Sequential Decision Making with Risk, Offline Data and External Influence: Bandits and Reinforcement Learning

A THESIS

SUBMITTED FOR THE DEGREE OF

Doctor of Philosophy

IN THE

Faculty of Engineering

BY

Ranga Shaarad Ayyagari

Computer Science and Automation
Indian Institute of Science
Bangalore – 560 012 (INDIA)

September, 2025

Declaration of Originality

I, **Ranga Shaarad Ayyagari**, with SR No. **04-04-00-10-12-18-1-16056** hereby declare that the material presented in the thesis titled

Sequential Decision Making with Risk, Offline Data and External Influence: Bandits and Reinforcement Learning

represents original work carried out by me in the **Department of Computer Science and Automation** at **Indian Institute of Science** during the years **2018–2025**.
With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: 2025-09-20 Student Signature

In my capacity as supervisor of the above-mentioned work, I certify that the above statements are true to the best of my knowledge, and I have carried out due diligence to ensure the originality of the report.

Advisor Name: Prof. Ambedkar Dukkipati Advisor Signature

DEDICATED TO

My Parents

Acknowledgements

This thesis would not be possible without my supervisor, Prof. Ambedkar Dukkipati, who spent countless hours with us discussing research directions and ways to solve the problems that crop up during research. I would also like to thank my lab-mate Parag Dutta for keeping me abreast of the latest research trends in his field and his papers, as well as his help in generating ideas and running experiments for our collaborative work.

I would like to thank Parag, Bodhisattwa Dasgupta, and Prabhas Onteru for their ideas as well as their help with the experiments related to active reinforcement learning.

For the work on hierarchical reinforcement learning, I would like to express my thanks to Anurita Ghosh, whose efforts and coding skills were invaluable towards implementing the algorithms discussed in this thesis.

I thank Revanth for helping me with the experiments related to non-stationary reinforcement learning.

Abstract

Reinforcement Learning (RL) serves as a foundational framework for addressing sequential decision-making problems under uncertainty. In recent years, extensive research in this domain has led to significant advancements across various fields, including healthcare, recommendation systems, networks, robotics, finance, and navigation. To develop learning algorithms for these problems, RL is built upon the mathematical frameworks of Multi-Armed Bandits (MAB) and Markov Decision Processes (MDP). As opposed to supervised and unsupervised learning, sequential decision-making problems involve optimizing an objective over a time horizon and often involve either explicit or implicit planning to make optimal decisions at each step over a time period. Thus, Reinforcement Learning is a core component for building Artificial Intelligence (AI) agents that seamlessly take in information from the outside world, take decisions, and perform actions to accomplish a given task.

A major portion of the literature on reinforcement learning focuses on finding optimal algorithms to maximize the expected returns achieved by an agent that can continuously interact with an environment to gain relevant information. However, such a scenario is a very special case, and real-world problems deal with more complex objectives, as well as additional conditions and constraints imposed on the agent. The agent could be forced to operate in environments with limited data or changing dynamics. Or it could be expected to follow a policy that is risk-averse or able to plan over complex tasks. In this thesis, we study sequential decision-making problems with (i) risk, (ii) lack of online access, and (iii) under the influence of an exogenous temporal process and provide some theoretical and practical solutions. While the problem of risk is studied in the framework of bandits, the rest of the problems are studied in the framework of MDPs.

First, we consider the problem of minimizing risk in a combinatorial semi-bandit, in which an agent has to choose at each time step a subset of possible actions that satisfy given constraints. We consider the optimization objective to be the CVaR (Conditional Value-At-Risk), a risk measure that considers the worst-case returns obtained by the agent, parameterized by some risk level α. We propose algorithms to tackle this problem for the cases of (i) Gaussian rewards

and (ii) bounded rewards. We theoretically analyze these algorithms along with some practical considerations and experimentally corroborate our findings with numerical experiments in a simulated environment.

Next, we consider the problem of learning optimal RL policies when there is little or no online access to the environment. First, we consider the problem of learning a hierarchy of policies solely from an offline dataset of experiences without any access to the environment. Many sequential tasks require a degree of temporal abstraction and planning to achieve a long-term goal. To incorporate these elements, a hierarchy of policies is learned that operates at different time scales. While most works train these policies by exploring the environment, we attempt to learn such a hierarchy from a given offline dataset collected by an unknown behavior agent. We propose a model-based algorithm that uses a Conditional Variational Auto-Encoder (CVAE) along with an ensemble-based uncertainty metric to apply existing online RL algorithms on offline data. On various continuous control and robotic manipulation tasks, we show that this method reliably preserves the benefits of a hierarchical agent even without online exploration.

