A Framework for Safe Evaluation of Offline Learning

by

Hager Radi

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science
University of Alberta

© Hager Radi, 2022

Abstract

The world offers unprecedented amounts of data in real-world domains, from which we can develop successful decision-making systems. It is possible for reinforcement learning (RL) to learn control policies offline from such data but challenging to deploy an agent during learning in safety-critical domains. Although an environment is essential, offline RL learns from historical data without access to an environment. Therefore, it is essential to find a technique for estimating how a newly-learned agent will perform when deployed in the real environment before actually deploying it. For instance, in medical domains, we cannot afford to deploy a bad policy. Moreover, if data is costly, we would like to know how much data is needed to learn a good enough policy so that we stop paying for data and let the new policy take over.

To achieve this, we introduce a framework for safe evaluation of offline learning using approximate high-confidence off-policy evaluation (HCOPE). We focus on safety so that the probability that our agent performs below a baseline is approximately δ, where δ specifies how much risk is reasonable. In our setting, we assume access to data, which we split into a train-set to learn an offline policy, and a test-set to estimate a lower-bound on the offline policy using off-policy evaluation with bootstrap confidence intervals. A lower-bound estimate allows us to decide when to deploy our learned policy with minimal risk of overestimation. We verify our proposed framework on a range of tasks as well as real-world medical data. Since current offline RL methods rely on some environment for evaluation, this thesis fills a real gap on how offline agents can be evaluated while learning given data only and with high confidence.

Preface

This thesis is an original work by Hager Radi. Part of this thesis has been previously presented at NeurIPS 2021 Workshop on Deployable Decision Making in Embodied Systems (DDM)

To my father, who left my world very early, this is one story of the millions I didn't get the chance to tell you.

Till we meet again..

Acknowledgements

My journey as a graduate student was really dense and it helped me grow on both professional and personal levels. First and foremost, I would like to thank my advisor, Matthew E. Taylor, for his support throughout this thesis. Other than the technical part, Matt is a very supportive and understanding human who always knows how to make things better for everyone. I was lucky to work with him and always inspired by his vision for solving real-world problems with RL. Secondly, I would like to thank Josiah P. Hanna, for advising and mentoring me even before he was officially a co-supervisor. I am grateful to them for helping me pursue this research and advance my knowledge. I would also like to thank Martha White, for being part of the thesis committee, being a great professor and a mentor during an early research project as well as TAing with her. Martha is a true inspiration for me and there is always more to learn from her. I would like to thank Peter Stone for his input and guidance for this work whenever needed. I would like to thank Rich Sutton for inspiring me during the first RL course I took with him, which boosted my confidence to pursue RL research. I would like to thank everyone of the IRL and RLAI lab whom i interacted with throughout my masters, specifically, Alex L, Manan, Sriram, Archit, Andy Patterson, Shivam, Sahir, Rohan and Fatemah.

Most importantly, I would like to thank my family, Mama, Reham and Rabab for their love and support, my bigger family, as well as my partner's family. I would like to thank my true friend, Hadeer, for being the real support through all the ups and downs over the past 8 years. Finally, I wouldn't have done all of this without my partner's support on all levels starting from submitting the application to UoA up

until this moment. Thank you Mo'men for always believing in me even when I lose it all.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Suppose someone else is controlling a sequential decision making task for you. This could be a person trading stocks for you, a hand-coded controller for a chemical plant, or even a PID for temperature regulation. Offline reinforcement learning (RL) allows us to learn policies from historical data. But when would one want to switch from the existing controller to the learned policy? This decision may depend on the cost for continued data collection from the existing controller (e.g., someone is getting paid to manage your stock portfolio), your risk appetite, and your confidence in the performance of the policy you have learned. This thesis takes a critical first step towards the following question: how to best learn offline and simultaneously evaluate that policy with confidence when we have no access to the environment nor the policy generating the data?

Offline reinforcement learning is a way to train off-policy algorithms using already existing data. Learning with offline data is challenging because of the distribution mismatch between the data collected by the behavior policy, that collected the data, and the offline agent, which we aim to learn from data [1]. What is even more challenging is evaluating offline agents in the offline RL setting if we assume no access to the environment; in some domains, we cannot execute our learned policy until it is good enough. This limitation raises the question about the possibility of using off-policy policy evaluation (OPE) methods, where we can estimate the value

of a policy using trajectories from another policy, to predict what the performance of the offline agent is, at any point of time during learning. Further, we investigate if we can use high-confidence off-policy evaluation (HCOPE) to control the risk of overestimating the policy's performance.

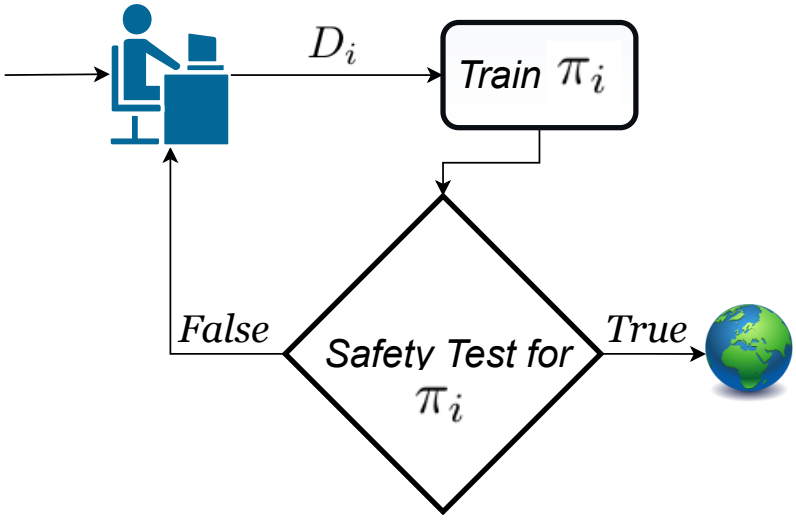


Figure 1.1: **A framework for continual safety-evaluation of offline RL**: This figure illustrates the loop of interaction between an offline agent and data while the agent is trained, evaluated and deployed if it passes the safety test.

We present a framework for safe evaluation of RL agents learning offline. *Safe* refers to defining a lower bound on our policy value estimates, a point where we think the new policy is at least as good as it is in practice, which is important for safety-critical applications. The thesis tackles the setup where we combine offline policy learning with high confidence off-policy evaluation; we learn a policy from pre-existing data, and evaluate it to provide a confidence lower-bound estimate on its return. The setup we are tackling is under-studied in the literature when the new policy, that we learn offline, is not necessarily similar to the behavior policy. The goal is to learn a target policy purely out of a data buffer, and evaluate its performance while learning so that we can tell when it is ready for deployment. The framework is summarized in Figure 1.1: we have some source of data, from which we use samples. At each step, data is split into training and testing. After each offline training step, we evaluate

the policy using approximate[1] high-confidence off-policy evaluation (HCOPE). We continue the process of training/testing for a few iterations until the testing shows the policy can outperform the data with a confidence level δ. We dynamically receive samples, continuously perform RL updates, and continuously monitor a confidence interval on the changing policy until it reaches a sufficient level for deployment. We prefer high-confidence off-policy evaluation to off-policy evaluation (OPE) because OPE is prone to overestimation, which is problematic for safety-critical problems.

1.1 Motivation

Offline RL presents a great opportunity for learning data-driven policies without environment interaction, as we discuss in Section 2.2.1. In safety-critical applications such as in healthcare or autonomous driving, there is a large amount of data that we can use to learn RL policies and hence use for decision making [2]. This data is not infinite nor free, so there are domains where data efficiency is essential as the data collection process is either expensive or dangerous. If we want to learn policies in such domains, we need to find a way to tell how good the performance of a policy is before actually deploying it to the real-world environment. However, the execution of a new policy can be costly or dangerous if it performs worse than the current policy or controller. Hence, in this thesis, we focus on safety when evaluating offline agents such that the probability that the performance of our agent below a baseline is at most δ, where δ specifies how much risk is reasonable for a certain domain. This will allow us to identify when we can trust a policy to take control in the real world. Given access to trajectories generated by an unknown behavior policy, we assume an iterative setting where, at each iteration, we can either use another batch of trajectories to continue learning or deploy our own policy. Our objective is to only deploy our own policy if a $1 - \delta$ lower bound on its expected return is greater than

[1] We refer to HCOPE as approximate because bootstrapping lower bounds may have error rates larger than δ but provide a practical alternative to guaranteed bounds that are too loose to use.

the expected return of the unknown behavior policy (the data distribution).

1.2 Thesis Objectives

In this thesis, the proposed framework acts as a workflow for offline improvement and evaluation in safety-critical domains. The thesis **contributions** are summarized as:

- A framework combining offline RL and approximate high-confidence off-policy evaluation for both discrete and continuous control tasks.
- Studying the feasibility and challenges of offline evaluation without access to the environment or the behavior policy.
- Exploring the effect of offline improvement on HCOPE methods given constantly-improving target policy (as opposed to a fixed policy, as previously studied in the literature).
- Presenting a case study on the applicability of safe evaluation for medical data.

1.3 Thesis Outline

We start with a background in chapter two, discussing the preliminaries, literature review and, the gap in current research. In chapter three, we discuss the proposed framework and the methodologies used. Then, we talk about the experimental work and results in chapter 4 on the simulated environments. In chapter 5, we test our proposed framework on real-world data. We end the thesis by discussing the challenges, conclusions, and future work in chapter 6.

Chapter 2

Background

This chapter introduces the main preliminaries, a review of the current literature of offline reinforcement learning and high-confidence off-policy evaluation. We also discuss the gap in research that this thesis fills.

2.1 Preliminaries

In this section, we introduce reinforcement learning (RL), the Markov decision process (MDP), on-policy vs. off-policy RL, and the offline RL notation.

2.1.1 Reinforcement Learning

Reinforcement learning (RL) algorithms are designed to learn to obtain goals through the interaction between an agent and an environment. An agent follows a policy π to take an action a_t at any point of time given the current state of the environment s_t. By taking an action, an agent receives a reward r_t and moves to the next state s_{t+1}. The goal is to find a policy that maximizes the accumulated reward received over time.

2.1.2 Markov Decision Process

We define a control system by a Markov decision process or an MDP. An MDP is defined as $\langle \mathcal{S}, \mathcal{A}, P, r, \gamma, d_0 \rangle$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, P is a set of probabilities defining the transition from the current states to the next state,

and d_0 is the initial state distribution. An agent acts in an episodic environment by sampling actions from a policy π given state S_t.

A trajectory H of a non-fixed horizon L is a state-action history, $s_0, a_0, ..., s_{L-1}, a_{L-1}$ where $s_0 \sim d_0, a_t \sim \pi(.|s_t)$ and $s_{t+1} \sim P(.|s_t, a_t)$. The return of a trajectory $g(H)$ is the sum of its discounted rewards, $g(H) = \sum_{t=0}^{L-1} \gamma R(S_t, A_t)$. The value of a policy $v(\pi)$ is defined by the expected return over all the trajectories generated while following the policy, $v(\pi) = \mathbb{E}_{H\sim\pi}[g(H)]$.

2.1.3 On-policy and Off-policy RL

On-policy RL methods estimate the value of the policy π used for control, such that the current data is collected with the same policy [3]. In off-policy RL, there is one policy used to generate the data, called the behavior policy π_b. A different policy is being evaluated and improved, called the target policy π_θ [3]. Off-policy RL is important for exploration as an agent can follow a stochastic behavior policy to explore the environment while optimizing a deterministic target policy.

2.1.4 Offline RL

Offline reinforcement learning refers to RL algorithms learning from a fixed dataset $\mathcal{D}$ without environment interaction or further data collection. Such data is the transition tuples $\mathcal{D} = \langle s_t, a_t, r_t, s_{t+1}, \gamma_{t+1} \rangle$ resulting from historical agent-environment interaction. We learn a new policy π_θ, a target policy that is not necessarily similar to the behavior policy π_b. In this work, we assume we have access to a dataset $\mathcal{D} = \{H_1, H_2, ..., H_n\}$ as a set of n trajectories generated by an unknown behavior policy π_b.

