Exploration for the Efficient Deployment of Reinforcement Learning Agents

Siddhant Agarwal[1,†,*], **Max Rudolph**[1,†,*], **Omer Gottesman**[2], **Amy Zhang**[1], **Akhil Bagaria**[2], **Sohrab Andaz**[2], **Udaya Ghai**[2], **Carson Eisenach**[2]
mrudolph@cs.utexas.edu

[1]**The University of Texas at Austin**
[2]**Amazon**

[†] denotes equal contribution, order chosen by coin flip
[*] work done while interns at Amazon

Abstract

Reinforcement learning (RL) provides a rich toolbox with which to learn sequential decision making policies. Notably, the ability to learn solely from offline interaction data has been a highly successful modality for training real-world policies without ever interacting in the real world. However, a gap exists in this paradigm when the offline dataset does not cover all the behaviors necessary to extract optimal policies. Naively, one can pre-train a policy using offline RL and fine-tune it using online RL; this can lead to catastrophe in safety critical settings, like healthcare and autonomous driving, where deploying an unverified policy is irresponsible. Deployment efficient learning is a potential solution, where the number of distinct data collection policies is relatively low compared to the number of updates to the policy. We argue that safely improving a dataset requires a deployment efficient algorithm with a carefully constructed data collection policy. We introduce a framework with a stationary exploration policy that aims to reduce out-of-distribution uncertainty while maintaining strong returns. We establish theoretical guarantees of this exploration framework without finetuning and demonstrate our methods on a small-scale toy environment and a large-scale supply chain environment with real-world data.

1 Introduction

Offline RL algorithms are popular in domains where online interactions can be costly or unsafe (Achiam & Amodei, 2019; Hu et al., 2024; Liu et al., 2023). To avoid potentially dangerous states, many offline algorithms (Kumar et al., 2020; Kostrikov et al., 2021) regularize the learned policy to take actions that stay within the distribution of interactions within the dataset. Naturally, the performance of policies trained with offline RL algorithms depends significantly on the coverage of this offline data. However, this restriction can be mitigated by performing some online exploration to improve the dataset's coverage.

Offline-to-online RL algorithms seek to overcome this limitation by refining offline trained policies via online interaction with the environment (Nair et al., 2020; Mark et al., 2024; Ball et al., 2023; Wang et al., 2023; Zhou et al., 2025). However, this approach falls short in the real world due to cost and safety considerations of online exploration. Real-world applications like health care or autonomous driving (Gottesman et al., 2019; Kiran et al., 2021) have little tolerance for error and typically require policies to be verified and tested before deployment. For this reason,

updating the policy parameters online, as most offline-to-online methods do, cannot be permitted in the real world as every update to the policy parameters must be thoroughly tested and verified. As testing and verification can be time consuming and expensive, it makes sense to use the number of policy deployments as a suitable measure of efficiency. Specifically, the concept of *deployment efficiency*, which is defined as the ratio of the number of unique policies deployed during training to the number of samples collected, is most relevant for many real-world systems (Matsushima et al., 2020). On-policy RL methods are at one extreme of the deployment efficiency spectrum, requiring one deployment per update. Purely offline methods are at the other end, only utilizing one deployment in total. In real-world settings, the only feasible way to improve an offline-trained policy is to explore in a manner that is neither costly in terms of deployment efficiency nor leads to losses due to poor selection of actions.

A high-profile and socio-economically relevant setting in which deployment efficiency is crucial is that of supply chain management (Madeka et al., 2022), where an agent must place inventory orders to vendors to stock a company's warehouses. In this domain, state-of-the-art RL agents (Andaz et al., 2023) are trained offline using historical data collected from a traditional linear programming-based behavior policy. However, to improve the offline policy, it is necessary to collect quality online interactions through exploration in the real world, but each suboptimal action has severe financial overheads. Since policies need to be verified and back-tested for safety and performance (Corsi et al., 2024; Matsushima et al., 2020; Madeka et al., 2022; Amir et al., 2021) before deployment, it is difficult to justify continuous online finetuning. Instead, we propose to use a stationary exploratory policy to collect high quality samples that can be used to re-train the policies offline on the *augmented* dataset. This stationary exploration policy should satisfy two desiderata: (a) it should be *safe*, meaning that it should not deviate too far from the well-tested policy, and (b) its exploration should be *targeted*, meaning that it should carefully collect data that are close to the known safe regions, while still meaningfully augmenting the dataset for high-quality offline policy improvements for future deployments.

Our contributions can be summarized as: (1) establishing a framework for exploring through online interactions without fine-tuning the policy online, (2) developing a principled exploration strategy from the offline dataset that can collect informative samples through online interactions, and (3) empirically analyzing the samples collected using our exploration strategy on a discrete navigation environment and a real-world supply chain application (Madeka et al., 2022; Andaz et al., 2023).

2 Preliminaries

We consider problems that can be modelled as Markov Decision Processes (MDPs) (Puterman, 1990), which are defined as $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r, \gamma, \rho_0 \rangle$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \longmapsto \mathcal{S}$ is the transition probability function, $\gamma \in [0, 1)$ is the discount factor, ρ_0 is the initial state distribution and $r : \mathcal{S} \times \mathcal{A} \longmapsto \mathbb{R}$ is the reward function. A policy $\pi_\theta : \mathcal{S} \longmapsto \mathcal{A}$ represents the probability of taking an action a from state s. The policy induces a trajectory distribution $p^\pi(\tau) = \rho_0 \Pi_{t=1}^{\infty} \pi(a_t|s_t) P(s_{t+1}|s_t, a_t)$ and a corresponding state-visitation distribution $d^\pi(s) = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t p^\pi(s_t = s)$. The expected return is defined as $J(\pi_\theta, \mathcal{M}) = \mathbb{E}_{\tau \sim p^\pi(\tau)}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$. The goal of the agent is to find the optimal policy π^* that maximizes the expected return, i.e,. $\pi^* = \arg\max_\pi J(\pi, \mathcal{M})$.

Offline RL algorithms try to regularize the policy to avoid taking actions out of dataset support. The regularization can be in the form of pessimism in Q-value for out-of-distribution actions (Kumar et al., 2020; Shimizu et al., 2024) or pessimism in the reward function based on uncertainties in the model predictions (Yu et al., 2020; Kidambi et al., 2021). In some sense, it can be assumed that the general objective for offline RL algorithms are,

$$\begin{aligned} \max_{\pi} \quad & J(\pi, \mathcal{M}) \\ s.t. \quad & D(p^\pi(s,a) || \hat{p}(s,a)) \leq \epsilon, \end{aligned} \tag{1}$$

where D is written as a divergence, but can more generally be any measure of the difference between distributions, $p^\pi(s,a)$ is a measure of the true likelihood of state-actions under the policy, and $\hat{p}(s,a)$ is a measure of the likelihood of the state-action in the dataset (Nachum et al., 2019).

3 Method

The success of exploring in the real world hinges on two key considerations. First, the policy must targetedly collect data that that will augment the offline dataset in such a way that it improves future policy deployments. Second, the exploration conducted online must be "safe", meaning that it should carefully stay close to the support of the offline data. In this section, we present the theoretical formulation of our efficient exploration framework, along with practical approximations. We obtain the optimal exploratory policy by improving the performance of the policy extracted from running offline RL algorithms on the historical dataset.

3.1 The Exploratory Policy

First, we construct the exploratory policy given a dataset $\mathcal{D} = \{(s_i, a_i, s'_i, r_i)\}_{i=1}^N$; this is done using a distribution matching objective against a target distribution. First, we formalize the sub-optimality of a policy obtained using an offline dataset. We define $u(s,a) \approx \log p((s,a) \notin D)$ to provide a measure of the likelihood of (s,a) not being in the dataset. Using $u(s,a)$ we can formally define the uncertain set (Kidambi et al., 2021):

Definition 3.1. Let $\mathcal{U}_\mathcal{D}$ be the set of state-action pairs (s,a) that are not contained in the dataset D. Let $u(s,a)$ represent the likelihood of $(s,a) \notin \mathcal{D}$. Define $\mathcal{U}_\mathcal{D} = \{(s,a)|u(s,a) > \alpha\} \approx \{(s,a) \in \mathcal{S} \times \mathcal{A}|(s,a,.,.) \notin \mathcal{D}\}$, where α is a scalar parameter.

If $(s,a) \in \mathcal{U}_\mathcal{D}$, $p^\pi(s,a)$ would be low and if the probability $\pi(a|s)$ is high, then $D(p^\pi(s,a)||\hat{p}(s,a))$ would be high. In other words, creating this partition in the state space allows the creation of an approximation of the given MDP $\mathcal{M}$ where the expected return is a pessimistic estimate of the objective in Equation (1). This pessimistic MDP will ensure that the trained policy will restrict the policy from taking actions outside the support of the dataset (Kidambi et al., 2021).