In the offline setting, we also solve the problem of active learning in RL, where the agent is given an offline dataset and limited access to the environment are available. We tackle the problem of optimally augmenting the given offline dataset with minimal extra exploration. The performance of a policy learned from an offline dataset is limited by the quality of the data in the dataset. In many circumstances, there is an opportunity for an agent to flexibly collect additional trajectories in the environment, although the number of samples collected needs to be limited due to issues related to cost, etc. To this end, we propose an active reinforcement learning framework that uses a representation-based uncertainty model to optimally augment the existing dataset to minimize overlap and maximize the information gain due to new samples. We demonstrate our proposed method on various continuous control environments such as Gym-MuJoCo locomotion environments as well as Maze2d, AntMaze, CARLA and IsaacSimGo1.

Finally, we consider the problem of dealing with a changing environment due to the effect of exogenous events. We introduce a notion of non-stationarity wherein the dynamics of a Markov Decision Process are affected by the influence of an exogenous temporal event process. We show that under suitable conditions on the decay of this external process, the sequential decision problem can be solved by a tractable policy within ϵ-suboptimality. More specifically, when the perturbations caused by events older than t time steps on the MDP transition dynamics and the event process itself are bounded in total variation by M_t and N_t respectively, where $\sum_t M_t, \sum_t N_t$ are convergent series, we show that the problem has tractable sliding-window solutions whose approximation error ϵ depends on M_t and N_t. We propose a policy iteration algorithm and analyze its properties as a function of the properties of the external events. Fur-

ther, we analyze the sample complexity of an extension of the Least-Squares Policy Evaluation algorithm applied to this setting and demonstrate the results experimentally.

Contents

List of notations and abbreviations

Standard mathematical notation.

Notation	Description
$\mathbb{I}\{\cdot\}$	Indicator function: $\mathbb{I}\{p\}$ is 1 when proposition p is true, 0 otherwise.
$\mathbb{N}$	Set of natural numbers, $\{1, 2, 3, \dots\}$.
$\mathbb{R}$	Set of real numbers.
$\mathcal{X} \backslash \mathcal{Y}$	Set difference: Set of elements of $\mathcal{X}$ that are not in set $\mathcal{Y}$.
$\lvert\mathcal{X}\rvert$	Cardinality of set $\mathcal{X}$.
inf	Infimum of a given set
sup	Supremum of a given set
$\lvert\lvert \cdot \rvert\rvert_2$	l_2 norm of a vector.
$[N]$	The set $\{1, 2, \dots, N\}$ for some $N \in \mathbb{N}$.
$\mathsf{E}[\cdot]$	Expectation with respect to some distribution.
$\mathsf{P}(\cdot)$	Probability of a given event with respect to some distribution.
$\Delta(\cdot)$	Class of probability distributions over a given set
$\oplus$	Binary sum of two probability distributions i.e, the distribution arising from adding two independent random variables sampled respectively from two probability distributions
$\bigoplus$	N-ary sum of probability distributions
$\mathcal{N}(\mu, \sigma^2)$	Normal distribution with mean μ and variance σ^2
Φ	Cumulative distribution function of $\mathcal{N}(0, 1)$
φ	Probability density function of $\mathcal{N}(0, 1)$
χ^2	Chi-squared distribution
$\mathsf{Mean}(\cdot)$	Mean of a given distribution
$\mathsf{Variance}(\cdot)$	Variance of a given distribution

$\mathsf{Supp}(\cdot)$	Support of a given distribution
$\mathsf{VaR}_\alpha(\cdot)$	Value-at-Risk of given distribution at risk level α
$\mathsf{CVaR}_\alpha(\cdot)$	Conditional Value-at-Risk of given distribution at risk level α

Notation related to multi-armed bandits.

Notation	Description
$\mathcal{R}$	Regret
T	Time horizon
K	Number of arms
L	Maximum size of a super-arm in a combinatorial bandit
Δ	Performance (risk/expectation) gap between the optimal arm and other arms

Notation related to reinforcement learning.