2.2 Literature Review

The literature of off-policy RL algorithms learning from fixed datasets, without environment interaction, lies under offline RL. However, Offline RL is a relatively new

term that had been previously referred to as Batch RL. There are a few inconsistencies in how the methods are grouped and cited. In the literature, Batch RL is the traditional term for off-policy evaluation and improvement, where early offline RL work could be found. Below, we distinguish between the two camps in the literature. In a book for state-of-the-art RL [4], one chapter on batch RL discusses how we can identify batch RL algorithms. A batch RL algorithm is about the data perspective, whether we use immediate data or previously stored data. An offline vs. online RL algorithm is about the interaction between the agent and an environment. Figure 2.1 is our adaptation from the book terminologies to the current terms and algorithms in the literature. A batch RL algorithm relies on the agent storing and reusing data collected from the environment as the policy is being optimized. An offline RL algorithm does not allow an agent to interact with the environment or collect further data. Note that every offline RL algorithm is a batch RL algorithm; however, not every batch algorithm is an offline algorithm.

2.2.1 Offline Reinforcement Learning

Offline RL is the fully off-policy way of learning a policy from historical data for control, not just evaluation. Fully off-policy/offline means that the target policy has no access to the environment and only observes the data collected without having access to the behavior policy [5]. This setting is recently referred to as offline RL [1]. It is an approach for creating data-driven algorithms and reusing already collected data by training fully off-policy agents from scratch. Offline RL can be viewed as similar to supervised learning, where a fixed amount of data is available and we try to learn the best policy out of such data. This contrasts with normal off-policy methods relying on the growing batch concept where further data collection is possible through interacting with the environment [6].

For offline RL, a survey paper [1] categorizes offline RL algorithms into model-free methods and model-based methods. This thesis only considers model-free approaches

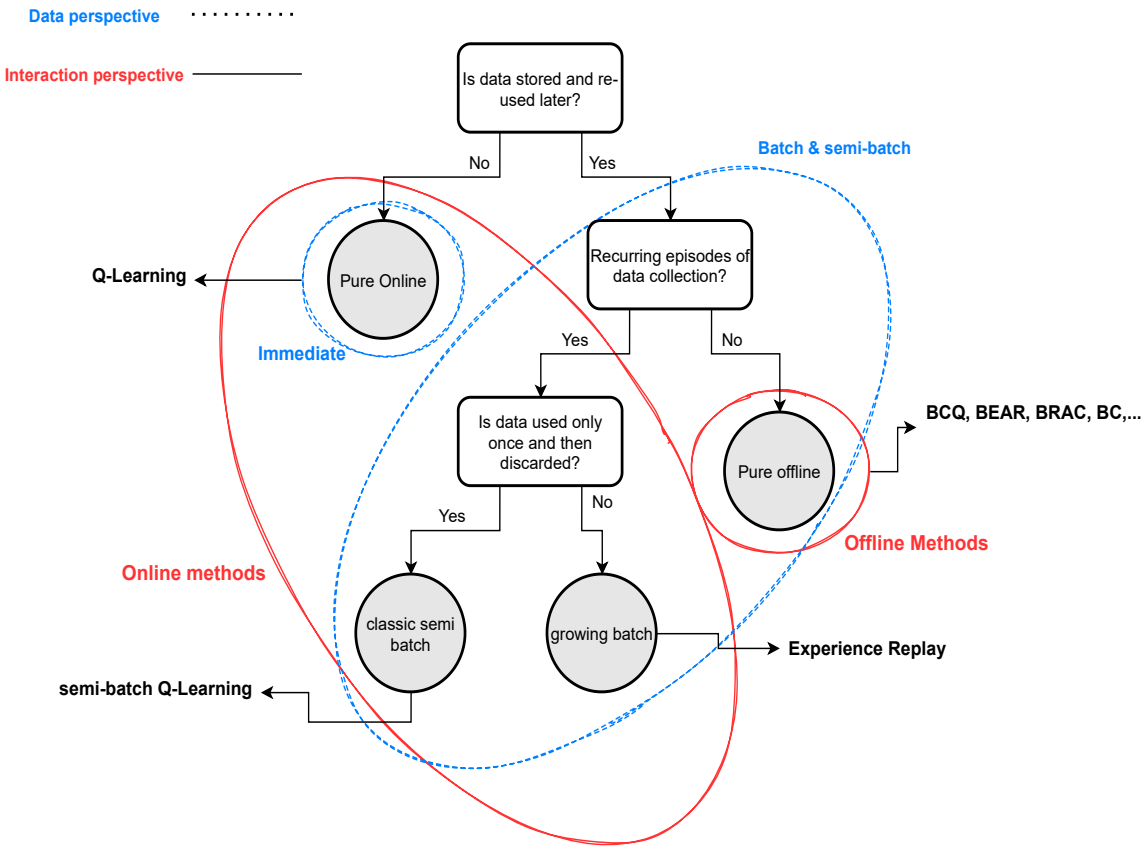


Figure 2.1: **Classification of offline vs. batch RL algorithms**: This figure shows how we can draw boundaries between online, batch, and offline RL algorithms as adapted from the state-of-the-art RL book []. Pure online RL algorithms do not store data for later use. Batch and semi-batch RL algorithms store data for reuse in future episodes of learning. Offline RL algorithms do not interact with the environment nor collect data.

for offline RL as they are more stable. Model-free methods are sub-categorized into policy constraints that constrain the learned policy to be close to the behavior policy and uncertainty-based methods that attempt to estimate the epistemic uncertainty of Q-values to reduce distributional shift. For policy gradient methods, a policy constraint can be enforced directly on the actor update to keep the new policy as similar as possible to the behavior policy while learning. This constraint can be implicit KL-divergence, which does not model the behavior policy such as the case in Advantage-weighted regression [7] or Advantage Weighted Actor-critic [8]. A constraint can also be an explicit divergence constraint that require an estimation for the behavior policy as done in the WOP algorithm [9]. In practice, policy constraint methods so far seem to outperform pure uncertainty-based methods, as shown with the batch constrained Deep Q-learning paper [6]. In Bootstrapping Error Accumulation Reduction (BEAR) [10], authors identify bootstrapping error as the source of instability of offline RL methods; they propose doing distribution-constrained backups via maximum-mean discrepancy (MMD) [10]. In Behavior-Regularized offline Reinforcement learning [11], a general framework, Behavior Constrained Actor-Critic (BRAC), is introduced to cover different ways of regularization to offline policies, whether as a value penalty or a policy penalty. Non-constrained methods can include the traditional Q-learning [12], Double DQN [13] or Soft-Actor Critic [14], which are not always successful in the fully offline setup. Behavioral cloning (BC) [15], as a main method for imitation learning, is another way for learning offline policies from historical data; BC proved to perform well compared to offline RL methods under medium-quality and expert data [10].

2.2.2 Batch Reinforcement Learning

Three main directions under the classical Batch RL research are off-policy policy evaluation (OPE) [16], high confidence off-policy policy evaluation (HCOPE) [17], and safe policy improvement (SPI) [18]. **Safe** refers to defining a lower bound to our

policy estimates, which is important for safety-critical applications.

We start with OPE methods for policy evaluation which refers to evaluating a target policy π_θ from data generated by a behavior policy π_b. When the off-policy estimate is guaranteed with some confidence that its performance is not worse than the behavior policy π_b, it is called HCOPE. An empirical study of OPE methods [16] discussed the applicability of both methods and presented method selection guidelines depending on the environment parameters and the mismatch between π_θ and π_b. This study categorized OPE methods into importance sampling (IS) methods, direct methods, and hybrid methods that combine aspects from both IS and direct methods. Importance Sampling (IS), or Inverse Propensity Scoring, as known in statistics, is one of the widely used methods for off-policy evaluation where rewards are re-weighted by the ratio between π_θ and π_b [19]. This weighting results in a consistent and unbiased off-policy estimator. Later versions of importance sampling provide lower-variance estimates such as weighted importance sampling (WIS) [20], per-decision importance sampling (PDIS), and per-decision weighted importance sampling (PDWIS) [19]. Then, there are direct methods that rely on regression techniques to directly estimate the value function of the policy. This category includes model-based methods where the transition dynamics and reward are estimated from historical data via a model. Then, the off-policy estimate is computed with Monte-Carlo policy evaluation in the model [21]. Another direct method is fitted q-evaluation (FQE) [22], a model-free approach that acts as the policy evaluation counter-part to batch Q learning or fitted q-evaluation (FQI) [23]. The last category contains the hybrid methods that combine different features from IS and direct methods. This category mainly involves doubly-robust methods [24] that use a direct model to lower IS variance. A few other methods improve upon this approach, such as weighted doubly-robust (WDR) [25], where IS weights are normalized. Another approach is MAGIC [25] which adaptively mixes IS and model-based methods.

2.3 Gap in The Literature

In the offline RL literature, there is no defined way to evaluate a policy while learning offline; most of the literature still evaluates the performance of offline algorithms in the environment or in a simulator, by running a policy online. A workflow for offline RL [26] recently proposed a set of metrics to indicate how the algorithm can be further improved, such as detecting over-fitting and under-fitting, providing no clear workflow for estimating policy performance.

At the Intersection of Offline RL and OPE, there has been recent work combining these two research areas. One paper discussed the applicability of FQE [22] to test the performance of offline policies under different hyper-parameters and select the best performance [27]. Active offline policy selection [28] uses OPE to warm-start the evaluation process and choose which policy to evaluate online when given a limited budget of online interactions. In Benchmarks for Deep Off-Policy Evaluation [29], the authors evaluate offline policies learned on continuous control tasks using various OPE methods ranging from importance sampling, model-based methods, and doubly-Robust estimators. They show how challenging OPE is as an evaluation method [29].

Given the previous work [27] [29], none discuss the feasibility of evaluating offline RL agents during learning π_θ, and how much we can trust HCOPE as an approach for testing. The setup we are studying is quite different from the current literature because previous work for off-policy evaluation assumed a behavior policy π_b and a target policy π_θ, where both policies are fixed and may be related. For example, in a study for HCOPE methods [17], the target policy is initialised as a subset of the behavior policy such that they are close to each other. In another study for safe improvement [18], the Daedulus algorithm learns a safe target policy as a continuous improvement over the behavior policy. This is quite relevant to what we have here; however, their algorithm uses data from an older version of the policy it currently improves to perform both the improvement step and safety tests, so it is not offline

learning. With the doubly-robust estimator, results are presented with different versions of π_θ such that π_θ is always a mixture of π_b and π_θ with different degrees to ensure their relevance [24].

Recent work on offline RL focuses on continuous control in near-complex domains and environments [1]. As detailed above, Batch RL focuses on much simpler domains in discrete and continuous control [16]. None of the previous work discussed the feasibility of using HCOPE to evaluate offline reinforcement learning. There is a gap in how HCOPE methods would perform if the target policy is improved independently from the behavior policy, which is the case for offline RL.

Chapter 3

Framework

This chapter introduces our novel framework. This includes the methods used for offline policy learning, bootstrapping for high-confidence estimation, and off-policy evaluation.

3.1 Proposed Framework

Our framework couples offline RL algorithms with approximate high-confidence off-policy evaluation as a feasible evaluation method for offline agents in safety-critical domains. In this framework, we sample data dynamically at each iteration, perform policy updates, and continuously monitor its performance with a confidence interval until it reaches a reasonable performance and is ready to be deployed. The policy learning offline does not interact with the environment unless it passes the safety test. Specifically, we have k iterations where, in each iteration i, the data D_i of size n is sampled from the main data source. D_i is split between D_{train} to train our target policy $\pi_{\theta i}$ and D_{test} to perform the safety test using approximate high-confidence off-policy evaluation methods. We return when the learned policy is ready to be deployed with appropriate confidence. Instantiating our framework requires selecting an offline policy learning method and an off-policy evaluation method for computing off-policy estimates. We study various methods for offline policy learning and off-policy evaluation as our methodology.

Pseudo-Code

We summarize an interaction between an agent and a source of data as follows:

1. An agent uses a set of data D_i without knowing the behavior policy.
2. We optimize a policy π_{θ_i} offline using D_{train} with an offline RL algorithm from Section 3.2.
3. We perform a safety check using D_{test} with a high-confidence evaluation method from Section 3.4.
4. Our new policy is either ready to be deployed or we go back to step 1.