Definition 3.2. Given an MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r, \gamma, \rho_0 \rangle$, a dataset $\mathcal{D} = \{(s_i, a_i, s'_i, r_i)\}_{i=1}^N$, define a pessimistic MDP $\mathcal{M}_\mathcal{D} = \langle \mathcal{S} \cup HALT, \mathcal{A}, P_p, r_p, \gamma, \rho_0 \rangle$ where,

$$P_p(s'|s,a) = \begin{cases} \delta(s' = HALT), & (s,a) \in \mathcal{U}_\mathcal{D} \\ P(s'|s,a), & \text{otherwise} \end{cases} \qquad r_p(s,a) = \begin{cases} -R_{max}, & (s,a) \in \mathcal{U}_\mathcal{D} \\ r(s,a), & \text{otherwise} \end{cases}$$

Additionally, we make the following assumption to derive the suboptimality gap in the offline data. This assumption is also made by most offline RL algorithms using Equation (1) as their objective. It simply states that the offline RL algorithm produces the optimal policy for the pessimistic MDP $\mathcal{M}_\mathcal{D}$.

Assumption 3.3. Let π^*_D be the best policy obtained from a dataset $\mathcal{D}$, essentially by optimizing the objective in Equation (1). We assume that π^*_D maximizes the expected return in $\mathcal{M}_\mathcal{D}$ i.e. $J(\pi^*_D, \mathcal{M}_\mathcal{D}) \geq J(\pi, \mathcal{M}_\mathcal{D}) \quad \forall \quad \pi$.

Let $d^\pi(\mathcal{X})$ be the visitation of set $\mathcal{X}$ following policy π and using the true dynamics P. We can now bound the true returns of the best policy obtained using the offline RL objective eq. (1) with the return of the optimal policy on MDP $\mathcal{M}$ or in other words define the suboptimality gap in the offline dataset.

Theorem 3.4. *Let $\mathcal{M}$ be a given MDP and $\mathcal{D}$ be the offline dataset. Let $\pi^* = \arg\max_\pi J(\pi, \mathcal{M})$ be the true optimal policy on $\mathcal{M}$ and $\pi^*_D = \arg\max_\pi$ Equation (1) $\approx \arg\max_\pi J(\pi, \mathcal{M}_\mathcal{D})$ be the best offline RL policy on dataset $\mathcal{D}$. The following holds: $J(\pi^*, \mathcal{M}) - J(\pi^*_D, \mathcal{M}) \leq \frac{2R_{max}}{(1-\gamma)^2} d^{\pi^*}(\mathcal{U}_\mathcal{D})$.*

A direct consequence of Theorem 3.4 is that if $d^{\pi^*}(\mathcal{U}_\mathcal{D}) = 0$, i.e. all transitions taken by the optimal policy are in the dataset $\mathcal{D}$, the optimal return of the best offline policy and the true optimal policy are the same. In other words, the best offline policy is in fact the true optimal policy. In general, if we want to improve the dataset, we need to tighten the bound or minimize $d^{\pi^*}(\mathcal{U}_\mathcal{D})$. This means the exploratory policy must aim to maximize this visitation.

Formally, let $\mathcal{U}^*_{\mathcal{D}}$ be the set of state-action pairs visited by the optimal policy π^* which are in the set $\mathcal{U}_{\mathcal{D}}$. Mathematically, $p((s,a) \in \mathcal{U}^*_{\mathcal{D}}) = d^{\pi^*}\delta((s,a) \in \mathcal{U}_{\mathcal{D}})$. In practical situations, it is difficult to obtain the set $\mathcal{U}_{\mathcal{D}}$. Rather, we will be using the measure $u(s,a)$ to provide an estimate of $(s,a) \in \mathcal{U}_{\mathcal{D}}$. We will approximate $\delta((s,a) \in \mathcal{U}_{\mathcal{D}})$ as $\frac{1}{Z(s)}\exp(u(s,a))$. The exploratory policy must be the one that maximizes its visitation for the state-action pairs in $\mathcal{U}^*_{\mathcal{D}}$. Using $p((s,a) \in \mathcal{U}^*_{\mathcal{D}})$ as a target distribution, we can obtain the optimal policy minimizing an f-divergence between the policy's visitation distribution and the target distribution as discussed in several previous works (Ma et al., 2022; Agarwal et al., 2024). We use Forward KL because of its performance and exploration properties. The optimization can be further simplified into a regularized RL objective.

$$\begin{aligned}\pi^*_{exp} &= \arg\min_{\pi} D_{FKL}(d^{\pi}(s,a)||d^*(s,a)\exp(u(s,a))) \\ &= \arg\max_{\pi} \mathbb{E}_{d^{\pi}}[u(s,a)] - D_{KL}(d^{\pi}(s,a)||d^*(s,a)).\end{aligned} \tag{2}$$

Practically, we do not have access to π^*, preventing us from defining π^*_{exp} exactly. We will use a biased estimate of π^*, which is the best offline policy, π^*_D. In general this isn't true, but if we restrict the marginal state visitation to be close to the dataset, i.e. ensuring $D(d^{\pi^*}(s)||d^{\pi^*_D}(s)) \leq \epsilon$, we can further simplify this objective to,

$$\pi^*_{exp} = \arg\max_{\pi} \mathbb{E}_{d^{\pi}}[u(s,a)] - \beta D_{KL}(\pi||\pi^*_D). \tag{3}$$

Common offline RL algorithms (Kostrikov et al., 2021; Kumar et al., 2020) learn π^*_D that are highly constrained to induce transitions that are found in D. For any given state, a policy trained using AWR (Peng et al., 2019b) has negligible probability of taking an action that is not present in the dataset, making out-of-distribution exploration difficult. We use fitted Q learning for some of our experiments to avoid the regularizations that are common to recent offline RL methods.

Note that the KL constraint on visitations has been approximated to be on behavior or policy. This approximation makes sense if the marginal state visitation is bounded to be close to the dataset. Further, it has been shown in Mao et al. (2024) that semi-gradient updates for visitation regularized offline RL is, in fact, performing the same updates as behavior regularized RL.

Since the policy π^*_{exp} needs to be obtained only from offline data, offline RL algorithms will overly constrain π^*_{exp} to be within the dataset, which is not ideal for an exploratory policy. Hence, we construct the exploratory policy to approximate the solution of Equation (3) directly using $\pi^*_D, u(s,a)$ and the offline dataset $\mathcal{D}$.

3.2 Approximate Construction

Neither online interactions nor offline RL can be used to optimize Equation (3). We use the following two sampling based approximations for π^*_{exp}:

Single Step: Equation (3) has a closed form solution as discussed in prior works (Peng et al., 2019a; Rafailov et al., 2024). π^*_{exp} can be written in closed form as,

$$\pi^*_{exp}(a|s) \propto \pi^*_D \exp\big(\frac{1}{\beta}u(s,a)\big). \tag{4}$$

Equation (4) provides a single step exploratory policy. The partition function can be obtained for simple discrete action spaces but for general action spaces, sampling from this distribution can be complicated.

Multi-Step: Given the difficulty of sampling from a distribution such as Equation (4), we also propose an alternative way to use the uncertainty metric $u(s,a)$ for decision-making. Specifically, we formulate the following online trajectory optimization problem with a learned dynamics model $\hat{T}$ and learned reward function $\hat{r}$,

$$\pi^*_{exp}(a|s) = \arg\max_{a_0,\cdots,a_{H-1}} \mathbb{E}_{\hat{\pi}} \left[\sum_{t=0}^{H-1} \gamma^t \left(\hat{r}(s_t, a_t) + c \cdot u(s_t, a_t) \right) + \gamma^H V^*_D(s_H) \right], \quad (5)$$

such that

$$s_0 = s, \quad a_t \sim \hat{\pi}(\cdot|s_t), \quad s_{t+1} \sim \hat{T}(s_t, a_t),$$

where $c \in (0, 1]$ is a hyperparameter that trades off between exploration and exploitation. The H-step rollouts are taken using $\hat{\pi}$ which a high-temperature version of the dataset policy:

$$\hat{\pi}(a|s) = \frac{e^{\pi^*_D(a|s)/T}}{\sum_{a'} e^{\pi^*_D(a'|s)/T}},$$

where $T \in \mathbb{R}^+$ is the temperature parameter of the softmax function. The argmax is over sequences of actions, but we can use model predictive control (MPC) and only commit to the first action in the sequence. This multi-step exploratory policy can be used to sample actions for arbitrary complex action spaces.