Notation	Description
$\mathcal{M}$	A Markov Decision Process (MDP)
$\mathcal{S}$	State space of an MDP
$\mathcal{A}$	Action space of an MDP
r	Reward function
Q	Transition kernel of an MDP
γ	Discount factor
π	A policy that acts in an MDP
π^*	Optimal policy
V^π	Value function of policy π
V^*	Optimal value function
Q^π	Action-value function of policy π
Q^*	Optimal Q function
$\mathcal{T}$	Bellman operator
$\mathcal{D}$	Dataset

Abbreviations.

Notation	Description
MAB	Multi-Armed Bandit
MDP	Markov Decision Process
P-MDP	Pessimistic MDP
D4RL	Datasets for Deep Data-Driven Reinforcement Learning
PPO	Proximal Policy Optimization
MLP	Multi-Layer Perceptron
ReLU	Rectified Linear Unit
DOF	Degrees Of Freedom
AC	Active Collection
FT	Finetuning
Bandit algorithms	
UCB	Upper Confidence Bound
CUCB	Combinatorial Upper Confidence Bound
MARAB	Multi-Armed Risk-Aware Bandit
SDCB	Stochastically Dominant Confidence Bound
RL algorithms	
MOReL	Model-Based Offline Reinforcement Learning
ActiveORL	Active Offline Reinforcement Learning
OC	Option-Critic
MOC	Multi-Updates Option Critic
CVAE	Conditional Variational Auto-Encoder
SAC	Soft Actor-Critic
HER	Hindsight Experience Replay
BC	Behavior Cloning
UOF	Universal Option Framework
DDPG	Deep Deterministic Policy Gradient
TD3	Twin-Delayed Deep Deterministic Policy Gradient
CQL	Conservative Q-Learning
IQL	Implicit Q-Learning

BPPO	Behavior Proximal Policy Optimization

Chapter 1

Introduction

Reinforcement Learning provides a class of learning algorithms to create AI agents that are capable of taking decisions to achieve long-term goals. These agents take decisions or actions in an environment and are trained by being given positive or negative reinforcement or punishment that acts as feedback on whether the decision was right or not.

This is quite different from other fields of machine learning. In supervised learning, the agent is given a set of examples, each of which consists of a "question" (generally a feature vector or data point), and an "answer" (the label or target value), with the goal being predicting the target given the input. However, reinforcement learning does not need access to the correct target for each data point. All it needs is a noisy or even partial feedback whether the decision of the agent is in the right direction or not. This results in a more general class of problems that come under this umbrella. On the other end of the spectrum, unsupervised learning deals with recognizing patterns in a given dataset to make informed choices, with no objective metric to decide the performance of the agent, save its own biases and the practitioner's model of the data. In contrast, a reinforcement learning agent still has to ultimately optimize its obtained rewards, thereby performing the task intended by the practitioner who designed the rewards. This gives flexibility to design an easily determined reward without requiring the right decision to be known. These learning paradigms only address a narrow subset of problems associated with learning intelligent agents. While unsupervised learning helps analyze and simplify large real-world datasets, helping to learn and replicate patterns present in the data, supervised learning assists humans in making decisions on individual data points.

However, developing general AI agents necessitates learning algorithms that take decisions and perform actions interactively in an environment. The agents should learn to take actions based on the past experiences of interactions with the environment, while also taking into account the effect of these actions on future rewards. Such problems come under the umbrella