We refer to each offline policy learning method with Φ and each high-confidence off-policy evaluation method with Ψ. Our safe evaluation framework is further explained in Algorithm 1. The inputs are a dataset of trajectories D, a confidence level δ appropriate for the problem in hand, offline policy learning method Φ, and an off-policy evaluation method Ψ to combine with bootstrap confidence intervals (BCI) shown in Algorithm 2. The output will be a policy trained offline π_θ and a high-confidence lower-bound estimate of its return $\hat{v_\delta}(\pi_\theta)$. Since the stopping condition in Algorithm 2 is not always satisfied if no HOPCE method can detect an improved policy, we set a maximum number of iterations for training and evaluation in our experiments.

3.2 Methodology: Offline Policy Learning

Is it possible to learn a good policy offline given data and no environment interaction? In one of the early offline RL papers [5], authors tried to answer the question: *Can standard off-policy RL algorithms with no environment interactions succeed?* Their approach paved the way for how capable RL algorithms are in the offline setting, since they were able to learn a successful DQN agent totally offline when provided

Algorithm 1 Offline Safe Evaluation Framework

Input: initial π_θ, dataset D of m trajectories, confidence level $\delta \in [0, 1]$, number of bootstrap estimates B, offline learning method Φ, and an OPE method Ψ
Output: π_θ, $\hat{v_\delta}(\pi_\theta)$: $1 - \delta$ lower-bound on $\hat{v}(\pi_\theta)$

Let $i = 0$
while $\hat{v_\delta}(\pi_\theta) \leq \hat{v}(\pi_b)$ **do**
    Sample a subset of trajectories D_i with size n
    Split D_i into D_{train} and D_{test}
    Optimize policy $\pi_{\theta_i} = \Phi(D_{train})$
    Evaluate policy $\hat{v_\delta}(\pi_\theta) = \text{BCI}(\pi_\theta, D_{test}, \delta, B, \Psi)$
end while
return π_θ, $\hat{v_\delta}(\pi_\theta)$

with diverse data. However, a benchmark on batch RL [30] shows how RL without correction can fail in the offline setting given ordinary data.

Given our setup, we investigate different techniques for learning from offline data, ranging from normal off-policy RL algorithms (off-policy), imitation learning techniques (imitation learning), and policy constraint methods specific for offline learning (offline). Imitation learning is a form of supervised learning and is good at learning from expert data or demonstrations. Policy constraint methods are designed to mitigate the challenges for normal off-policy to learn offline, such as bootstrapping error [1]. This categorization is inspired by the offline RL tutorial [1] and is further shown in Table 3.1.

For continuous control, we investigate a few approaches in the three categories mentioned above. For the off-policy category, we use Soft Actor-Critic (SAC) [14]; this means a SAC agent is optimized offline using a fixed buffer of data without interaction with the environment or control over the buffer. For the imitation learning category, we study Behavioral Cloning (BC) [15]; BC is a supervised learning algorithm where the model learns how to predict actions from the current state.

In the offline category, we study batch-constrained Q-learning (BCQ) [6] and bootstrapping error accumulation reduction (BEAR) [10]. Batch-constrained Q-learning [6] argue that off-policy algorithms often fail in the offline setting due to the extrap-

olation error such that unseen state-action pairs will have unrealistic values [6]. This is a result of the mismatch between the state-action visitation of the current policy and the state-action pairs in the offline dataset. To solve this, the policy should induce a similar state-action visitation to the offline data. BCQ [6] proposed the use of a state-conditioned generative model to produce only likely actions to the current offline data. Then, this generative model, effectively a variational auto-encoder [31], is combined with a network which aims to optimally perturb the generated actions in a small range. When this is combined with the Q-network, the network will only select the highest valued actions similar to the data in the batch [6].

BEAR [10] argues that the source of instability for off-policy algorithms learning offline is the bootstrapping error. This error results from bootstrapping with actions that lie outside of the training data distribution, and it accumulates via the Bellman backup operator [10]. BEAR is built on top of any actor-critic algorithm, such as SAC [14], by modifying the policy improvement step to use a distribution-constrained backup. To apply the constraint, the sampled version of maximum-mean discrepancy is used between the unknown behavior policy π_b and the current actor π_θ to constrain the distribution of the actor to the support of the behavior policy [10]. Hence, BEAR allows the actor to maximize the Q-function while being constrained to remain in the valid support space of the behavior policy defined by the data samples [10].

For discrete control, we use Double DQN [32] as a normal off-policy RL algorithm; similar to SAC, Double DQN is used to learn solely from offline data. For the imitation learning category, we use behavioral cloning (BC) [12] in a supervised learning manner with cross-entropy loss. In the offline category, we experiment with the discrete version of BCQ [30]. Discrete BCQ is much simpler than its continuous version, such that it trains Q-learning with a constrained argmax operator. This only allows actions in the backup with probability, given by the generative model, above some threshold τ. The generative model is a behavioral cloning network trained in standard supervised learning with a cross-entropy loss [30]. BCQ is also based on Double DQN,

and we use the threshold to be $\tau = 0.3$ as reported in their original paper. If the threshold τ is 1, the algorithm returns an imitator of all the actions in the dataset, while a threshold $\tau = 0$ returns the Q-learning objective.

-	**Off-policy**	**Imitation Learning**	**Offline**
Continuous Control	SAC	Behavioral Cloning	BCQ / BEAR
Discrete Control	Double DQN	Behavioral Cloning	BCQ

Table 3.1: **Categories for offline Learning**: This table shows different categories for offline learning methods for both discrete and continuous control, with examples of algorithms used in our experiments

Given the data samples used from a data source, we aim to learn a policy totally offline with no environment interaction. This offline policy should be at least as good as the behavior policy in case of imitation learning and better in the case of other algorithms, after a few iterations.

3.3 Methodology: Bootstrapping

Consider a sample D of n random variables H_j for $j = 1, 2, ..., n$ where we can sample H_j from some *i.i.d.* distribution of data. From the sample of data D, we can compute a sample estimate $\hat{x}$ of a parameter x such that $\hat{x} = f(D)$ where f is any function that estimates x. Given a dataset D, we create B resamples with replacement, where B is the number of bootstrap resamples, and compute an estimate for x, $\hat{x}$, on each of these resamples. Bootstrapping [33] allows us to estimate the distribution of $\hat{x}$ with confidence bounds. The estimates computed with different resamples will determine the $1-\delta$ confidence interval. In our setup, the parameter of interest x is the expected return of a policy $v(\pi_\theta)$.

Hence, with a confidence level $\delta \in [0, 1]$ and B resamples of the dataset of trajectories D, we use bootstrapping to approximate a lower-bound confidence interval of

Algorithm 2 Bootstrap Confidence Intervals: BCI

Input: a target policy π_θ, dataset D_{test} of m trajectories, confidence level $\delta \in [0, 1]$, number of bootstrap estimates B, an estimate of the behavior policy $\hat{\pi_b}$, and off-policy estimator Ψ

Output: $\hat{v_\delta}(\pi_\theta)$: $1-\delta$ lower-bound on $\hat{v}(\pi_\theta)$

1: **for** $j \in [1, B]$ **do**
2: $\quad \tilde{D_j} = H_1^j, .., H_m^j$ where H_i^j is sampled uniformly
3: $\quad \hat{v} = \Psi(\pi_\theta, \tilde{D_j}, \hat{\pi_b})$
4: **end for**
5: sort$(\hat{v_j} | j \in [1, B])$ // ascending
6: $l = \lfloor \delta B \rfloor$
7: **return** $\hat{v_l}$

$v(\pi_\theta)$ on $v(\pi_\theta)$ such that $v_\delta(\pi_\theta) \leq v(\pi_\theta)$ with probability at least $1-\delta$. This bootstrap method is referred to as the percentile bootstrap for confidence intervals [34]. After obtaining B resamples of the dataset, we compute $\hat{v}(\pi_\theta)$ with each of these resamples. Then, all the off-policy estimates $\hat{v}(\pi_\theta)$ are sorted ascendingly and the index $\delta \times B$ is chosen, to get $\hat{v_\delta}(\pi_\theta)$

As the size of data $n \to \infty$, bootstrapping has strong guarantees, but it lacks guarantees for finite samples [21]; this is because it assumes that the bootstrap distribution is representative of the true distribution of the statistic of interest, which may be false for finite samples [21]. As a result of this false assumption, bootstrapping is considered semi-safe; an algorithm is called semi-safe if it ensures safety while making a reasonable assumption that may be false. Previous work [35] shows that bootstrapping is still safe enough for high-risk medical predictions with a known record of producing accurate confidence intervals. Moreover, other exact concentration inequalities may be too loose with small amounts of data so bootstrapping is practically more useful for our setup.

When bootstrapping is combined with off-policy evaluation methods, we can estimate the distribution of $v(\pi_\theta)$, and use it to estimate $\hat{v}_\delta(\pi_\theta)$ with probability at least $1-\delta$.

3.4 Methodology: Policy Evaluation

While a policy is being improved offline, we are interested in evaluating such a policy. Off-policy evaluation is a promising technique to evaluate a policy that is continuously learning given access to fixed data. An off-policy estimator is a method for computing an estimate $\hat{v}(\pi_\theta)$ for the true value of the target policy $v(\pi_\theta)$ using trajectories D collected while following another policy π_b, which is what OPE methods do. For HCOPE, this extends OPE to lower-bounding the performance of the target policy, π_θ. When evaluating, we use new samples for testing each time to avoid the multi-comparison problem[2]. This is also to ensure that we do not over-fit our OPE estimates or tune training parameters to reduce the estimate error.

Below, we detail one method in each category of the OPE methods as discussed in Section 2.2.2. Each method is combined with the bootstrapping method to approximate a confidence lower bound of $\hat{v_\delta}(\pi_\theta)$ on $\hat{v}(\pi_\theta)$ such that $\hat{v_\delta}(\pi_\theta) \leq \hat{v}(\pi_\theta)$ with probability at least $1 - \delta$. Table 3.2 shows the three categories of OPE methods, with examples under each category. The methods in bold are the ones we use for evaluation.

Importance Sampling	**Direct**	**Hybrid**
IS[19]	Fitted q-evaluation [22]	Doubly-robust[24]
WIS[20]	**Model-Based**[25]	**Weighted doubly-robust** [25]
PDIS[19]/PDWIS[19]	-	MAGIC [25]

Table 3.2: **Different OPE methods categorized**: This table shows the three main categories of OPE methods, importance sampling, direct methods and hybrid methods with examples.

[2]The problem occurs when conducting multiple statistical tests simultaneously with reusing data, so the confidence bound does not strictly hold anymore

3.4.1 Importance Sampling with Bootstrapping

Importance Sampling (IS) [19] is a method for handling mismatch between distributions and hence presented as a consistent and unbiased off-policy estimator. For a trajectory $H \sim \pi_b$ of length L, as $H = S_1, A_1, .., S_L, A_L$, we can define the importance sampling up to time t for policy π_θ as follows:

$$IS(\pi_\theta, \pi_b, D) = \sum_{i=1}^{m} \rho_L^H R^i, \quad \rho_t^H := \prod_{j=0}^{t} \frac{\pi_\theta(A_j|S_j)}{\pi_b(A_j|S_j)} \tag{3.1}$$

By re-weighting the returns, we can tell how likely each reward is under π_θ versus π_b. However, IS often assumes a known π_b, which is not always the case for off-policy policy evaluation. Since we assume no access to π_b in our setting, we estimate a behavior policy $\hat{\pi}_b$ from data with a behavior cloning model [36]. To come up with a high confidence estimate, we first compute the importance-weighted returns then use bootstrapping, described in Section 3.3, to get the lower-bound estimate. In our experiments, we use weighted importance sampling (WIS) [20] with bootstrapping as a representation for importance sampling family of estimators.

$$WIS(\pi_\theta, D, \pi_b) = \sum_{i=1}^{m} \frac{\rho_{L-1}^i}{\sum_{j=1}^{m} \rho_{L-1}^j} g(H_i) \tag{3.2}$$

3.4.2 Direct Model-Based Bootstrapping

Model-based estimation is another off-policy estimator that falls under direct methods. The model-based off-policy estimator (MB) computes $\hat{v}(\pi_\theta)$ by building a model using all the available trajectories D to build a model $\hat{M} = (S, A, \hat{P}, r, \gamma, \hat{d}_0)$ where $\hat{P}$ and $\hat{d}_0$ are estimated with trajectories sampled from π_b. Then, MB will compute $\hat{v}(\pi_\theta)$ as the average return of trajectories simulated in the estimated model $\hat{M}$ while following π_θ. Despite having lower variance than IS methods, MB is a biased estimator for: 1) we lack data for particular state-action pairs, so we assume their transition probabilities, 2) we assume that the model class includes the true model for the transitions. As a result, as $n \to \infty$, the model estimates may converge to a value different

from the true $v(\pi_\theta)$. This effect is due to its dependency on the modeling assumptions we make whether we assume a linear or a non-linear model.