3.3 Rolling out the Exploration Policy

Given that we have π^*_{exp}, we can simply execute it to collect samples. However, we want to stay close to our "safe" dataset. We had made an assumption that $D(d^{\pi^*_{exp}}(s)||d^{\pi^*_D}(s)) \leq \epsilon$ which needs to maintained in the real world. Ideally we would want to start at each state in the offline dataset and run k-steps of exploration from there but resetting at a random state is not possible in the real world. We have the following choices,

Reset: The exploratory policy starts executing from the start-state defined by the initial state distribution. While this strategy is the most simple, the deviation from the "safe" policy grows as $\mathcal{O}(T^2)$ (Ross et al., 2010).

ϵ-exploration: This method mimics the epsilon greedy method. At any state, an exploratory action is taken according to π^*_{exp} with probability ϵ and a greedy action is taken with probability $(1 - \epsilon)$. Mathematically, this is synonymous to choosing a time step $t \sim Geom(\epsilon)$ to explore. This method can lead to better exploration with the exploration happening closer to π^*_D. One might argue that the time step could be chosen simply uniformly over the trajectory but the trajectory length is not often known which makes a uniform sampling difficult.

We describe the connections between our framework and well-studied exploration methods in Appendix B.

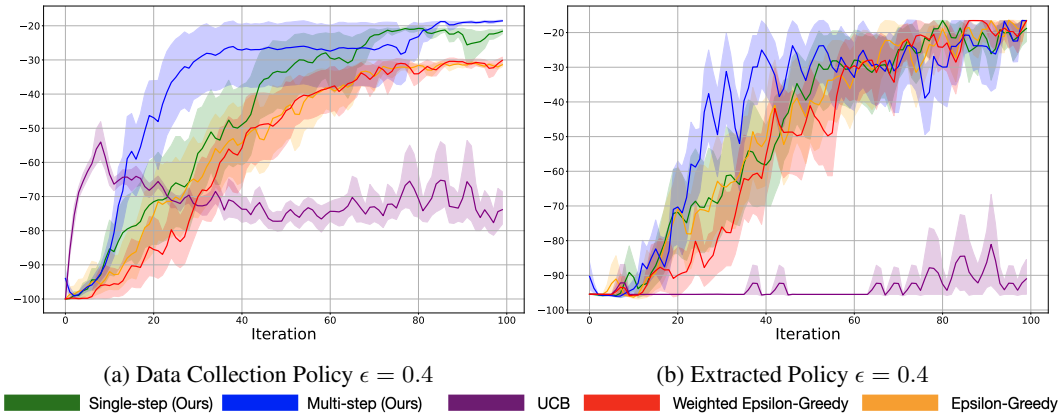
4 Experiments

We present experiments that represent the exploration challenges of the real world; Namely, we use a stationary exploratory policy throughout each deployment and desire the exploratory policy perform similarly to the exploitation policy. Through these experiments we aim to demonstrate that careful consideration of uncertainty is critical when deploying stationary exploration policies. We use two domains: 1) a discrete navigation environment called *Nav-Chambers* for ablating design choices and 2) a large scale supply chain environment that is backtestable with real-world data (Andaz et al., 2023; Madeka et al., 2022) and has in fact been used to train real-world policies.

4.1 Discrete Navigation Environment: Nav-Chambers

Environment Description: We create a discrete navigation environment called Nav-Chambers. The environment has discrete state and action spaces. It acts as a playground to run ablations and compare the different design choices for the exploratory policies. Nav-Chambers consists of a discrete space divided into four chambers using walls with passages to move across each other. The agent needs to traverse these chambers to reach to the other end of the grid (see Appendix D.2 for details).

Baselines: There are two variants of our exploration: **Single-Step** (defined in Equation (4)) and **Multi-Step** (defined in Equation (5)). These are compared against prototypical exploration method-



(a) Data Collection Policy $\epsilon = 0.4$ (b) Extracted Policy $\epsilon = 0.4$

Figure 1: Comparison of our exploration policies (Single-Step and Multi-Step) against the baselines using **collect-returns** (left) and **eval-returns** (right). Each curve shows a mean of 5 seeds with a rolling average of window length 5 and the shaded region represents the 95% confidence interval.

ologies: ***Naive*-ϵ-greedy** chooses an uniformly random action with probability ϵ and the greedy action with probability $1 - \epsilon$. ***Weighted*-ϵ-greedy** chooses an action based on the uncertainty with probability ϵ and the greedy action with probability $1 - \epsilon$. **Bandit-Style Reward Bonus** has no explicit exploration action, but trains a Q-network with an uncertainty augmented reward target where $r' = r(s, a) + c\frac{1}{\sqrt{N_{(s,a)}}}$, where $N(s, a)$ are tabular visitation counts, and $c \in [0, 1]$ is a fixed hyperparameter.

Experiment Setup: We iterate between "explore" and "train" phases as would happen in the real world. The *explore* phase executes the exploratory policy in the environment to collect $\mathcal{T}$ trajectories and the *train* phase uses the combined buffer to train the policy for K steps using fitted Q-learning (Ernst et al., 2005; Riedmiller, 2005). We perform N such deployments. Our exploratory policy (both Single-Step and Multi-Step) and baselines are run as $\epsilon-$**exploration**. We provide results for the **reset** setting in Appendix C.1. In these experiments, we show 100 deployments with 1000 environment and 1000 training steps, each.

As a measure of performance, we measure the mean return of the exploration policy and the greedy policy obtained from the offline RL algorithm and call these **eval-returns** and **collect-returns** respectively. **Collect-returns** is compared against the **eval-returns** to measure the deviation from the exploitation policy.

Uncertainty Estimation: Our exploration requires estimates of uncertainty of states with respect to the dataset. To isolate the design of the exploration policy, we use the ground truth state-action visitation to approximate uncertainty. Let $N_{(s,a)}$ denote the number of times the agent has taken action a in state s. As in previous work (Strehl & Littman, 2008; Lobel et al., 2023), the square-root inverse of the count $\frac{1}{\sqrt{N_{(s,a)}}}$ is used as a measure of uncertainty.

Results: Figure 1 shows the **collect-returns** (left) and **eval-returns** (right) of both of our exploratory policies compared against the baselines in the $\epsilon-$exploration setting. The results in Figure 1a demonstrate that our exploratory policies perform better when doing data collection than both variants of ϵ-greedy and all outperform the UCB baseline. We suspect the UCB baseline to perform poorly because the reward targets are non-stationary and change significantly between deployments. The performance of the exploitation policies, as shown in Figure 1b, is similar, which indicates the similar quality of the samples collected during exploration. In short, our and the baseline exploration methods collect similar data but our methods do so in a safer and more performant manner. Additional discussion and results that focus on the performance difference across different values of ϵ and annealed-ϵ-greedy can be found in Appendix E.1 and Appendix E.2, respectively.

4.2 Backtesting Exploration: Application to a Real-World Supply Chain

Next, we consider an application to an inventory control problem where a retailer needs to procure inventory and balance the costs associated with over-stocking versus missing sales due to under-stocking. Andaz et al. (2023) considers how to learn dynamics models—specifically for inventory arrivals from vendors—to construct a simulator for a large e-commerce retailer's supply chain. Transitions in the simulator are modeled through dynamics models (Andaz et al., 2023) trained using offline data. The dynamics models are only reliable on-policy (for the behavior policy) which affects the performance of any policy trained on the simulator. To get a higher quality policy, exploration is needed to improve the simulator. Policies can be trained using backpropagation through time (Madeka et al., 2022) within this simulator and launched in the real-world supply chain pipeline.

Experiment Setup: Let the simulator described in Madeka et al. (2022) be M. Andaz et al. (2023) showed that this simulator is well-calibrated with the real world. We create a suboptimal simulator $\hat{M}$ by training the dynamics models on data collected in M using a behavior policy which does not cover the entire state, action spaces. In our experiments, M acts as the proxy for the real world and $\hat{M}$ as the proxy for the simulator. The Supply Chain Environment is grounded in a dataset containing 5 years of real-world supply chain data (e.g., inventory levels, demand, delivery status, etc.) for thousands of products sold by a large e-commerce company. Further, to demonstrate the deployment efficiency regime, we segment the data into three chronological splits: T_1, T_2, and T_3 representing three, one, and one years of data, respectively. Let M_{T_i} be the simulator trained using the real-world data from time segment T_i. $\hat{M}_{T_i}$ denotes the simulator trained using the data collected from MDP M_{T_i} for time period T_i (Madeka et al. (2022) uses an approach where samples are replayed from historic data). Finally, we train a deterministic policy in M using DirectBackprop (Madeka et al., 2022) and denote it as π^*_M. The complementary exploration policy, π^*_{exp} is constructed from π^*_M and the uncertainty measure

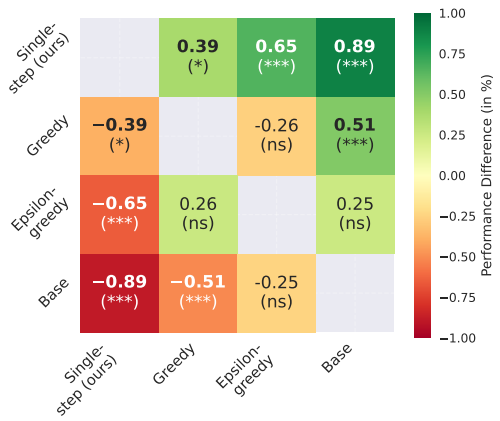


Figure 2: This figure shows a confusion matrix describing the relative difference in performance of a policy trained using data collected by the Single-step, Greedy, and Epsilon-greedy exploration methods as well as the original policy (Base). Each box represents the percentage difference of the `row` method against the `column` method. Bold numbers denote statistical significance and "ns" denotes not statistically significant. One, two, and three asterisks denote a p-value of < 0.05, < 0.01, and < 0.001, respectively.