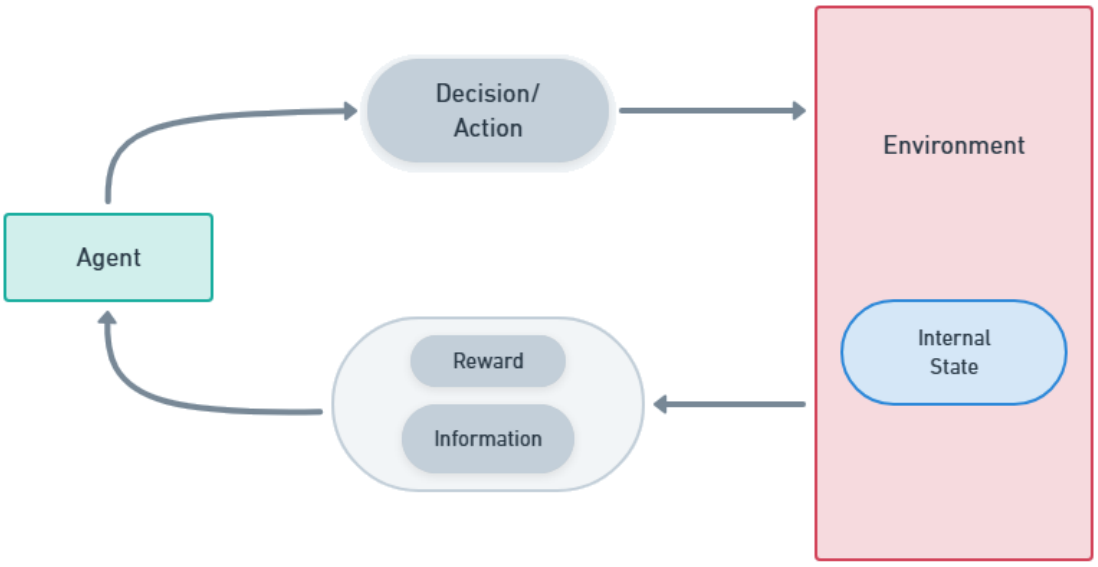


Figure 1.1: Sequential Decision-Making. The agent iteratively takes actions in the environment, which provides feedback for each action.

of *Sequential Decision-Making*.

Many sequential decision-making problems have been studied under the multi-armed bandit framework, in which an agent has to choose from a fixed set of arms and is rewarded stochastically based on its actions. While they are simpler than other forms of sequential decision-making, Bandits and their generalizations have been found to be useful for a wide variety of practical applications, such as healthcare for modeling treatment allocation, studying influence in social networks, recommendation systems, cognitive radio networks, stochastic versions of problems such as the shortest path problem, etc. In these problems, the performance of a strategy is quantified as the expected total sum of rewards obtained, or using the "regret", defined as the expected difference between the rewards obtained by an oracle with perfect information about the reward distributions of all arms, and the rewards obtained by the agent.

Most often, bandit problems have been studied in the online setting, wherein the agent has to choose optimal actions throughout an episode, while not being aware of the underlying distribution of rewards. Either implicitly or explicitly, a good strategy has to combine the exploration of the action space to gather information about the reward distributions, and exploitation of the apparent optimal arm. Variants of algorithms built on this idea, such as the Upper Confidence Bound algorithm (Auer et al., 2002a) and Thompson sampling (Agrawal and Goyal, 2012), have all but solved the vanilla version of the multi-armed bandit under various distribu-

tional constraints on the reward distributions, such as Gaussian rewards (Auer et al., 2002a), Bernoulli and bounded rewards (Garivier and Cappé, 2011), and heavy-tailed rewards (Bubeck et al., 2013). However, more general versions of the bandit problem that aim to incorporate privacy (Mishra and Thakurta, 2015), causality (Lattimore et al., 2016), federation (Shi and Shen, 2021), etc., still constitute an active research area.

While we discuss sequential decision-making problems, it is worthwhile to recall that the standard criteria for judging machine learning models or algorithms are expressed in terms of an optimization of the probabilistic expectation of a performance metric. This extends to multi-armed bandits and reinforcement learning in general, where one wishes to train agents that optimize the expected rewards obtained. Although this formulation is useful and effective in addressing many practical learning problems, there exist situations in which one wishes to take into consideration more information about the distribution of the performance metric, in addition to just its expectation. The aim is to prevent scenarios wherein a model performs well on average, but breaks down in unexpected ways, which leads to the problem of mitigating risk. This objective is captured by various risk measures.

A straightforward extension to the expectation is a risk measure that considers a combination of the expectation and the variance of a distribution, such as the Mean-Variance criterion (Sani et al., 2012) and Sharpe ratio (Sharpe, 1966). Such measures incentivize learning an agent that takes care of the volatility of the obtained rewards. In our work, we consider the Conditional Value-at-Risk (CVaR), a risk measure that is defined as the expected worst-case reward obtained by the agent. What constitutes a worst-case scenario is parameterized by some quantile α.