To get a confidence lower-bound on the MB estimate, we use bootstrap confidence intervals as one of the approximate confidence intervals for OPE, as discussed in Section 3.3. So, the model-based estimator is combined with bootstrapping for high-confidence off-policy evaluation. To clarify, as mentioned in Section 2.2.1, we rely mainly on model-free policy optimization algorithms for learning a policy offline. However, we use model-based estimation to evaluate a policy, which is independent from the policy optimization part.

3.4.3 Weighted Doubly-Robust with Bootstrapping

Weighted doubly-robust (WDR) [25] is a hybrid method for off-policy estimation, presented as an extension to the doubly-robust (DR) method [24]. DR is an unbiased estimator of $v(\pi_\theta)$ that uses an approximate model of the MDP to reduce the variance of importance sampling [24]. Although biased, WDR is based on per-decision weighted importance sampling (PDWIS) and improves upon the DR method as it balances the bias-variance trade-off. In addition, the approximate model value functions act as a control variate for PDWIS.

$$PDWIS(\pi_\theta, D, \pi_b) = \sum_{i=1}^{m} \sum_{t=0}^{L-1} \frac{\rho_t^i}{\sum_{j=1}^{m} \rho_t^j} \gamma^t R_t^i \tag{3.3}$$

$$\begin{aligned} WDR(\pi_\theta, D, \pi_b) = PDWIS(\pi_\theta, D, \pi_b) - \\ \sum_{i=1}^{n} \sum_{t=0}^{L-1} \gamma^t (w_t^i \hat{q}_{\pi_\theta}(S_t^i, A_t^i) - w_{t-1}^i \hat{v}_{pi_\theta}(S_t^i)) \end{aligned} \tag{3.4}$$

Similar to direct model-based methods, we use bootstrapping with WDR to provide a confidence lower-bound estimate on $\hat{v}(\pi_\theta)$. WDR with bootstrapping is guaranteed to converge to the correct estimate as n increases, given the statistical consistency of PDWIS [21]. For the approximate model, a single model is estimated with the

available trajectories D, then used to compute the value functions of WDR for each bootstrap data. We choose the weighted doubly robust estimator to represent OPE hybrid methods in our study.

Chapter 4

Experiments: Simulated Environments

This chapter discusses the experiments and the results of the proposed framework on simulated offline setups in discrete and continuous control tasks.

4.1 Environments

In our experiments, two simulated environments were used to demonstrate the framework in discrete and continuous control. This shows how the complexity of the environment, state and action dimensions, and horizon are affecting high-confidence off-policy evaluation. To mimic the setup where there exists a source of pre-collected trajectories without access to the environment itself, we simulate a source for data in each environment as an online partially-trained policy that collects data of a medium quality. A medium-quality data that is not optimal nor random resembles real-world data and leaves room for improvement by offline RL algorithms.

- **MountainCar-v0**: we use discrete MountainCar [3] with a continuous state (velocity and position) and 3 possible discrete actions. At each time-step, the reward is -1, except for the terminal state when it is 0. However, we used the modified version of Mountain-Car as described here [37]. The horizon is shortened by holding an action a_t constant for 4 updates of the environment state. We also change the start state such that an episode starts with a ran-

dom position in the range of $\langle -1.2, 0.6 \rangle$ and random velocity in the range of $\langle -0.07, 0.07 \rangle$ [24, 37]. The data collector we use for this environment is an online actor-critic [3] agent that is partially trained with added 30% randomization when generating data. This means when collecting the offline data, an agent takes a random action 30% of the time instead of following the online policy, to include exploratory transitions.

- **Pendulum-v0**: inverted pendulum swing-up problem is a classic problem in the control literature with one continuous action [38]. In this version of the problem, the pendulum starts in a random position, and the goal is to swing it up to stay upright. We also shorten the horizon following the same procedure as done in MountainCar [37] by holding an action constant for 4 updates of the environment state. This limits the horizon of the environment to 50 instead of 200. The data source we use for this environment is a soft actor-critic [14] agent that is trained partially, with added 30% randomization when generating the dataset. This means when collecting the offline data, an agent takes a random action 30% of the time instead of following the online policy to resemble real-world data.

4.2 Experimental Details

For each environment, a policy is optimized offline using three different offline learning methods and evaluated simultaneously using three different off-policy evaluation methods. We follow algorithm 1, but the loop is limited to a maximum of 10 iterations because we cannot rely on all HCOPE methods detecting $\hat{v_\delta}(\pi_\theta) > \hat{v}(\pi_b)$. We have m trajectories from the source of data which we split between training and testing; we use more data for testing since HCOPE requires more data to estimate tight bounds. For each improvement method, we load the train split into the training buffer and do k policy updates where k is tuned per algorithm such that the offline policy converges

within the total number of iterations. Since we have access to the true environment in these simulated domains, we rely on the environment to compute the true value of the offline policy; hence, we use this information to optimize an offline policy that can outperform the behavior policy.

For bootstrapping, we use $\delta = 0.05$ to get a 95% confidence lower-bound using $B = 2000$ bootstrap estimates as recommended by practitioners [33]. In weighted doubly-robust with bootstrapping 3.4.3, we use a value of $B = 224$ for the number of bootstrap estimates to avoid heavy computation; this can still get us a good approximation as suggested by MAGIC [25].

Since we have no access to π_b, we estimate $\hat{\pi_b}$ given the test data [36] with a behavior cloning model that is improved gradually as more test data gets in.

4.3 Results

Below, we show experimental details and results on MountainCar-v0 and Pendulum-v0.

4.3.1 MountainCar-v0

In each iteration, 300 trajectories are sampled, where 20 trajectories are used for training the policy and 280 trajectories are used for evaluation. This totals to 3000 trajectories over the 10 iterations. Each iteration includes performing k policy updates to the offline policy; a policy update is a single optimization step of the policy over a batch of data. k is set to be 2^9 for BCQ and Double DQN, while BC does 2^6 policy updates in one iteration. Figure 4.1 shows how different offline policy improvement methods perform given medium-quality data along with HCOPE estimators, indicating which method can tell when $\hat{v_\delta}(\pi_\theta) > \hat{v}(\pi_b)$. Behavior cloning, as an offline policy improvement method, can only perform as well as the data distribution π_b. Double DQN and BCQ were able to outperform π_b. The true value of a target policy is calculated as the average return when running the policy in the actual environment

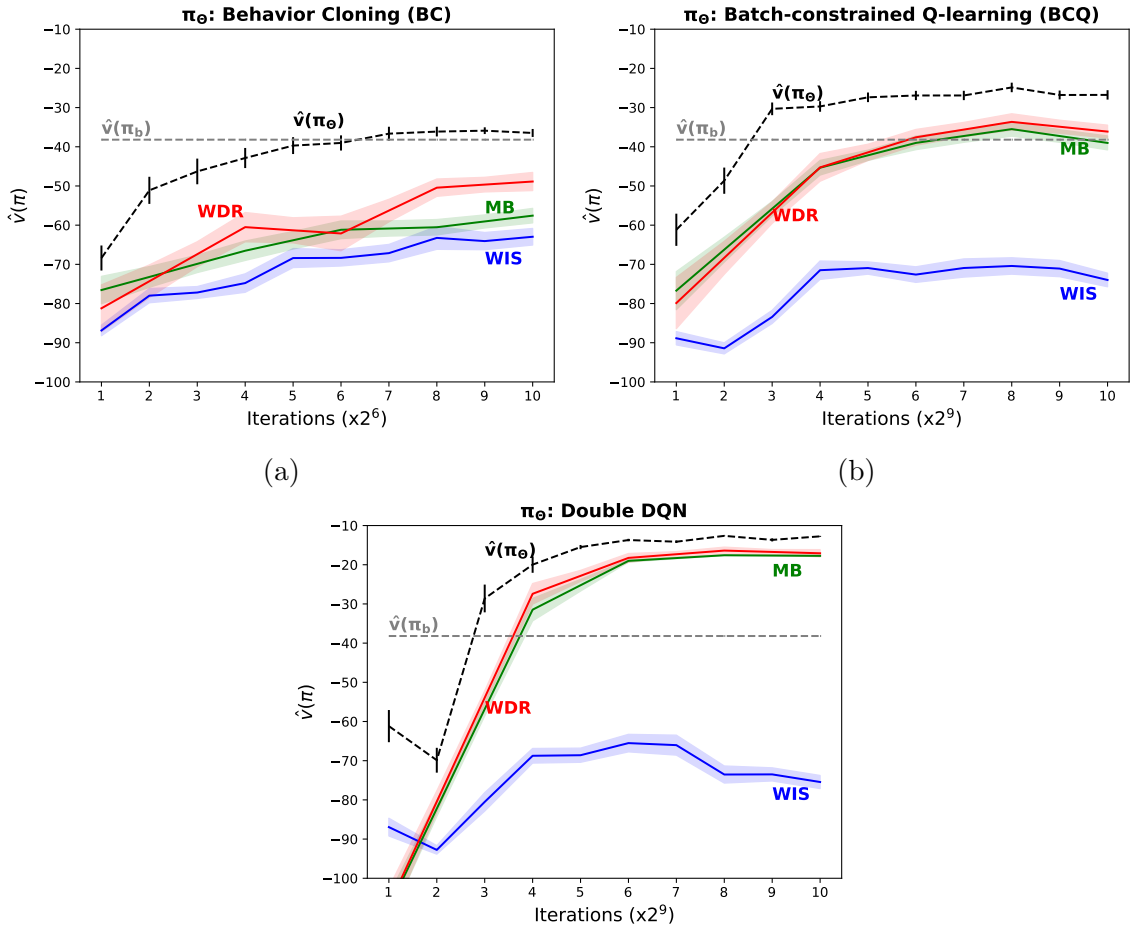


Figure 4.1: **Results of offline learning and high-confidence evaluation on MountainCar-v0**: This figure shows results of evaluating the performance of the target policy with HCOPE as it is training offline. Weighted importance sampling (WIS) failed to detect that the offline policy outperforms π_b. The model-based estimator (MB) and weighted doubly-robust (WDR) were able to determine when an offline agent outperforms π_b.

(not possible in practice) for 1000 episodes. All reported results are an average of 40 runs, while the shaded area shows the standard error. The value of the behavioral policy, $\hat{v}(\pi_b)$, is the sum of undiscounted rewards of all trajectories in the data set. Since π_b is unknown, we first estimate $\hat{\pi_b}$ with a behavior cloning model [36] given the test data and improve our estimate as we sample more test data over iterations.

For high-confidence evaluation of each offline agent, weighted importance sampling (WIS) with bootstrapping failed to detect that the offline policy outperforms π_b for all learning methods. The model-based estimator (MB) and weighted doubly-robust with bootstrapping (WDR) have much less error with respect to the true value of a policy and were able to infer when an offline agent outperforms π_b and hence is ready to be deployed. For instance, with 95% confidence, in the case of offline improvement with Double DQN, we were able to tell that our new target policy is better than the behavior policy at iteration 4 using two different estimators (MB and WDR).