We perform single deployment experiments in three phases: explore (i.e. deployment), retrain, and evaluate. At the start of the experiments, we are given $\hat{M}_{T_1}$ on which $\pi^*_{\hat{M}_{T_1}}$ is trained and π^*_{exp} constructed. Next, we deploy π^*_{exp} in M_{T_2} to collect data and subsequently train $\hat{M}_{T_{1,2}}$. Finally, $\pi^*_{\hat{M}_{T_{1,2}}}$ is trained and evaluated in M_{T_3}. Our experiments demonstrate the effectiveness of our uncertainty-seeking exploration policy.

We provide implementation details and experimental setup for the Supply Chain Environment in Appendix D.1 and uncertainty estimation in Appendix F.

Constructing π^*_{exp}: The Supply Chain Environment is high-dimensional and continuous, so it is infeasible to measure uncertainty using a count-based metric as is done in the Nav-Chambers environment. Instead, we model uncertainty by fitting an N-component Gaussian Mixture Model (GMM) to a dataset of state-action pairs, but only use a subset of the state features. We denote the resulting joint distribution as $p_{\mathcal{G}}(s_r, a)$, where $s_r = s|_{\mathcal{I}}$ is the restriction of s to a selected index set $\mathcal{I} \subseteq \{1, \ldots, \dim(s)\}$. To calculate an exploratory action while in state s, we resample a component-reweighted version of $p_{\mathcal{G}}$ according to the likelihood of the current state and proposed action pair. Further details can be found in Algorithm 2.

Baselines: We compare our **single-step** exploration against two common-sense exploration frameworks: 1) **greedy**, which uses $\pi^*_{\hat{M}_{T_1}}$ to explore in $M_{T_{1,2}}$, as is done in Matsushima et al. (2020), and 2) **epsilon-greedy**, which takes uniformly random action with probability ϵ and otherwise use $\pi^*_{\hat{M}_{T_1}}$.

Results: Figure 2 shows a comparison of the return achieved by various policies evaluated in M_{T_3} in terms of their relative performance. The policies being compared are trained in three distinct versions of $\hat{M}_{1,2}$, each constructed from data collected in $M_{1,2}$ by the greedy, epsilon-greedy, and our single-step exploration policies. We also compare against a policy that was only trained in $\hat{M}_1$ (Base). Firstly, we confirm that exploration is necessary by seeing that the Base policy under-performs all other methods (left column). Further, we see that the our single-step exploration method outperforms the other three policies to a statistically significant degree (top row). Importantly, the greedy policy does not sufficiently cover the state space to learn the most robust policy in the next round of deployment.

5 Related Work

Offline to Online finetuning: Offline RL algorithms are essentially off-policy RL algorithms that add additional constraints to the algorithm to ensure that the policy always stays within the data support. A number of such algorithms explicitly constrain the value functions (Kumar et al., 2020; Yu et al., 2020; Kidambi et al., 2021) or use regularization to ensure that the policy stays within the data support (Wu et al., 2019; Nachum et al., 2019). In all cases, the performance of the policy is restricted by the dataset quality (Kidambi et al., 2021).

Online finetuning has been extensively used (Nair et al., 2020; Mark et al., 2024; Ball et al., 2023; Wang et al., 2023) to mitigate the dataset bias. While the offline RL algorithms themselves can be directly used for online finetuning like in Peng et al. (2019a); Nair et al. (2020); Kumar et al. (2020); Kostrikov et al. (2021), they are overly pessimistic. A number of works have been introduced that simply reduce the regularization for conservatism during online finetuning (Lee et al., 2021; Zhang et al., 2023; Hong et al., 2023; Ball et al., 2023; Zheng et al., 2023). There have also been works that have used offline datasets as a prior for online RL algorithms (Mark et al., 2024; Ball et al., 2023).

All these methods have one major assumption: there is no restriction on online finetuning. The focus of all these works is to produce good policy updates combining the online samples and offline dataset. As discussed in works like Matsushima et al. (2020), there is often costs associated before deploying any policy in the real world. Our work differs from all these offline-to-online methods in that it aims to develop an exploration scheme suitable for applications where it is not possible to deploy arbitrary policies.

Exploration: Our method constructs an exploratory policy from batched experience to deploy in the real world. Here, we will discuss some commonly used methods for constructing exploratory policies. Epsilon greedy (Watkins, 1989; Auer et al., 2002) is a simple, yet highly effective method used to take exploratory actions with probability ϵ. The idea is to ensure that the policy has some entropy and chooses every action with a non-zero probability. Another way of increasing the entropy of the policy is through explicit entropy maximization as in the MaxEnt RL methods (Haarnoja et al., 2017; 2018).

Selecting actions using an upper confidence estimator for the average return (Q-function in case of RL algorithms) have been inspired from the UCB algorithm in multi-arm bandits (Auer et al., 2002). Strehl & Littman (2008) proposed using an uncertainty bonus for the reward function providing convergence guarantees. Algorithms have been proposed that extend the uncertainty bonus in several ways: using pseudocounts (Bellemare et al., 2016; Lobel et al., 2023), model errors (Burda et al., 2019) and using ensembles to estimate uncertainty (Kidambi et al., 2021). Distribution matching approaches have also been proposed that aim to increase the entropy of the state-visitation distribution (Lee et al., 2019; Agarwal et al., 2024). While our exploration method also draws from a similar objective, these exploration strategies have been proposed in settings where the aim is to explore all states, ultimately leading to a nearly zero uncertainty bonus for all states. Our method, on the other hand, uses the uncertainty bonus to select a few states from the prior policy distribution.

6 Conclusion

In many real-world settings, policies cannot be continuously updated online; we proposed exploration strategies for such domains. We adopted deployment efficiency as a relevant evaluation metric, and showed how our algorithms lead to high return and deployment efficiency. We constructed two exploration policies (single-step and multi-step) by combining a model of the uncertainty of a dataset along with the optimal policy that can be extracted from it. Along with theoretical bounds on the sub-optimality of these policies, we provide extensive experiments in the Nav-Chambers environment and a well-calibrated simulator of a real-world supply chain environment. Our proposed methods balance safety with exploration and show that it is possible to maintain performance while exploring in the deployment efficient regime.

References

Joshua Achiam and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning, 2019. URL https://api.semanticscholar.org/CorpusID:208283920.

Siddhant Agarwal, Ishan Durugkar, Peter Stone, and Amy Zhang. f-policy gradients: A general framework for goal-conditioned rl using f-divergences. *Advances in Neural Information Processing Systems*, 36, 2024.

Guy Amir, Michael Schapira, and Guy Katz. Towards scalable verification of rl-driven systems. *CoRR*, abs/2105.11931, 2021. URL https://arxiv.org/abs/2105.11931.

Sohrab Andaz, Carson Eisenach, Dhruv Madeka, Kari Torkkola, Randy Jia, Dean Foster, and Sham Kakade. Learning an inventory control policy with general inventory arrival dynamics. *arXiv preprint arXiv:2310.17168*, 2023.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2–3):235–256, may 2002. ISSN 0885-6125. DOI: 10.1023/A:1013689704352. URL https://doi.org/10.1023/A:1013689704352.

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data, 2023. URL https://arxiv.org/abs/2302.02948.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in Neural Information Processing Systems*, 29, 2016.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *Seventh International Conference on Learning Representations*, pp. 1–17, 2019.

Davide Corsi, Guy Amir, Andoni Rodriguez, Cesar Sanchez, Guy Katz, and Roy Fox. Verification-guided shielding for deep reinforcement learning, 2024. URL https://arxiv.org/abs/2406.06507.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(18):503–556, 2005. URL http://jmlr.org/papers/v6/ernst05a.html.

Omer Gottesman, Fredrik Johansson, Matthieu Komorowski, Aldo Faisal, David Sontag, Finale Doshi-Velez, and Leo Anthony Celi. Guidelines for reinforcement learning in healthcare. *Nature medicine*, 25(1):16–18, 2019.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. *CoRR*, abs/1702.08165, 2017. URL http://arxiv.org/abs/1702.08165.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018. URL http://arxiv.org/abs/1801.01290.

