-i)\times N}{L} \rfloor : \lfloor \frac{(L-i+1)\times N}{L} \rfloor$ in our final dataset and discard all the remaining transitions. This essentially means that the data closer to the initial states of the game comes from a good, less stochastic policy, whereas the data close to the final states of the game from a worse, highly stochastic policy. We develop two such datasets corresponding to $N = 2$ and $N = 5$ chunks. These chunks concatenated together construct the replay buffer of transitions that correspond to the 2 and 5 policy experiments seen in Section 5.

To see why this data is heteroskedastic, note that at different states of the game, we observe actions with different amounts of stochasticity and bias. This is because, as the game progresses, the effective behavior policy induced by the offline dataset exhibits a bias towards suboptimal actions (from the chunks that are earlier in DQN training) while also exhibiting substantial noise. The states that are closer to the initial states of the game, on the other hand, have an effective behavior policy that is primarily centered around a good action, with very little noise. In order to succeed, an offline RL algorithm must have different amount of conservatism at different states.

In our experiments, we considered 10 games including several standard games, and this is a subset of games studied in prior work (Kumar et al., 2021). The games we considered are: ASTERIX, BREAKOUT, Q*BERT, SEAQUEST, SPACEINVADERS, BEAMRIDER, MSPACMAN, WIZARDOF-WOR, JAMESBOND, PONG.

E EXPERIMENTAL DETAILS

For our experiments on the AntMaze domains, we built on the following open-source implementation of CQL: https://github.com/young-geng/JaxCQL, for our visual robotic experiments, we utilized our own port of the following implementation from Singh et al. (2020) in Jax: https://github.com/avisingh599/cog, and for our Atari experiments, we use the official implementation of CQL built on Dopamine (Castro et al., 2018): https://github.com/aviralkumar2907/CQL/tree/master/atari. For certain baselines (e.g EDAC, BEAR), we utilize the source implementation to stay consistent with the author's tested and tuned implementation. We additionally verified the results for D4RL benchmark for these tasks. We will summarize the hyperparameters in the next sections.

E.1 HYPERPARAMETERS FOR CQL (REDS)

Antmaze domains. For the AntMaze domains, we utilized a temperature parameter $\tau = 0.3$ in our experiments (found by sweeping over $\tau \in [0.1, 0.3, 1.0, 5.0]$), for all the four dataset types in Table 4b. Every other hyperparameter was kept identical to CQL, which for the case of antmaze corresponds to applying the CQL regularizer $\mathcal{R}(\theta)$ with the dual version of CQL, where the threshold on the CQL regularizer is specified to be 1.0. Following CQL, we used 3-hidden layer critic and actor networks with layers of size 256, a critic learning rate of 3e-4 and an actor learning rate of 1e-4. We utilized the Bellman backup that computes the target value by performing a maximization over target values computed for $k = 10$ actions sampled from the policy at the next state.

Atari domains. For our Atari experiments, we tuned the value of α in CQL (Equation 2) between two values $[0.1, 0.2]$, and present the sensitivity results in Figure 5, and found that $\alpha = 0.1$ work better for CQL. We swept the value of $\tau \in [2.0, 5.0, 7.0]$ and report the sensitivity sweep in Figure 12.

Visual Robotic Domains. For the visual pick and place domains, we follow exactly the same hyperparameters as the CQL implementation from COG (Singh et al., 2020): a critic learning rate of 3e-4, an actor learning rate of 1e-4, using $k = 4$ actions from the policy for computing the target values for computing the TD error, and using $k = 4$ actions to compute the log-sum-exp in CQL. For the value of τ, we swept over $\tau \in [0.1, 1.0, 10.0, 100.0]$, and used a $\tau = 1.0$ for our experiments.

F Additional Ablation Studies

In this section, we present some results of an ablation study of the performance of CQL (ReDS) with respect to the temperature hyperparameter τ that appears in Equation 9. Before discussing the results, let us intuitively aim to understand the significance of this hyperparameter. When τ is extremely small we would expect ρ_ψ to be a distribution centered at the worst possible action, within the support of the behavior policy. When τ is large, we would expect the learned ρ_ψ to be close to the behavior policy, since the exponentiated advantage term would essentially behave as a constraint against a uniform distribution. Neither of these extremes are desirable, while the former does not behave much differently than a distributional constraint (except that the Q-value at the action with the smallest Q-value in the dataset support is not pushed up anymore), the latter also behaves like a distributional constraint, but with just half the effective multiplier α on the CQL regularizer. We would therefore expect an intermediate τ to perform the best.

To verify these insights, we study the sensitivity of the performance of CQL (ReDS) with respect to α on the Atari datasets. Our results shown in Figure 13 confirm that indeed an intermediate value of $\tau = 5.0$ out of the tested values, $\tau \in [2.0, 5.0, 7.0]$ works the best.