4.3.2 Pendulum-v0

In each iteration, 500 trajectories are sampled, where 100 trajectories are used to train the policy and 400 trajectories for evaluation. This totals 5000 trajectories over the 10 iterations. Figure 4.2 shows how different offline policy improvement methods perform given medium-quality data along with HCOPE estimators, indicating which method can tell when $\hat{v_\delta}(\pi_\theta) > \hat{v}(\pi_b)$. Behavior cloning as an offline policy improvement method can only perform as well as the data by π_b, while BEAR and BCQ were able to outperform π_b. We did not report results on SAC [14] since it failed to learn a policy better than the behavior policy. The true value of a target policy is calculated as the average return when running the policy in the actual environment (not possible in practice) for 1000 episodes. All reported results are an average of 40 runs, while the shaded area shows the standard error. The value of the behavioral policy $\hat{v}(\pi_b)$ is the sum of undiscounted rewards of the data set. Since π_b is not known, we first estimate $\hat{\pi_b}$ [36] given the test data and improve it as we sample more test data.

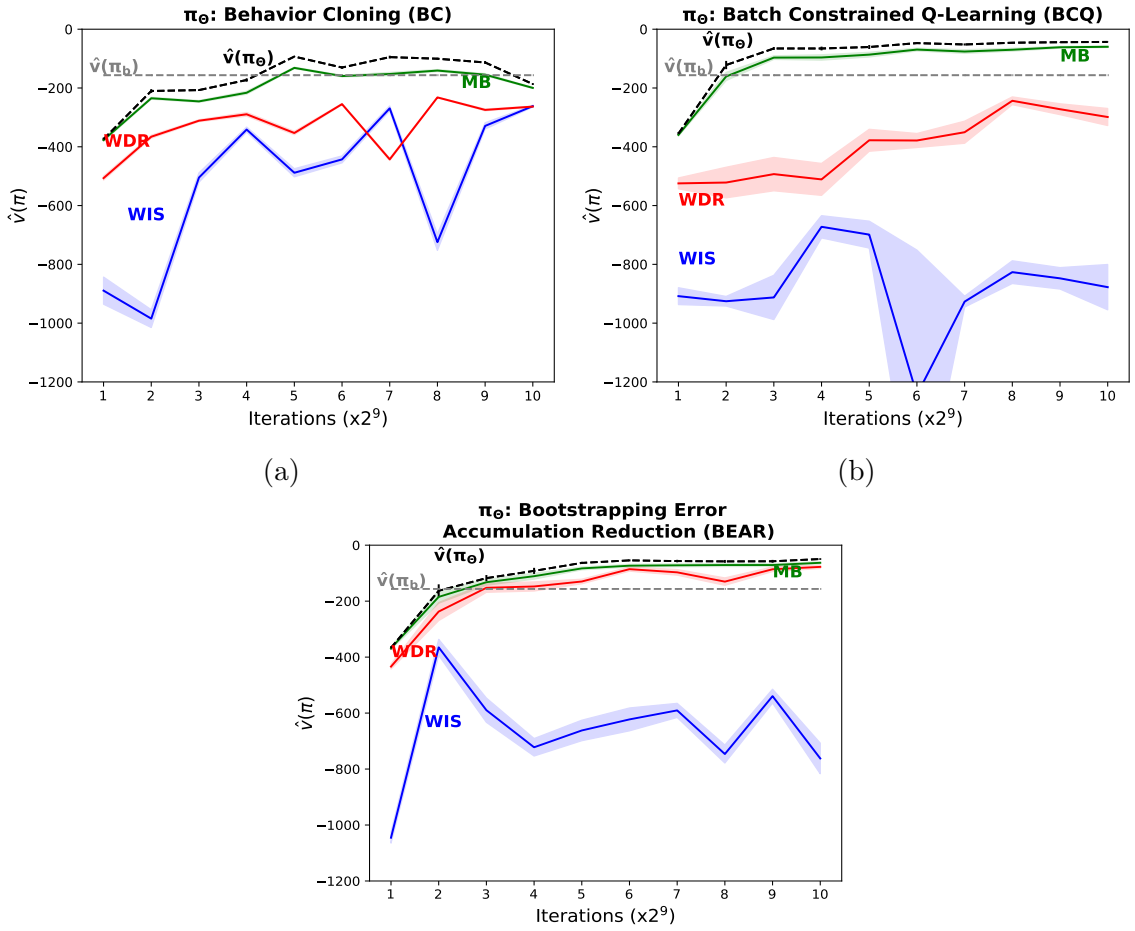


Figure 4.2: **Results of offline learning and high-confidence evaluation on Pendulum-v0**: This figure shows results of evaluating the performance of the target policy with HCOPE as it is training offline. Weighted importance sampling (WIS) failed to detect that the offline policy outperforms π_b. The model-based estimator (MB) and weighted doubly-robust (WDR) were able to determine when an offline agent outperforms π_b.

Similar to the results on MountainCar-v0, WIS with bootstrapping failed to detect that the offline policy outperforms π_b. The model-based estimate (MB) achieve a very tight lower-bound with the true value of a policy and were able to inform when an offline agent outperforms π_b and hence is ready to be deployed. Weighted doubly-robust with bootstrapping (WDR) estimates are also affected by the improvement methods as in the case with WIS.

4.4 Analysis of Evaluation Methods

To better understand how evaluation is affected by how the offline learning method, multiple measures can show the similarity between two probability distributions (as a policy is a distribution over actions). One simple measure is the total variation (TV) distance. TV distance measures the difference between action probabilities taken under two policies given the data set in hand. TV would be the sum of differences in probabilities between the behavior policy π_b and the target policy π_θ for each state-action pair in the test dataset. Since π_b is not known, we estimate $\hat{\pi_b}$ given the test data [36].

HCOPE	**Off-policy** (Double DQN)	**Imitation Learning** (BC)	**Offline**(BCQ)
WIS	49.032 ± 0.57	**28.742** ± 1.22	43.66 ± 0.84
MB	**7.85** ± 0.72	22.716 ± 1.09	13.645 ± 0.65
WDR	**6.18** ± 1.45	17.642 ± 2.79	12.81 ± 2.42

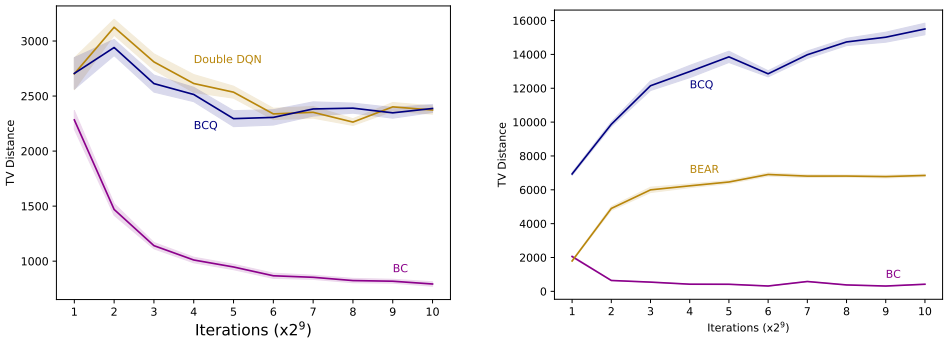
Table 4.1: **Mean error between HCOPE estimates and true value of π_θ on MountainCar-v0**: This table shows the mean error of HCOPE estimates, $\hat{v_\delta}(\pi_\theta)$, over all training iterations and multiple runs given each offline improvement method. Lowest mean errors are in bold. Importance sampling (WIS) achieves the lowest error with imitation learning. MB and WDR achieve the lowest error when the offline policy is optimal, as in the case for Double DQN.

Some HCOPE methods, such as importance sampling (IS), rely on the similarity between the behavior policy π_b and the target policy π_θ. We analyze how the im-

HCOPE	**Off-policy**(SAC)	**Imitation Learning**(BC)	**Offline**(BCQ)	**Offline**(BEAR)
WIS	-	**268.5** ± 28.93	602.49 ± 78.05	432.22 ± 47.5
MB	-	33.48 ± 0.77	25.02 ± 2.5	**19.0** ± 5.1
WDR	-	161.1 ± 13.47	307.11 ± 37.3	**54.75** ± 10.43

Table 4.2: **Mean error between HCOPE estimates and true value of π_θ on Pendulum-v0**: This table shows the mean error of HCOPE estimates, $\hat{v_\delta}(\pi_\theta)$, over all training iterations and multiple runs given each offline learning method. Lowest mean errors are in bold. Importance sampling (WIS) achieves the lowest error with imitation learning. MB and WDR achieve the lowest error when the offline policy is optimal, as in the case BEAR, which is also quite close for BCQ. We omit results on learning a target policy by SAC since it failed to learn a policy that outperforms the behavior policy.

provement method is affecting TV distance between $\hat{\pi_b}$ and π_θ and hence affecting the high-confidence off-policy evaluation estimates. TV distance is correlated to KL-Divergence, showing how two policies are different from each other. Figure 4.3 shows the total variation distance between the offline policy π_θ and the estimated behavior policy $\hat{\pi}_b$ across the different offline learning methods. For discrete control, it shows that behavioral cloning (imitation learning) can achieve a much lower distance than other improvement methods whether they are constrained or non-constrained (batch-constrained Q-learning and Double DQN). This is because behavioral cloning forces its target policy to be close to the behavior policy while other methods do not. The same applies to continuous control in Pendulum-v0; behavior cloning achieves the lowest error between $\hat{\pi_b}$ and π_θ. BEAR achieves a higher distance but much less than BCQ; this is because BEAR enforces explicit divergence minimization between $\hat{\pi_b}$ and π_θ while exploiting the data [10]. This insight on divergence explains why weighted importance sampling achieves the lowest error between the estimate and the true value of a policy in the case of behavioral cloning, as shown in Tables 4.1



(a) MountainCar-v0 (b) Pendulum-v0

Figure 4.3: **Divergence between $\hat{\pi}_b$ & π_θ over training iterations**: This figure shows the total variation (TV) distance between $\hat{\pi}_b$ & π_θ on MountainCar-v0 & Pendulum-v0 over all training iterations. For MountainCar-v0, behavior cloning (BC) learns a policy that is very similar to $\hat{\pi}_b$. Discrete BCQ and Double DQN do not limit the distance with the behavior policy, and their TV distance is quite large as opposed to BC. For Pendulum-v0: behavior cloning (BC) also learns a policy that is very similar to $\hat{\pi}_b$ and achieves the lowest TV distance. BEAR limits the divergence compared to BCQ since BEAR's objective is constrained to be close to π_b.

and 4.2; the error grows for other methods that do not constrain the policy to be close to the data distribution.

Other than the divergence between $\hat{\pi}_b$ & π_θ, the horizon of the trajectory affects importance sampling such that it suffers from high variance with longer horizons [39]. The MB estimator performs well when it is possible to model the environment dynamics such as the case for both MountainCar-v0 and Pendulum-v0 environments. The WDR estimator also performs well because q-values can get more accurate when it is easy to learn a model of the environment. However, the WDR estimator is still affected by the divergence between $\hat{\pi}_b$ & π_θ as in the case for BC and BCQ algorithms. WDR estimates get worse as the divergence increases.