Hado Hasselt. Double q-learning. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta (eds.), *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010. URL https://proceedings.neurips.cc/paper_files/paper/2010/file/091d584fced301b442654dd8c23b3fc9-Paper.pdf.

Joey Hong, Aviral Kumar, and Sergey Levine. Confidence-conditioned value functions for offline reinforcement learning, 2023. URL https://arxiv.org/abs/2212.04607.

Xuemin Hu, Pan Chen, Yijun Wen, Bo Tang, and Long Chen. Long and short-term constraints driven safe reinforcement learning for autonomous driving, 2024.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel : Model-based offline reinforcement learning, 2021. URL https://arxiv.org/abs/2005.05951.

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE transactions on intelligent transportation systems*, 23(6):4909–4926, 2021.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning, 2021. URL https://arxiv.org/abs/2110.06169.

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

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *CoRR*, abs/2006.04779, 2020. URL https://arxiv.org/abs/2006.04779.

Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric P. Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *CoRR*, abs/1906.05274, 2019. URL http://arxiv.org/abs/1906.05274.

Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. *CoRR*, abs/2107.00591, 2021. URL https://arxiv.org/abs/2107.00591.

Zuxin Liu, Zijian Guo, Haohong Lin, Yihang Yao, Jiacheng Zhu, Zhepeng Cen, Hanjiang Hu, Wenhao Yu, Tingnan Zhang, Jie Tan, and Ding Zhao. Datasets and benchmarks for offline safe reinforcement learning, 2023.

Sam Lobel, Akhil Bagaria, and George Konidaris. Flipping coins to estimate pseudocounts for exploration in reinforcement learning, 2023. URL https://arxiv.org/abs/2306.03186.

Yecheng Ma, Andrew Shen, Dinesh Jayaraman, and Osbert Bastani. Versatile offline imitation from observations and examples via regularized state-occupancy matching. In *International Conference on Machine Learning*, pp. 14639–14663. PMLR, 2022.

Dhruv Madeka, Kari Torkkola, Carson Eisenach, Anna Luo, Dean P. Foster, and Sham M. Kakade. Deep inventory management, 2022. URL https://arxiv.org/abs/2210.03137.

Liyuan Mao, Haoran Xu, Weinan Zhang, and Xianyuan Zhan. Odice: Revealing the mystery of distribution correction estimation via orthogonal-gradient update. In *The Twelfth International Conference on Learning Representations*, 2024.

Max Sobol Mark, Archit Sharma, Fahim Tajwar, Rafael Rafailov, Sergey Levine, and Chelsea Finn. Offline RL for online RL: Decoupled policy learning for mitigating exploration bias, 2024. URL https://openreview.net/forum?id=lWe3GBRem8.

Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization. *CoRR*, abs/2006.03647, 2020. URL https://arxiv.org/abs/2006.03647.

Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. *Advances in neural information processing systems*, 32, 2019.

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

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *CoRR*, abs/1910.00177, 2019a. URL http://arxiv.org/abs/1910.00177.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning, 2019b. URL https://arxiv.org/abs/1910.00177.

Martin L Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

Martin Riedmiller. Neural fitted q iteration – first experiences with a data efficient neural reinforcement learning method. In João Gama, Rui Camacho, Pavel B. Brazdil, Alípio Mário Jorge, and Luís Torgo (eds.), *Machine Learning: ECML 2005*, pp. 317–328, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31692-3.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. No-regret reductions for imitation learning and structured prediction. *CoRR*, abs/1011.0686, 2010. URL http://arxiv.org/abs/1011.0686.

Yutaka Shimizu, Joey Hong, Sergey Levine, and Masayoshi Tomizuka. Strategically conservative q-learning. *arXiv preprint arXiv:2406.04534*, 2024.

Harshit Sikchi, Qinqing Zheng, Amy Zhang, and Scott Niekum. Dual rl: Unification and new methods for reinforcement and imitation learning. In *The Twelfth International Conference on Learning Representations*, 2023.

Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

Shenzhi Wang, Qisen Yang, Jiawei Gao, Matthieu Gaetan Lin, Hao Chen, Liwei Wu, Ning Jia, Shiji Song, and Gao Huang. Train once, get a family: State-adaptive balances for offline-to-online reinforcement learning, 2023. URL https://arxiv.org/abs/2310.17966.

Christopher John Cornish Hellaby Watkins. Learning from delayed rewards, 1989.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *CoRR*, abs/1911.11361, 2019. URL http://arxiv.org/abs/1911.11361.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: model-based offline policy optimization. *CoRR*, abs/2005.13239, 2020. URL https://arxiv.org/abs/2005.13239.

Haichao Zhang, We Xu, and Haonan Yu. Policy expansion for bridging offline-to-online reinforcement learning, 2023. URL https://arxiv.org/abs/2302.00935.

Han Zheng, Xufang Luo, Pengfei Wei, Xuan Song, Dongsheng Li, and Jing Jiang. Adaptive policy learning for offline-to-online reinforcement learning, 2023. URL https://arxiv.org/abs/2303.07693.

Zhiyuan Zhou, Andy Peng, Qiyang Li, Sergey Levine, and Aviral Kumar. Efficient online reinforcement learning fine-tuning should not retain offline data. In *The Thirteenth International Conference on Learning Representations*, 2025.

Appendix

A Offline RL and Pessimism

Offline RL refers to the class of RL algorithms that train on a fixed dataset. Due to the inherent overestimation biases (Hasselt, 2010) in RL algorithms, naively trained policies have a tendency to steer out of the dataset towards the highly-biased value estimates. As a result, regularizations are added to the ensure pessimism in the value functions. These regularizations can be behavior based (Kumar et al., 2019; Wu et al., 2019), value based (Kumar et al., 2020), visitation based (Sikchi et al., 2023) or reward based (Kidambi et al., 2021; Yu et al., 2020). Recent works have shown several interconnections among these (Sikchi et al., 2023; Mao et al., 2024). We consider the following constraint optimization as offline RL optimization problem:

$$\begin{aligned} \max_{\pi} \quad & J(\pi, \mathcal{M}) \\ s.t. \quad & D(p^{\pi}(s,a)||\hat{p}(s,a)) \leq \epsilon, \end{aligned} \tag{6}$$

Equation 6 has been studied through common offline RL and imitation learning algorithms (Nachum et al., 2019; Sikchi et al., 2023). We approximate the optimization problem using a pessimistic MDP (as defined in Kidambi et al. (2021)). The pessimistic MDP ensures that $D(p^{\pi}(s,a)||\hat{p}(s,a)) \leq \epsilon$ is always satisfied as any transition outside the dataset is heavily penalized.

A.1 Proof of Theorem 3.4

We will be introducing some definitions and lemmas in order to prove Theorem 3.4. We begin by defining hitting time,

Definition A.1. *(Hitting time)* Given an MDP $\mathcal{M}$, starting state distribution ρ_0, state-action pair (s,a) and a policy π, the hitting time $T^{\pi}_{(s,a)}$ is defined as the random variable denoting the first time action a is taken at state s by π on $\mathcal{M}$, and is equal to ∞ if a is never taken by π from state s. For a set of state-action pairs $S \subseteq S \times A$, we define $T^{\pi}_{S} \stackrel{\text{def}}{=} \min_{(s,a)\in S} T^{\pi}_{(s,a)}$. In other words, it is the time it takes to arrive at state s.

Using the definition, we can introduce the following Lemma that bounds the returns obtained in the pessimistic MDP $\mathcal{M}_{\mathcal{D}}$ using the returns in the true MDP $\mathcal{M}$.

Lemma A.2. *Let $\mathcal{M}$ be a given MDP. The following is true for any offline dataset $\mathcal{D}$.*

$$J(\pi, \mathcal{M}) - \frac{2R_{max}}{1-\gamma}\mathbb{E}[\gamma^{T^{\pi}_{\mathcal{U}_{\mathcal{D}}}}] \leq J(\pi, \mathcal{M}_{\mathcal{D}}) \leq J(\pi, \mathcal{M}) \tag{7}$$

Proof. A more extensive theorem and corresponding proof exists in Kidambi et al. (2021). The definition of pessimistic-MDP in the two works is slightly different leading to different bounds.