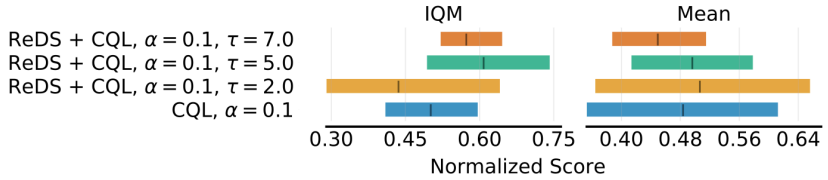

Figure 12: **Sensitivity of CQL (ReDS) to the temperature hyperparameter τ in Equation 9 evaluated on the Atari game experiments with 5 policies.** Observe that an intermediate value of temperature $\tau = 5.0$ works best

In addition, we study the sensitivity of the performance of CQL (ReDS) with respect to α on the Atari datasets. We report the performance for two different values of $\alpha \in \{0.1, 0.2\}$ from CQL (Equation 2) in Figure 13. Observe that CQL (ReDS) with a given α outperforms base CQL for the corresponding α. Additionally note that the degradation in performance of CQL (ReDS) as α increases is lesser than base CQL.

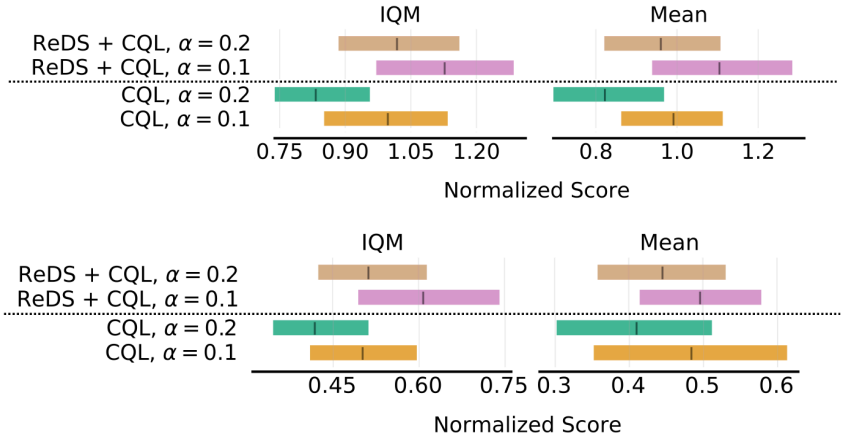

Figure 13: **Sensitivity of ReDS + CQL to the temperature hyperparameter α in Equation 2** We report the performance of CQL (ReDS) vs CQL on the IQM normalized score and the mean normalized score over ten Atari games, for the case of **two** (top) and **five** (bottom) policies. We consider this performance for two different values of $\alpha \in \{0.1, 0.2\}$ in CQL (Equation 2). Observe that CQL (ReDS) with a given α outperforms base CQL for the corresponding α. Additionally note that the degradation in performance of ReDS (CQL) as α increases is lesser than base CQL.

G Additional Baseline Comparison for Heteroskedastic Antmaze Navigation

In this section we will provide additional baseline comparison for REDS with two additional Offline RL methods: EDAC (An et al., 2021) and BEAR (Kumar et al., 2019).

G.1 Hyperparameters for EDAC

As done in An et al. (2021), we tune the method over two hyperparameters. The first hyperparameter is the ensemble size N which specifies the number of Q functions. The second parameter we consider is η, the weight of the ensemble gradient diversity term. Below in table 4, we show the values considered for each hyperparameter. There is significant overlap to these parameters with the ones used in the Mujoco Gym and Adroit Domains that the authors used. We utilized the publicly available code (`https://github.com/snu-mllab/EDAC`) released by the authors of EDAC and were able to replicate the results they reported for the D4RL MuJoCo Gym environments in An et al. (2021).

Table 4: EDAC Hyperparameters

Hyperparameters	**Values**
N	10, 20, 50, 100
η	0, 1, 5, 10, 50, 100, 1000

G.2 Hyperparameters for BEAR

As done in Kumar et al. (2019), we tuned this method over two hyperparameters. The first is the Kernel Type of the MMD between the behavior policy π_β and the actor π, and found that Laplacian performed better. The second parameter considered is σ, which is needed for the Laplacian kernel as defined. Below in table 5, we show the values considered for each hyperparameter. There is significant overlap to these parameters with the ones used in the Mujoco Gym and Adroit Domains that the authors used. We utilized the publicly available code (`https://github.com/rail-berkeley/d4rl_evaluations`) released by the authors of BEAR and were able to replicate the results they reported for the D4RL MuJoCo Gym environments in Kumar et al. (2019).

Table 5: BEAR Hyperparameters

Hyperparameters	**Values**
Kernel Type	Laplacian, Gaussian
σ	1, 10, 20, 50