4.5 Discussion

It is important to question whether high-confidence off-policy evaluation (HCOPE) is worth the added complexity and computation over off-policy evaluation (OPE). OPE can provide good estimates of an offline target policy. OPE is challenging; the

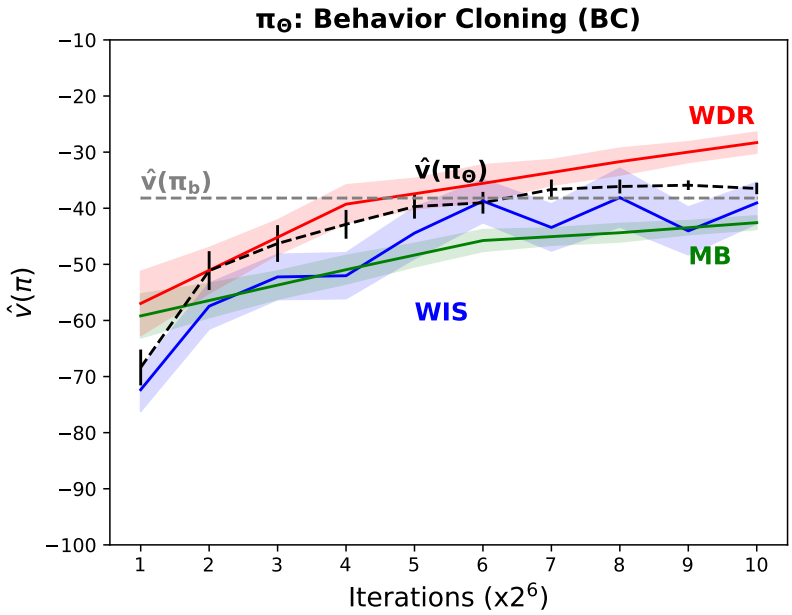
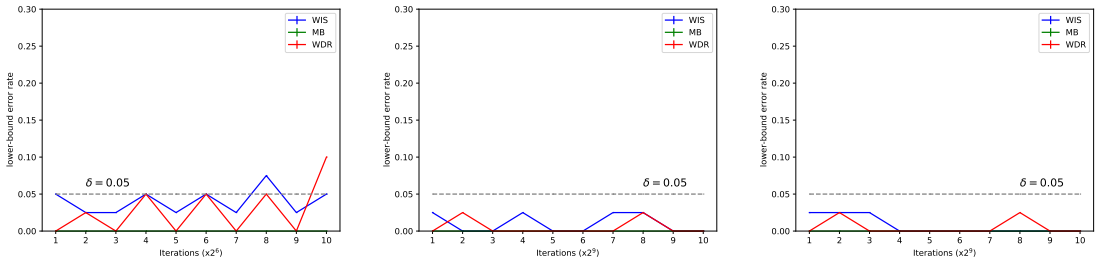


Figure 4.4: **Results of off-policy evaluation without bootstrapping on MountainCar-v0**: This figure shows results of evaluating a policy with OPE. The target policy is optimized offline with behavioral cloning and evaluated with OPE without bootstrapping. Results show that OPE overestimates the true value of the policy, and it is better to trust high-confidence OPE in safety-critical applications.

same challenges apply to high-confidence off-policy evaluation (HCOPE). However, HCOPE estimates ensure safety by underestimating the value of the offline policy. Empirical results on simulated environments showed how OPE estimates overestimate the true value of the offline policy, even if it achieves less error between the estimate and the true value, as shown in Figure 4.4. On the other side, HCOPE lower bounds are guaranteed to overestimate the true value of the offline policy only within the allowable 5% error rate for a 95% confidence, as shown in Figure 4.5.

Our ability to detect an improved policy and decide when to deploy, is a function of how much better the policy is and how good our high-confidence OPE methods are. It is better to rely on estimators that do not take π_b into account (e.g., direct methods or at least hybrid methods) so that value estimates are less affected by the divergence between π_θ $\hat{\pi}_b$ (as the case for WIS). Accordingly, high-confidence off-policy estimators (e.g., direct methods or hybrid methods) are considered a safe



(a) BC (b) BCQ (c) Double DQN

Figure 4.5: **The empirical error rate on MountainCar-v0 with HCOPE**: This figure shows the empirical error rate on MountainCar given different HCOPE methods. The lower bound is computed z times for each method ($z = 40$ for Mountain Car) and we count how many times the lower bound is above the true $\hat{v}(\pi_\theta)$ computed in the true environment. Given 200 trajectories only for evaluation, all methods correctly approximate the allowable 5% error rate for a 95% confidence lower bound. However, there are two instances of WIS and WDR that slightly exceed 5% which can be mitigated with more data.

evaluation method for offline learning with enough confidence that minimizes the risk of overestimating the true performance of an offline policy. In contrast, off-policy evaluation suffers from overestimation to the true values so it makes sense to rely on HCOPE instead, specially for safety-critical applications.

Finally, we would like to point out that the safety guarantee for the confidence bound specified is per-iteration [37]. If we want to consider this guarantee over multiple iterations, the probability $1 - \delta$ decreases for the early iterations as we perform more tests. As a result, we should only consider this guarantee for the last iteration of our loop, such that the error rate in the last iteration is at most δ.

Chapter 5

Applicability on Real-World Data

The main motivation for this thesis is to find a way to evaluate RL agents learning offline, to determine when to deploy such agents without access to the real environment. In this chapter, we test our proposed framework on real-world medical data to see how applicable it is and what the challenges are. Below, we describe the problem, data, experiments, challenges, and results.

5.1 Problem: Sepsis Treatment

Problems in healthcare can be considered a form of sequential decision making, such as clinical physicians making decisions about the best next step to take in care [40]. Current advances in artificial intelligence can provide personalized care that is equal to or better than humans by finding an optimal decision making policy given clinical data [40]. Sepsis, a severe infection with organ failure, is a leading cause of mortality in intensive care units (ICUs) in hospitals [41]. There are continuous efforts in research to understand and cure sepsis and, hence, increase the survival rates for ICU patients. Reinforcement learning for sepsis treatment is preferred over supervised learning because the ground truth of good treatment strategy is unclear in the current medical literature [42]. Sepsis data is a form of offline data, from which we can learn an offline policy given no access to an environment. The challenge lies in evaluating such policies while learning to determine when it is good enough to be

used in reality.

In this chapter, we use the well-known MIMIC III data [43] for sepsis treatment. Current literature in reinforcement learning for health [44] [45] finds it challenging to evaluate policies learned from such data when learning a decision-making policy given no agent-environment interaction. Using a deep reinforcement learning approach, one approach [45] relies on learning a policy from MIMIC III data with dueling DQN [13]. To assess the performance of the learned policy, authors [45] use the doubly-robust estimator [24] to estimate the value of their agents. They suggested that it is better than just comparing the Q-values of the learned policy to the Q-values of the estimated physician's policy. A following work [44] suggests that it is not enough to trust the doubly-robust estimator [24], and it is better to rely on qualitative analysis for this problem. This is because they are learning a deterministic RL policy; a deterministic target policy leads to importance sampling weights going to zero if the behavior and learned target policy take different actions at any given time-step [44]. Their qualitative analysis relies on the SOFA (measuring organ failure) score, given in the data, by categorizing the state space into low, medium, and high SOFA cohorts. Then, the actions taken by the RL policy, at each of these cohorts, are compared against the clinician's policy to see if there is an improvement [44].

For this problem, given no further data collection, it is better to rely on offline RL learning methods to optimize a policy from a fixed dataset. A recent paper [46] explores different representation learning techniques to learn a good policy offline with discrete BCQ [30]. To evaluate their policies, they rely on Weighted Importance Sampling [20] to compare the different representation learning techniques. A target policy learned by discrete BCQ [30] will not suffer from the same problem with importance sampling as Dual DQN [44] because BCQ is bounded to imitate actions in the dataset so no action will have a probability of 0. We build upon their work to apply our proposed training-evaluation framework.

The problem at hand is widely researched but, only one previous work [46] relies

on offline learning with off-policy evaluation. Relying on importance sampling for evaluation is good when comparing different approaches; however, we believe it is essential to perform high-confidence evaluation for this critical application.

Data Description

To build a decision-making policy for the treatment of septic patients, we use data from the Medical Information Mart for Intensive Care (MIMIC-III) dataset (v1.4) [43]. We follow previous relevant work [47] to extract and preprocess vital and lab measurements and build a cohort of 19418 patients; this cohort has an observed mortality rate just above 9% (determined by death within 48h of the final observation). Data extraction, preprocessing, and splitting follow the previous work [46].

A dataset D is a collection of trajectories; a trajectory H of a non-fixed horizon L refers to a single patient that ends by the survival or death of a patient, with a maximum horizon L of 19. A transition is composed of current observation O_t, the action taken by the clinician a_t to move to the next observation O_{t+1} receiving reward of r_t until a patient survives or dies. We treat the problem as a Markov decision process (MDP). Data consists of a continuous state space of a dimension 38 and a discrete action space of 25 actions. Observed actions are the administration of fluids or vasopressors that can be given, categorized by volume, and put into 5 discrete bins per action type resulting in 25 actions [46]. The data has sparse reward such that a trajectory has a terminal reward of +1 if a patient survived, -1 if a patient died, and 0 otherwise. The observations used as our state space are a combination of time-varying continuous features (33), such as heart rate, glucose, etc., and demographic features (5), such as gender, age, weight, etc. [46].

5.2 Experiment Design

To re-iterate, there is good progress on learning RL policies for sepsis treatment using the data mentioned above. However, there is no standard way for evaluation. A

safety-critical application like this one is where our framework is the most applicable. We would like to perform high-confidence off-policy evaluation for the target policy learning offline and see if we can trust our RL policy to make decisions in the real-world.

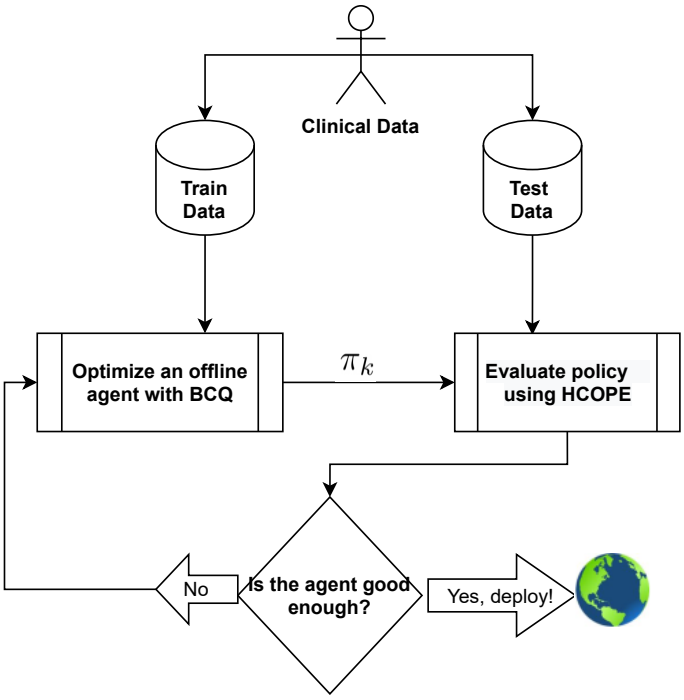


Figure 5.1: **Flowchart of how to apply our framework on sepsis data**: This figure shows the flow for how to use the sepsis data to optimize an offline policy and evaluate it with HCOPE.

To apply our framework, there are a few design decisions to make first. These decisions include: how much data to use for training/testing, bootstrapping size, confidence level, and choice of algorithm to optimize a policy offline. For the train and test dataset, we do a 70/30 train/test split using stratified sampling [46], which maintains the same proportions of each terminal outcome (survival or mortality). For the bootstrapping size, we use $B = 2000$ as in the previous chapter. We also use a confidence level $\delta = 0.05$. We optimize a policy offline using batch-constrained Q-learning (BCQ) for discrete control [30], coupled with an auto-encoder for better state representation learning. Another important decision is to decide how often we

will run our high-confidence evaluation methods because of its computational cost. For this, we train the policy for $300k$ updates, where we evaluate the policy every $10k$ updates. To estimate the behavior policy π_b, which is referred to as the clinician's policy, we rely on behavior cloning [36] as discussed in Chapter 4.

As the offline agent is training, we do not have a way to evaluate how it performs except for using off-policy estimators (OPE) and high-confidence off-policy estimators. As done in Chapter 4, we use weighted importance sampling (WIS), model-based estimator (MB), and weighted doubly-robust estimator (WDR) with bootstrapping to provide high-confidence lower-bound estimates $\hat{v_\delta}(\pi_\theta)$ for π_θ. We also report $v\pi_\theta$ using the same estimators without bootstrapping as the mean estimate. A flow chart of the training-evaluation cycle on sepsis data is shown in Figure 5.1. Data is split between training and evaluation. Train data is used to optimize a BCQ agent while test data is used for evaluation with HCOPE methods. For this experiment, we face the multi-testing problem in this experiment as we used the same data split for each evaluation iteration; this is because the size of the test data is too small to split among iterations. After evaluating, we check if the agent is good enough to deploy in the real-world, or not so we can continue improving our offline agent.

For this problem, we hypothesize that importance sampling [19] can provide good estimates given the short horizon and the fact that the target policy is constraining the action value to be close to the data distribution. The transition dynamics for this problem are challenging for the model-based estimator so we hypothesize that MB will not perform as well.