The rollout of any policy π on the pessimistic MDP $\mathcal{M}_{\mathcal{D}}$ would be the same as that of the true MDP $\mathcal{M}$ as long as an unknown state ($s \in \mathcal{U}_{\mathcal{D}}$) is encountered. At that point, the return of the policy on the pessimistic MDP will be $\frac{-R_{max}}{1-\gamma}$. The maximum return of the policy for that rollout segment can be $\frac{R_{max}}{1-\gamma}$. Hence,

$$J(\pi, \mathcal{M}) - \frac{2R_{max}}{1-\gamma}\mathbb{E}[\gamma^{T^{\pi}_{\mathcal{U}_{\mathcal{D}}}}] \leq J(\pi, \mathcal{M}_{\mathcal{D}}) \tag{8}$$

Since $\mathcal{M}_{\mathcal{D}}$ is a pessimistic MDP for the corresponding MDP $\mathcal{M}$ with each state in $\mathcal{M}_{\mathcal{D}}$ having rewards less than or equal to $\mathcal{M}$, for any policy, the return of will be higher in $\mathcal{M}$ compared to $\mathcal{M}_{\mathcal{D}}$ i.e. $J(\pi, \mathcal{M}_{\mathcal{D}}) \leq J(\pi, \mathcal{M})$. □

With this bound, we can compare the returns obtained by policies trained on the pessimistic MDP with the corresponding returns in true MDP. To simplify this comparison, we use the following Lemma relates hitting times to visitation distributions.

Lemma A.3. *(Kidambi et al., 2021) (Hitting time and visitation distributions) For any set $\mathcal{X} \subseteq \mathcal{S} \times \mathcal{A}$, and any policy π, we have $\mathbb{E}[\gamma^{T^{\pi}_{\mathcal{X}}}] \leq \frac{1}{1-\gamma} d^{\pi}(\mathcal{X})$.*

Now, we have all the components required to prove Theorem 3.4.

Theorem 3.4. *Let $\mathcal{M}$ be a given MDP and $\mathcal{D}$ be the offline dataset. Let $\pi^* = \arg\max_{\pi} J(\pi, \mathcal{M})$ be the true optimal policy on $\mathcal{M}$ and $\pi^*_D = \arg\max_{\pi}$ Equation (1) $\approx \arg\max_{\pi} J(\pi, \mathcal{M}_{\mathcal{D}})$ be the best offline RL policy on dataset $\mathcal{D}$. The following holds: $J(\pi^*, \mathcal{M}) - J(\pi^*_D, \mathcal{M}) \leq \frac{2R_{max}}{(1-\gamma)^2} d^{\pi^*}(\mathcal{U}_{\mathcal{D}})$.*

Proof. From Lemma A.2,

$$J(\pi, \mathcal{M}) - \frac{2R_{\max}}{1-\gamma}\mathbb{E}[\gamma^{T^{\pi}_{\mathcal{U}_{\mathcal{D}}}}] \leq J(\pi, \mathcal{M}_{\mathcal{D}}) \leq J(\pi, \mathcal{M})$$
$$J(\pi^*, \mathcal{M}) - J(\pi^*, \mathcal{M}_{\mathcal{D}}) \leq \frac{2R_{\max}}{1-\gamma}\mathbb{E}[\gamma^{T^{\pi^*}_{\mathcal{U}_{\mathcal{D}}}}]$$

From Lemma A.3, we replace the upper bound

$$J(\pi^*, \mathcal{M}) - J(\pi^*, \mathcal{M}_{\mathcal{D}}) \leq \frac{2R_{\max}}{(1-\gamma)^2} d^{\pi}(\mathcal{U}_{\mathcal{D}})$$

By definition, we know that $J(\pi^*, \mathcal{M}_{\mathcal{D}}) \leq J(\pi^*_{\mathcal{D}}, \mathcal{M}_{\mathcal{D}}) \leq J(\pi^*_{\mathcal{D}}, \mathcal{M})$. Thus we substitute

$$J(\pi^*, \mathcal{M}) - J(\pi^*_{\mathcal{D}}, \mathcal{M}) \leq \frac{2R_{\max}}{(1-\gamma)^2} d^{\pi}(\mathcal{U}_{\mathcal{D}})$$

□

B Connections to Commonly Used Exploration

As discussed above, defining $u(s, a)$ is a major design decision that needs to be taken and there is no optimal way to define uncertainty. We can show that different heuristic definitions of $u(s, a)$ and π^*_D can draw connections to commonly used exploration techniques.

ϵ-greedy: Suppose $u(s, a)$ is not learned but defined using a crude heuristic: For any state s, with probability $1 - \epsilon$, the action is not uncertain i.e. $\exp \frac{1}{\beta} u(s, a) = 1$ and with probability ϵ, the action is uncertain with inversely depending on π^*_D i.e. $\exp \frac{1}{\beta} u(s, a) = \frac{c}{\pi^*_D(a|s)}$. Then, the corresponding exploration turns out to be ϵ-greedy. What does the uncertainty mean here? The value $\frac{c}{\pi^*_D(a|s)}$ means that the uncertainty is lower for policy actions which makes some sense as the actions taken by the policy are already known and present in the dataset.

Bandit-style UCB: Bandit-style UCB produces a policy that takes action according to $Q(s, a) + u(s, a)$ where $u(s, a)$ is of a specific function depending on the frequency of actions. UCB is very efficient exploration method as it provides a logarithmic regret bound in bandits. If π^*_D is assumed to be a softmax of the offline Q function i.e. $\pi^*_D \propto \exp Q(s, a)$, the exploration policy π^*_{exp} becomes a soft version of the Bandit-style UCB. It must be noted though, for most offline RL algorithms, π^*_D is not $\propto \exp Q(s, a)$ (Kostrikov et al., 2021; Sikchi et al., 2023).

Uncertainty aware reward bonus: A common way of exploring in RL is to use uncertainty based reward bonus. This method is inspired from UCB and is formalized better in Strehl & Littman (2008). Here the Q function are trained to be optimistic by adding $u(s, a)$ to the reward function. Mathematically, $Q^{\pi}(s, a) = \mathbb{E}_{\pi}[\sum_t \gamma^t (r(s_t, a_t) + c \cdot u(s_t, a_t))]$. These algorithms are online algorithms converging to optimal Q functions as $u(s, a) \to 0$.

C Additional Results

C.1 Reset Starting State

Figure 3 contains the results for using the *reset* setting to explore on the Chamber Navigation environment.

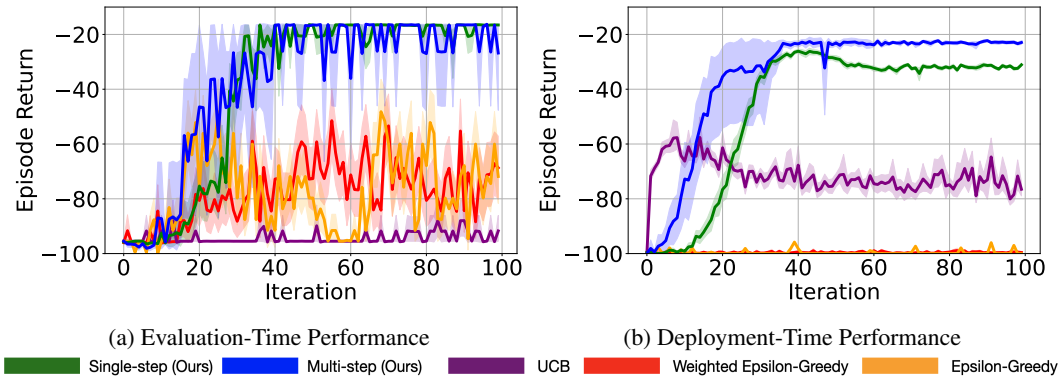


Figure 3: Comparison of our exploration methods against baselines while taking exploration steps beginning at each environmental `reset`.

D Experimental Setup

D.1 Supply Chain Environment

We use the supply chain environment described in detail in (Madeka et al., 2022; Andaz et al., 2023). Here the RL agent makes decisions about buying items from vendors. The agent takes in a state x and its history of length H to predict the next action. The state consists of several features including demand, inventory, cost, time of year, etc. The authors of Madeka et al. (2022) introduced an algorithm to solve this domain and use real-world data by assuming the environmental setup to be an Exogenous-MDP. A simulator is created with the learned dynamics models for the endogenous components and the exogenous components are sampled from the real-world data. Because of the exogenous assumption, this simulator allows for back-testing of the policies and is calibrated with the real world. An RL agent is trained in the simulator using the DirectBackprop algorithm (Madeka et al., 2022).

While our method is designed for real-world exploration, it is infeasible to test in the real world due to time-constraints (each step in the simulator represents one week). To overcome this, we partition our data into three segments, T_1, T_2, and T_3 that represent train, explore and test periods, respectively. Further, we construct a reduced fidelity versions of the data by clamping some of the extreme (but highly rewarding) features; this allows us to generate simulators that mimics low data coverage.

D.2 Nav-Chamber Experiments

The Nav-Chambers is a gridworld-like environment where the action space is [`Up`, `Down`, `Left`, `Right`] and the reward is -1 everywhere except the goal, where it is $+1$. In the Nav-Chambers environment, we implement deployment efficient exploration using count-based uncertainty estimation and train extract a policy using fitted Q-iteration, as described in Algorithm 1. The results shown use 100 deployments, each with 1000 train steps and 1000 collect steps.

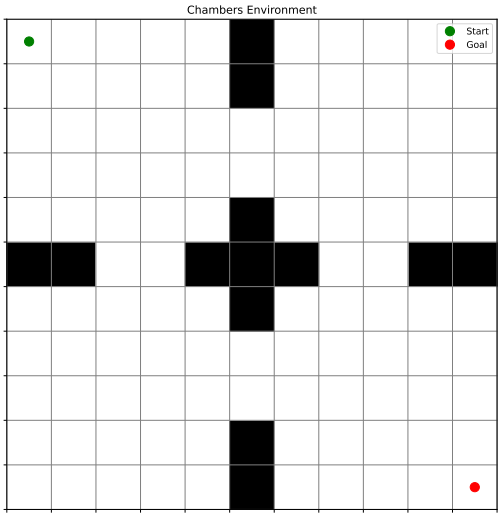


Figure 4: Obstacle map for the Nav-Chambers environment.

Algorithm 1 Deployment Efficient Exploration with Fitted Q-Iteration

Initialize dataset of interactions $\mathcal{D} \leftarrow \{\}$, initialize Q-function $Q_\theta(s, a)$
for $n = 1$ to N **do**
  for $k = 1$ to K **do**
    Sample s from environment or replay buffer
    Sample action $a \sim \pi^{\text{exploration}}(a \mid s)$
    Execute a, observe r, s'
    Add transition to dataset: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s, a, r, s')\}$
  end for
  Fit Q_θ to minimize Bellman loss on $\mathcal{D}$:

$$\mathcal{L}(\theta) = \sum_{(s,a,r,s') \in \mathcal{D}} \left(Q_\theta(s, a) - \left[r + \gamma \max_{a'} Q_{\theta'}(s', a') \right] \right)^2$$

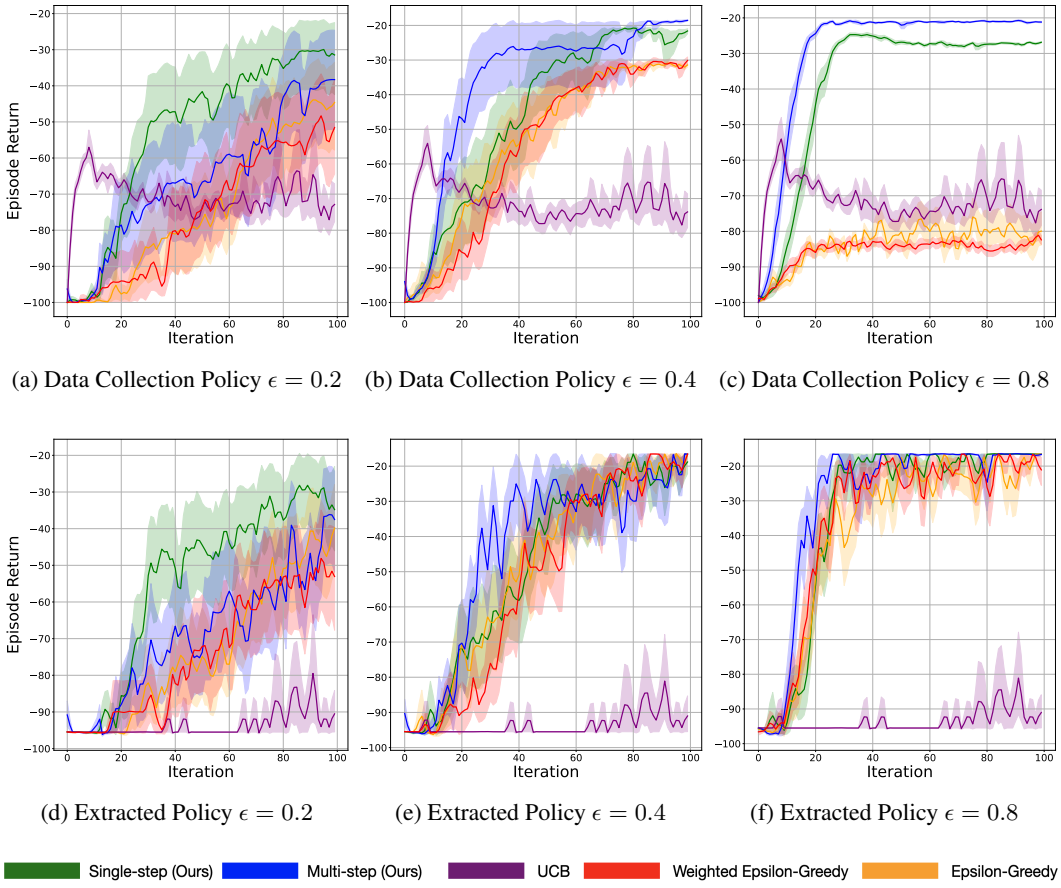
  Optionally update target network $\theta' \leftarrow \theta$
end for
return final Q-function Q_θ

Across the experiments, the exploration policy is deployed a total of 100 times, collecting 1000 transitions and training the policy for 1000 gradient steps. After each deployment, we use fitted

E Ablation

E.1 Epsilon

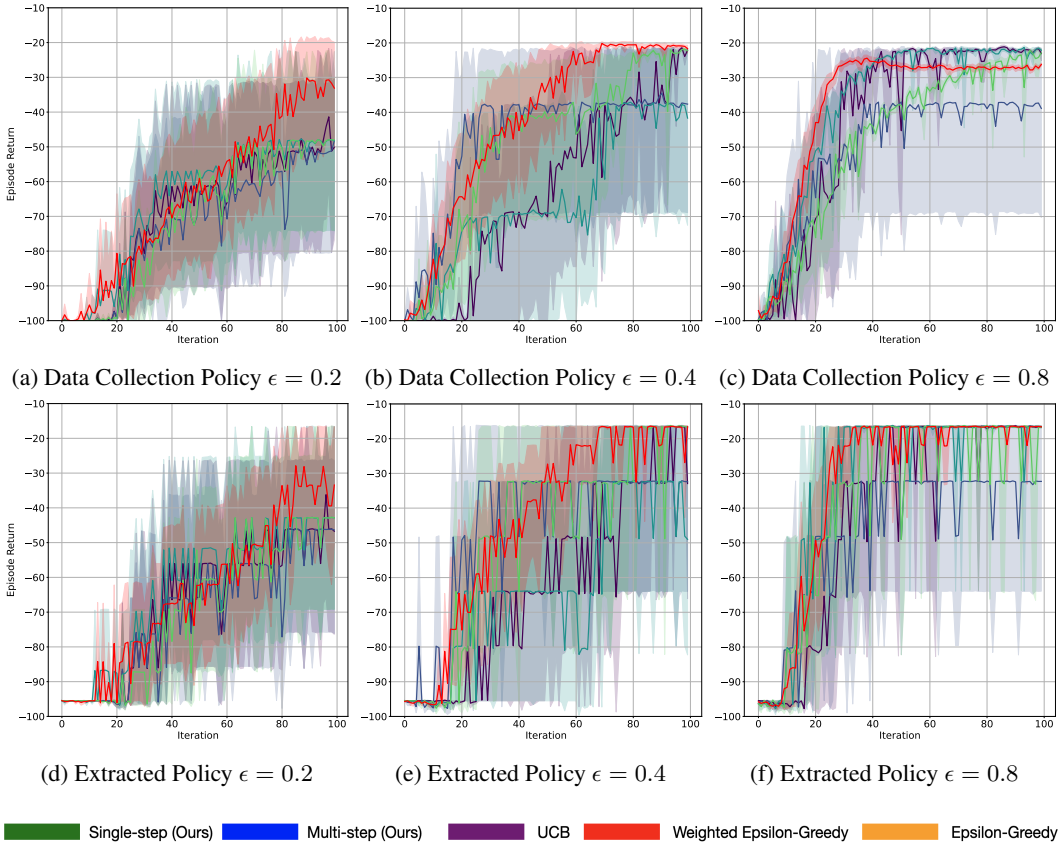
We ablate over different values of epsilon to demonstrate the importance of uncertainty across a variety of exploration budgets. Figure 5 shows the **eval-returns** (bottom row) and **collect-returns** (top row) of our method and various baselines across different epsilons. These results show the ϵ-exploration setting. Finally, from left to right, the value of ϵ increases. We see relative performance improvements on the **collect-returns** of our single- and multi-step methods as ϵ increases.