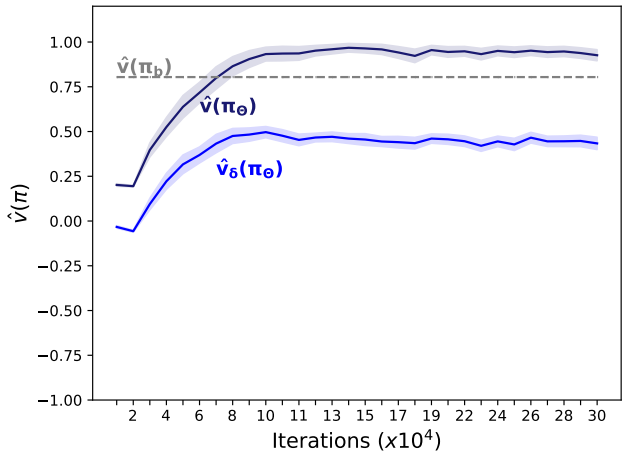
5.3 Results & Discussion

For sepsis data, there is no ground-truth to compare against, as opposed to the simulated environments we presented in Chapter 4. While learning our offline policy, we would like to see how well it performs over time and when it outperforms the behavior policy so we can deploy it into the real world. Figure 5.2 shows the results

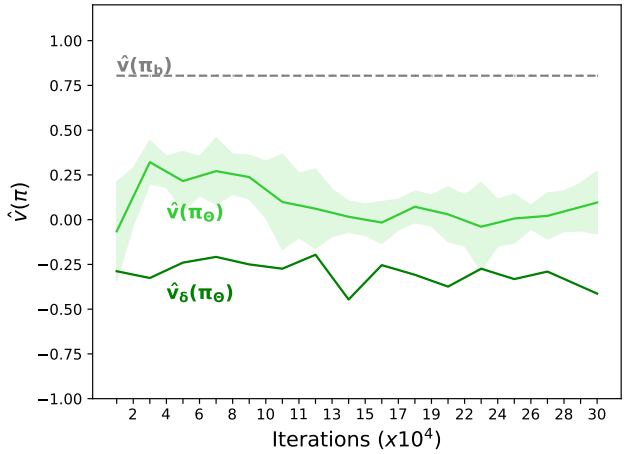
of off-policy evaluation and high-confidence off-policy evaluation estimates on the offline target policy. The value of the behavior policy, $v(\pi_b)$, is the return of the data in hand.

Figure 5.2 shows the results of training an offline RL policy and evaluating it with OPE and approximate HCOPE, as an average of 10 runs with the standard error indicated. With WIS, the lower-bound estimate $\hat{v_\delta}(\pi_\theta)$ and the mean estimate $\hat{v}(\pi_\theta)$ show the improvement of the policy over training iterations. The mean estimate of WIS can outperform the behavior policy but this can overestimate the true value. We focus on the lower-bound estimate which did not show our target policy outperforms the behavior policy. Figure 5.3 shows how the divergence of the offline target policy π_θ from $\hat{\pi}_b$ is bounded, which makes WIS estimate more reliable. The estimates by the model-based estimator (MB) failed to detect the improvement of the policy over time. This has to do with failing to model the complex dynamics of the dataset; as a result, it failed to estimate the true value of the target policy. In a model-based estimator, we model all the environment dynamics, the next state, reward, and terminal state. In MountainCar-v0, we only modeled the next state since the reward and terminal state can be computed deterministically from the current state. In Pendulum-v0, the environment always terminates at the maximum horizon, so we only modeled the next state and reward. With sepsis data, it was hard enough to model the next state because the state dimension is large; it is also hard to model the terminal state indicating when a patient terminates and whether they survive or not.

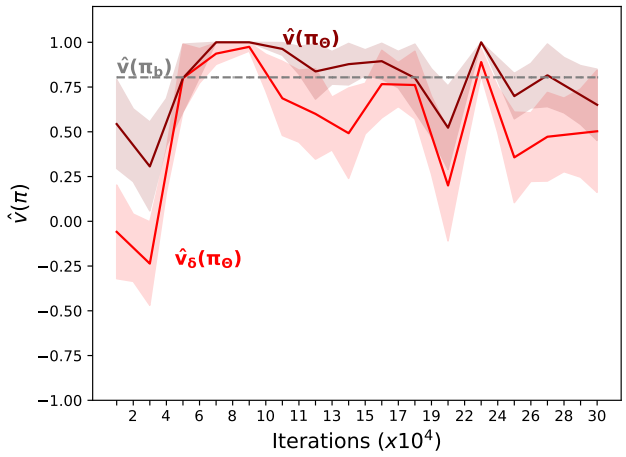
Weighted doubly-robust, values of $\hat{v_\delta}(\pi_\theta)$ and $\hat{v}(\pi_\theta)$ can show the improvement of the offline policy as it is training. WDR combines per-decision weighted importance sampling with an approximate model; the approximate model only fits q-values so it does not suffer from the modeling complexity as the MB estimator. The estimates by WDR go outside the range of typical values, which are $\langle -1.0, 1.0 \rangle$, so we clip the estimates to remain in that range. Results imply that the offline target policy can get as good as the data then the performance drops, if we rely on high-confidence



(a) WIS



(b) MB



(c) WDR

Figure 5.2: **Results of offline learning and high-confidence off-policy evaluation on sepsis data**: This figure shows results of evaluating the performance of the target policy with OPE and HCOPE as it is training offline. We show the lower-bound estimate from HCOPE $\hat{v}_\delta(\pi_\theta)$ and the OPE estimate $\hat{v}(\pi_\theta)$ without bootstrapping for each method.

estimates $\hat{v_\delta}(\pi_\theta)$ as shown by WDR. Another observation is that this offline BCQ policy may require further tuning to provide a better policy. Relying on the lower-bound estimate is important to determine whether a policy is good enough to deploy in the real world or not.

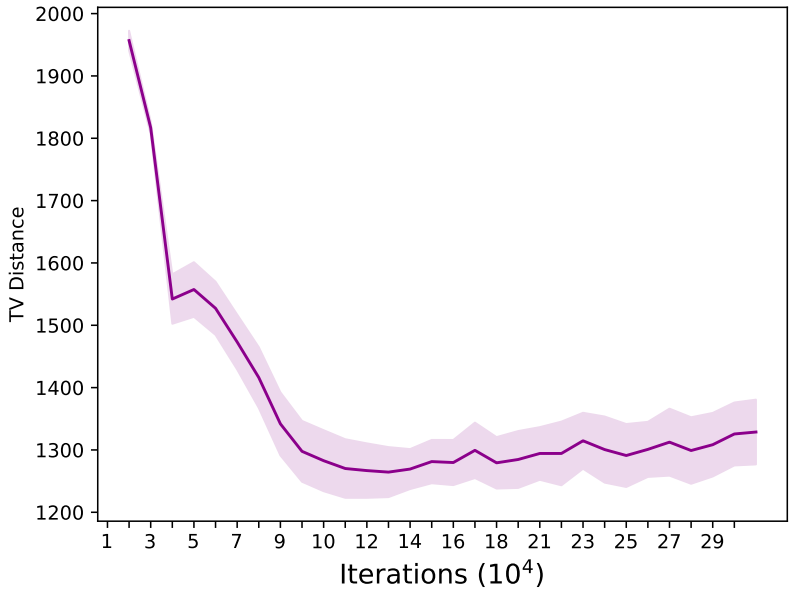


Figure 5.3: **Divergence between $\hat{\pi}_b$ & π_θ over training iterations on sepsis Data**: This figure shows the total variation (TV) distance between $\hat{\pi}_b$ & π_θ on sepsis data across training. TV distance slowly decreases over time, making it bounded and limiting the divergence of the target policy from $\hat{\pi}_b$.

For real-world sepsis data, it is possible to learn an optimal target policy offline and use OPE/HCOPE to evaluate the offline policy as it is training. We propose that relying on OPE only is not enough for a such a critical application, and it is better to rely on high-confidence estimates by HCOPE to minimize the risk of overestimation. More specifically, WIS can be reliable for this problem given the short horizon but WDR can provide better estimates as a hybrid method that improves upon importance sampling. It is challenging for the MB estimator to detect an improved policy given the complex environment dynamics. Fitted q-evaluation [22] is another good direction to use for evaluation in this problem.

Chapter 6

Challenges, Conclusions & Future Work

We conclude this thesis by discussing challenges, conclusions, and future work.

6.1 Challenges

High-confidence off-policy evaluation (HCOPE) is a feasible technique for evaluating offline RL algorithms, but there are a few challenges for the approximate HCOPE discussed below. Most of these challenges initially apply to off-policy evaluation (OPE).

Quality of Dataset: In our experiments, we tried to simulate data that mimics real-world data, such that it is suboptimal data that contains random trajectories. However, some OPE methods, such as importance sampling, become more challenging under this assumption of data quality. Since we assume no access to the policy that collected the data π_b, we estimate $\hat{\pi_b}$ with a behavior cloning model in a supervised learning manner such that each state maps to an action. Importance sampling with an estimated behavior policy [36] corrects for the errors due to sampling in the action space. Even though IS with an estimated π_b provides better estimates than ordinary importance sampling [36], it is empirically challenging to learn a good estimate when the data contains random exploratory trajectories, which is essential for learning a good offline RL policy.

Hyper-parameter Tuning: high-confidence off-policy estimators depend on hyper-parameters that should be tuned for a better performance. These parameters include bootstrapping size B, normalization of rewards and returns, as well as other hyper-parameters for the models in direct and hybrid methods. To fit a model for model-based evaluation [21] or fitted q-evaluation [22], tuning of hyper-parameters is essential, such as learning rates, batch size, number of layers, number of nodes, and the loss functions. Moreover, data usually suffers from class imbalance when modeling terminal states, or modeling success vs. failure states in real data. To make better predictions, this requires different tuning such as dataset oversampling, loss weighting, and regularization .

Modeling Assumptions: among high-confidence off-policy methods, the model-based estimator relies on a few assumptions related to how we model the environment. Simple environments/domains can be easily modeled compared to more complex domains. For instance, it is hard to model the state transitions, reward function, and terminal function altogether since errors resulting from the different models accumulate over time steps. Some of our empirical work failed to learn tight bounds with model-based estimator because of the complex environment dynamics, such as the LunarLander environment [38] or the real-world Sepsis data. MB has the potential for offline RL evaluation because if we were able to model the environment dynamics successfully, we can end up with a simulated environment which will be ideal for evaluation in the offline setting.

Bootstrapping: bootstrapping comes with heavy computational needs depending on the number of bootstrap samples B. This forces us to reduce the frequency at which we evaluate our learning policy with high-confidence. As a solution, we can rely on off-policy estimates without bootstrapping to get a sense of how the policy performs and then perform high-confidence evaluation once the mean estimates show improvement of the offline policy.

Multiple Testing Problem: this problem occurs when conducting multiple sta-

tistical tests simultaneously with the same data, so the confidence bound does not strictly hold anymore and the confidence decreases by the number of tests you perform. To avoid the multiple testing problem, more trajectories have to be allocated for evaluation to discard trajectories used in one iteration of evaluation. This is inefficient as the size of the data may be limited for real-world applications, or data can be costly. As a solution, we can only perform high-confidence evaluation once OPE estimates show improvement of the offline policy, and rely on the last iteration for the safety guarantee.

6.2 Conclusions

This thesis proposed a framework for evaluating offline RL methods with high-confidence to ensure safety. First, we studied the feasibility of different categories of off-policy estimators along with bootstrapping and how they are affected by offline learning algorithms. We propose how high-confidence evaluation is better for safety-critical applications since off-policy evaluation suffers from overestimation. So, we tested our proposed framework on real-world medical data for Sepsis treatment. While dynamically receiving data, we optimize offline RL agents and run safety tests to estimate a lower-bound on the value of the offline policy and control the risk of overestimating its true value. This is essential for safety-critical applications to be able to tell when it is safe enough to deploy a new policy. Although bootstrap confidence intervals provide approximate high-confidence estimates, they provide a practical approach with small data compared to exact concentration inequalities that may be too loose to use.

We believe safe evaluation is an important step for offline RL; offline RL has great potential for control in the actual environment, if they are good enough, where our proposed framework is most applicable. We can always use historical data to optimize an offline policy then continue learning online in the real environment; our framework can also determine when we can do that switch.

In conclusion, approximate high-confidence off-policy evaluation proved to be a

reliable evaluation method for offline agents that do not have access to a simulated environment. When combined with bootstrapping, direct and hybrid OPE methods can be trusted for testing offline agents. The horizon of the trajectory and the environment dynamics affect the choice for the off-policy estimator to rely on.

6.3 Future Work

In future work, we need to tackle learning offline from multiple data sources with different qualities and explore how the quality of the data affects offline learning and high-confidence evaluation. In addition, we can explore more OPE methods such Fitted Q-evaluation [22] to eliminate the challenges of model-based estimator. We also need to study how off-policy evaluation can work better with longer-horizon environments. We can also use different confidence intervals other than bootstrapping to mitigate its challenges.

To help high-confidence off-policy evaluation, we can use some policy convergence measures to detect to see if a policy still improving and rely more on high-confidence off-policy evaluation in later iterations of training.

Since data efficiency is essential, safety testing can be used to develop algorithms that are able to estimate how much data is needed before the policy can take over a real-world environment. Another important direction is how to do offline safe policy improvement such that we do not have to worry about the policy evaluation while safety is incorporated into the improvement part.

Bibliography

[1] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *CoRR*, vol. abs/2005.01643, 2020. arXiv: 2005.01643. [Online]. Available: https://arxiv.org/abs/2005.01643.

[2] O. Gottesman *et al.*, "Guidelines for reinforcement learning in healthcare," *Nature Medicine*, vol. 25, pp. 16–18, 2019.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, Second. The MIT Press, 2018.

[4] M. Wiering and M. van Otterlo, *Reinforcement Learning: State-of-the-Art*. Springer Publishing Company, Incorporated, 2014, ISBN: 364244685X.

[5] R. Agarwal, D. Schuurmans, and M. Norouzi, "Striving for simplicity in off-policy deep reinforcement learning," *CoRR*, vol. abs/1907.04543, 2019. arXiv: 1907.04543. [Online]. Available: http://arxiv.org/abs/1907.04543.

[6] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," *CoRR*, vol. abs/1812.02900, 2018. arXiv: 1812.02900. [Online]. Available: http://arxiv.org/abs/1812.02900.

[7] X. B. Peng, A. Kumar, G. Zhang, and S. Levine, "Advantage-weighted regression: Simple and scalable off-policy reinforcement learning," *CoRR*, vol. abs/1910.00177, 2019. arXiv: 1910.00177. [Online]. Available: http://arxiv.org/abs/1910.00177.

[8] A. Nair, M. Dalal, A. Gupta, and S. Levine, "Accelerating online reinforcement learning with offline datasets," *arXiv preprint arXiv:2006.09359*, 2020.

[9] N. Jaques *et al.*, "Way off-policy batch deep reinforcement learning of implicit human preferences in dialog," *CoRR*, vol. abs/1907.00456, 2019. arXiv: 1907.00456. [Online]. Available: http://arxiv.org/abs/1907.00456.

[10] A. Kumar, J. Fu, G. Tucker, and S. Levine, *Stabilizing off-policy q-learning via bootstrapping error reduction*, 2019. arXiv: 1906.00949 [cs.LG].

[11] Y. Wu, G. Tucker, and O. Nachum, *Behavior regularized offline reinforcement learning*, 2019. arXiv: 1911.11361 [cs.LG].

[12] C. J. C. H. Watkins and P. Dayan, "Q-learning," in *Machine Learning*, 1992, pp. 279–292.

[13] Z. Wang, N. de Freitas, and M. Lanctot, "Dueling network architectures for deep reinforcement learning," *CoRR*, vol. abs/1511.06581, 2015. arXiv: 1511.06581. [Online]. Available: http://arxiv.org/abs/1511.06581.

[14] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, PMLR, 2018, pp. 1861–1870. [Online]. Available: http://proceedings.mlr.press/v80/haarnoja18b.html.

[15] M. Bain and C. Sammut, "A framework for behavioural cloning," in *Machine Intelligence 15, Intelligent Agents [St. Catherine's College, Oxford, July 1995]*, GBR: Oxford University, 1999, 103–129, ISBN: 0198538677.

[16] C. Voloshin, H. M. Le, N. Jiang, and Y. Yue, "Empirical study of off-policy policy evaluation for reinforcement learning," *CoRR*, vol. abs/1911.06854, 2019. arXiv: 1911.06854. [Online]. Available: http://arxiv.org/abs/1911.06854.

[17] P. S. Thomas, G. Theocharous, and M. Ghavamzadeh, "High-confidence off-policy evaluation," in *AAAI*, 2015.

[18] P. S. Thomas, G. Theocharous, and M. Ghavamzadeh, "High confidence policy improvement," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15, Lille, France: JMLR.org, 2015, 2380–2388.

[19] D. Precup, R. S. Sutton, and S. P. Singh, "Eligibility traces for off-policy policy evaluation," in *Proceedings of the Seventeenth International Conference on Machine Learning*, ser. ICML '00, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, 759–766, ISBN: 1558607072.

[20] A. R. Mahmood, H. P. van Hasselt, and R. S. Sutton, "Weighted importance sampling for off-policy learning with linear function approximation," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds., vol. 27, Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper/2014/file/be53ee61104935234b174e62a07e53cf-Paper.pdf.

[21] J. P. Hanna, P. Stone, and S. Niekum, "Bootstrapping with models: Confidence intervals for off-policy evaluation," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '17, São Paulo, Brazil: International Foundation for Autonomous Agents and Multiagent Systems, 2017, 538–546.

[22] H. M. Le, C. Voloshin, and Y. Yue, "Batch policy learning under constraints," *CoRR*, vol. abs/1903.08738, 2019. arXiv: 1903.08738. [Online]. Available: http://arxiv.org/abs/1903.08738.

[23] M. Riedmiller, "Neural fitted q iteration – first experiences with a data efficient neural reinforcement learning method," vol. 3720, Oct. 2005, pp. 317–328, ISBN: 978-3-540-29243-2. DOI: 10.1007/11564096_32.

[24] N. Jiang and L. Li, "Doubly robust off-policy evaluation for reinforcement learning," *CoRR*, vol. abs/1511.03722, 2015. arXiv: 1511.03722. [Online]. Available: http://arxiv.org/abs/1511.03722.

[25] P. S. Thomas and E. Brunskill, "Data-efficient off-policy policy evaluation for reinforcement learning," *CoRR*, vol. abs/1604.00923, 2016. arXiv: 1604.00923. [Online]. Available: http://arxiv.org/abs/1604.00923.

[26] A. Kumar, A. Singh, S. Tian, C. Finn, and S. Levine, *A workflow for offline model-free robotic reinforcement learning*, 2021. arXiv: 2109.10813 [cs.LG].

[27] T. L. Paine *et al.*, "Hyperparameter selection for offline reinforcement learning," *CoRR*, vol. abs/2007.09055, 2020. arXiv: 2007.09055. [Online]. Available: https://arxiv.org/abs/2007.09055.

[28] K. Konyushkova *et al.*, "Active offline policy selection," *CoRR*, vol. abs/2106.10251, 2021. arXiv: 2106.10251. [Online]. Available: https://arxiv.org/abs/2106.10251.

[29] J. Fu *et al.*, "Benchmarks for deep off-policy evaluation," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=kWSeGEeHvF8.

[30] S. Fujimoto, E. Conti, M. Ghavamzadeh, and J. Pineau, "Benchmarking batch deep reinforcement learning algorithms," *CoRR*, vol. abs/1910.01708, 2019. arXiv: 1910.01708. [Online]. Available: http://arxiv.org/abs/1910.01708.

[31] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, 2014. arXiv: 1312.6114 [stat.ML].

[32] H. v. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16, Phoenix, Arizona: AAAI Press, 2016, 2094–2100.

[33] B. Efron, "Bootstrap methods: Another look at the jackknife," *Ann. Statist."*, vol. 7, 1979. DOI: 10.1214/aos/1176344552. [Online]. Available: https://doi.org/10.1214/aos/1176344552.

[34] J. Carpenter and J. Bithell, "Bootstrap confidence intervals: When, which, what? a practical guide for medical statisticians," *Statistics in Medicine*, vol. 19, pp. 1141–1164, May 2000. DOI: 10.1002/(SICI)1097-0258(20000515)19:9%3C1141::AID-SIM479%3E3.0.CO;2-F.

[35] A. C. Morrison *et al.*, "Prediction of Coronary Heart Disease Risk using a Genetic Risk Score: The Atherosclerosis Risk in Communities Study," *American Journal of Epidemiology*, vol. 166, no. 1, pp. 28–35, Apr. 2007, ISSN: 0002-9262. DOI: 10.1093/aje/kwm060. eprint: https://academic.oup.com/aje/article-pdf/166/1/28/17337361/kwm060.pdf. [Online]. Available: https://doi.org/10.1093/aje/kwm060.

[36] J. P. Hanna, S. Niekum, and P. Stone, "Importance sampling in reinforcement learning with an estimated behavior policy," *Machine Learning*, pp. 1–51, 2021.

[37] P. S. Thomas, "Safe reinforcement learning," Ph.D. dissertation, University of Massachusetts Libraries, 2015.

[38] G. Brockman *et al.*, *Openai gym*, 2016. eprint: arXiv:1606.01540.

[39] Y. Liu, P. Bacon, and E. Brunskill, "Understanding the curse of horizon in off-policy evaluation via conditional importance sampling," *CoRR*, vol. abs/1910.06508, 2019. arXiv: 1910.06508. [Online]. Available: http://arxiv.org/abs/1910.06508.

[40] M. Ghassemi, T. Naumann, P. Schulam, A. Beam, I. Chen, and R. Ranganath, "Practical guidance on artificial intelligence for health-care data," *The Lancet Digital Health*, vol. 1, e157–e159, Aug. 2019. DOI: 10.1016/S2589-7500(19)30084-6.

[41] Y. Sakr *et al.*, "Sepsis in Intensive Care Unit Patients: Worldwide Data From the Intensive Care over Nations Audit," *Open Forum Infectious Diseases*, vol. 5, no. 12, Nov. 2018, ofy313, ISSN: 2328-8957. DOI: 10.1093/ofid/ofy313. eprint: https://academic.oup.com/ofid/article-pdf/5/12/ofy313/33577612/ofy313.pdf. [Online]. Available: https://doi.org/10.1093/ofid/ofy313.

[42] P. E. Marik, "The demise of early goal-directed therapy for severe sepsis and septic shock," en, *Acta Anaesthesiol Scand*, vol. 59, no. 5, pp. 561–567, Feb. 2015.

[43] A. Johnson *et al.*, "Mimic-iii, a freely accessible critical care database," *Scientific Data*, vol. 3, p. 160 035, May 2016. DOI: 10.1038/sdata.2016.35.

[44] A. Raghu, M. Komorowski, I. Ahmed, L. A. Celi, P. Szolovits, and M. Ghassemi, "Deep reinforcement learning for sepsis treatment," *CoRR*, vol. abs/1711.09602, 2017. arXiv: 1711.09602. [Online]. Available: http://arxiv.org/abs/1711.09602.

[45] A. Raghu, M. Komorowski, L. A. Celi, P. Szolovits, and M. Ghassemi, "Continuous state-space models for optimal sepsis treatment - a deep reinforcement learning approach," *CoRR*, vol. abs/1705.08422, 2017. arXiv: 1705.08422. [Online]. Available: http://arxiv.org/abs/1705.08422.

[46] T. W. Killian, H. Zhang, J. Subramanian, M. Fatemi, and M. Ghassemi, "An empirical study of representation learning for reinforcement learning in healthcare," *CoRR*, vol. abs/2011.11235, 2020. arXiv: 2011.11235. [Online]. Available: https://arxiv.org/abs/2011.11235.

[47] M. Komorowski, L. Celi, O. Badawi, A. Gordon, and A. Faisal, "The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care," *Nature Medicine*, vol. 24, Nov. 2018. DOI: 10.1038/s41591-018-0213-5.