(a) Data Collection Policy $\epsilon = 0.2$ (b) Data Collection Policy $\epsilon = 0.4$ (c) Data Collection Policy $\epsilon = 0.8$

(d) Extracted Policy $\epsilon = 0.2$ (e) Extracted Policy $\epsilon = 0.4$ (f) Extracted Policy $\epsilon = 0.8$

Figure 5: Comparison of our exploration policies (Single-Step and Multi-Step) against the baselines using **collect-returns** (top row) and **eval-returns** (bottom row). Each curve shows a mean of 5 seeds with a rolling average of window length 5 and the shaded region represents the 95% confidence interval. From left to right, each column represents a different value of epsilon (ϵ used for the ϵ-exploration across methods.)



(a) Data Collection Policy $\epsilon = 0.2$ (b) Data Collection Policy $\epsilon = 0.4$ (c) Data Collection Policy $\epsilon = 0.8$

(d) Extracted Policy $\epsilon = 0.2$ (e) Extracted Policy $\epsilon = 0.4$ (f) Extracted Policy $\epsilon = 0.8$

Figure 6: Comparison of our exploration policies (Single-Step and Multi-Step) against the baselines using **collect-returns** (top row) and **eval-returns** (bottom row). Each curve shows a mean of 5 seeds with the shaded region as the 95% confidence interval. From left to right, each column represents a different value of epsilon (ϵ used for the ϵ-exploration across methods.

E.2 Annealed epsilon greedy experiment

One advantage of our single- and multi-step exploration methods is that they achieve high performance when in the collection phase of the deployment without sacrificing data quality for the policy extraction phase. We hypothesize that this advantage stems from the methods' ability to automatically converge to the exploiting policy once enough exploration has occurred. To test this, we compare the single-step exploration method to a linearly-decaying epsilon-greedy exploration. Define annealed epsilon-greedy as:

$$\pi_{\text{Annealed } \epsilon-\text{greedy}}(a|s,t) = \begin{cases} \arg\max Q(s,a) & \text{with probability } (1 - f(t)) \\ \mathcal{U}(\mathcal{A}) & \text{with probability } f(t) \end{cases} \tag{9}$$

where $f(t) \in [0, 1)$ is some scalar-valued function that takes as input a notion of time (usually how many steps of experience have been gained at that point) and outputs an exploration parameter ϵ. Typically, $f(t)$ is a linearly decays from a value of ϵ_{start} to ϵ_{end} over a period of time. Figure 6 demonstrates potential decays over a training period of 100k steps.

F Uncertainty-aware Exploration Policy in Supply Chain Environment

Because the Supply Chain Environment has high dimensional state and action spaces, we approximate uncertainty with a N-component Gaussian mixture model (GMM), and then construct an uncertainty-aware exploration policy by reweighting the model components at explore-time. The Gaussian

Algorithm 2 Uncertainty-aware Action Resampling

Given:
- Policy π_M
- Fitted GMM $p_{\mathcal{G}}(s_r, a)$
- Temperature parameter $\tau > 0$
- Input state s, with relevant feature subset $s_r = s|_{\mathcal{I}}$

Compute action: $a \leftarrow \pi_M(s)$
Evaluate state-action likelihood: $p_{\mathcal{G}}(s_r, \pi_M(s_r))$
Extract GMM component contributions: $\{w_i\}_{i=1}^K$
Compute reweighted components:

$$w_i' \leftarrow \frac{\exp(w_i/\tau)}{\sum_{j=1}^K \exp(w_j/\tau)} \quad \text{for } i = 1, \ldots, K$$

Construct reweighted GMM $p'_{\mathcal{G}}$ using $\{w_i'\}$
Sample new action: $a' \sim p'_{\mathcal{G}}(\cdot \mid s_r)$

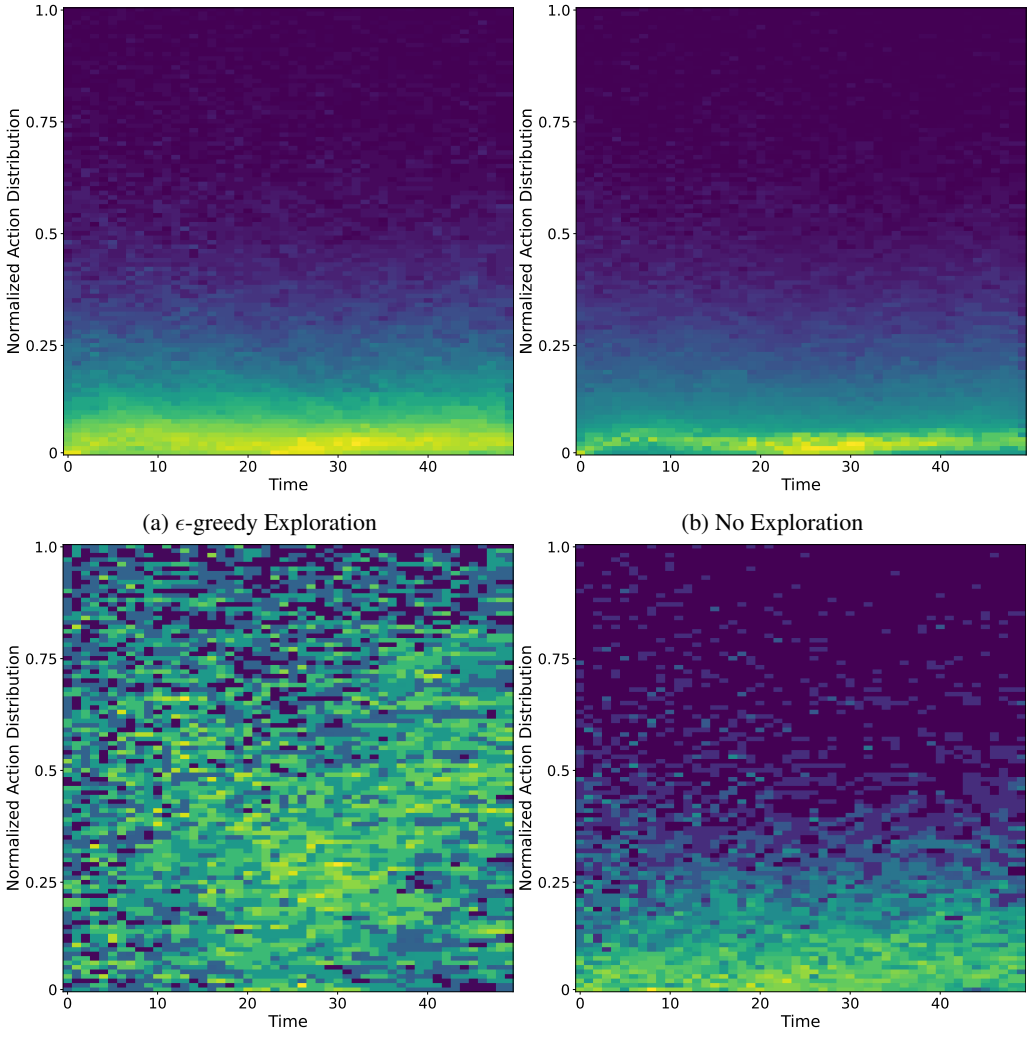
mixture model $p_{\mathcal{G}}$ is learned to approximate the joint distribution of s_r and a present in the dataset, where s_r is defined in Section 4.2. To take an uncertainty-maximizing action in the Supply Chain Environment, the agent first calculates its deterministic nominal action $a = \pi(s)$. Next, calculate the likelihood of the proposed state-action pair with respect to $p_{\mathcal{G}}$ and determine contributions $\mathbf{w}$ of each of the N component Gaussians. Lastly, resample the action a from the conditional GMM $p'_{\mathcal{G}}(a|s_r)$ but reweigh the components using a higher temperature distribution based on the component contributions $\mathbf{w}$ where

$$w_i' = \frac{\exp(w_i/\tau)}{\sum_i \exp(w_i/\tau)}. \tag{10}$$

The procedure is outlined in detail in Algorithm 2.

G Exploration Trajectories in the Supply Chain Environment

In the Supply Chain Environment, we have compared three different exploration strategies in Section 4.2. We have observed that our exploration policy does better than greedy and ϵ-greedy exploration policies. While it does makes sense as our strategy is theoretically better than the naive exploration strategies. But we investigate how different these explorations are qualitatively. It is not trivial to directly visualize the policies in the Supply Chain Environment. We plot the distribution of actions taken for every timestep of the trajectory to visualize the exploratory policies. Figure 7 shows these plots of four policies: a heuristic policy (one that takes actions to maintain static inventory levels), a greedy policy, an ϵ-greedy policy and our exploratory policy. It can be seen that while the heuristic policy is totally random across the entire space, the ϵ-greedy stays very close to the greedy policy. Our exploration on the other hand is not as random (and unsafe) as the heuristic policy but still explores farther than the ϵ-greedy.



(a) ϵ-greedy Exploration

(b) No Exploration

(c) Heuristic Exploration

(d) Our Exploration

Figure 7: This figure shows the distribution of normalized actions at each time step in the trajectory for thousands of trajectories. The bright regions denote higher density.