In problems such as influence maximization and cognitive radio networks, the agent has to choose a subset of available arms at each time step out of all the available arms. The reward obtained by the agent is thus based on this subset of arms, called a super-arm. To tackle such problems, Gai et al. (2010) formulated the combinatorial bandit problem and proposed an algorithm for the problem of optimally allocating channels to users in a cognitive radio network. This work was generalized to the LLR (Learning with Linear Rewards) algorithm (Gai et al., 2012) for combinatorial bandits in which each super arm is characterized as a linear combination of individual arms. Further works study this problem for more general reward and distributional settings.

> **Problem:** How does one incorporate risk-aversion in combinatorial semi-bandits? And how does this affect the regret and computational complexity of the algorithm?

In this thesis, we consider the problem of optimizing the CVaR regret of an agent for tackling

a combinatorial semi-bandit. We propose two algorithms to solve this problem for the case where the arms have Gaussian or bounded rewards, and the super-arm rewards are a sum of the constituent arm rewards. We provide bounds on the regret of these algorithms and discuss the performance and computational effects of discretizing our algorithm for the arbitrary bounded rewards case.

In the problems discussed above, it is assumed that the agent operates in a fixed environment and only needs to consider the rewards obtained due to taking one of many possible decisions. However, in many scenarios, the actions of the agent directly affect and change the state of the environment, which have to be taken into account when taking a decision at any time step. In other words, the agent has to not only consider the current state of the environment, but also possible future states resulting from its actions.

A generalization of a multi-armed bandit that captures this problem is a Markov Decision Process (MDP), wherein the agent is considered to be in an environment with an inherent "state". This state evolves in a Markov manner based on the previous state and the action taken by the agent, and the reward obtained is a function of the current state, current action, and the resultant next state. As a class of problems, being a more general setting, MDPs cover a wider range of applications that are more complex, including but not limited to video games (Mnih et al., 2015), visual vehicular navigation (Huang et al., 2021b), robotic manipulation (Han et al., 2023) and drone control (Song et al., 2021; Kaufmann et al., 2023), to name a few.

Algorithms for solving MDPs have been proposed for a wide variety of scenarios, depending on the cardinality and nature of the state and action spaces, the class of policies allowed, etc. Theoretically, many works consider the case where the state and action sets are finite (Dann and Brunskill, 2015; Domingues et al., 2021; Li et al., 2024). Much work has also been done to analyze linear MDPs (Jin et al., 2023) and approximations based on classes of functions with bounded complexity (Ayoub et al., 2020; Wang et al., 2020), neural networks (Huang et al., 2021a; Dong et al., 2021), etc.

However, in many cases, policies learned using standard reinforcement learning algorithms struggle to plan over long-horizon tasks that consist of multiple steps. This is due to various reasons, such as a sparse reward signal that requires extensive exploration and the need to learn and compose diverse behaviors to execute the aforementioned complicated tasks.

Various techniques have been studied to tackle this problem of long-term planning over complex sub-tasks. One such technique is the Options framework (Sutton et al., 1999), in which the agent learns multiple low-level policies, called options, controlled by a single high-level policy. The high-level policy chooses one of the options to be executed based on the current state, and the control is passed to the chosen option. This option then continues to be in control

until it decides to terminate. The control is then passed to the high-level policy, which then chooses an option again, and so on. The Option-Critic (OC) architecture (Bacon et al., 2017) is a method to discover and learn options that are parameterized by deep neural networks and learned using policy gradient updates. Further, many variants of the Option-Critic have been studied that aim to improve certain aspects of the algorithm (Khetarpal et al., 2020; Zhu et al., 2021; Smith et al., 2018).

Another important aspect of reinforcement learning algorithms in complex settings such as robotics is the issue of high sample complexity. This is especially relevant for hierarchical algorithms that need a lot of exploration to learn multiple levels of policies that interact with each other. This led to the study of reinforcement learning algorithms that do not need access to an environment to explore and try different strategies. Instead, they just need access to a fixed dataset of samples collected from the environment by some possibly unknown behavior policy. Such methods come under the category of Offline Reinforcement Learning (Levine et al., 2020; Prudencio et al., 2023), since they only use offline data and have no need for online access to the environment.

The offline setting poses new challenges, such as distributional shift, lack of samples in many regions of the environment, sub-optimality of the behavior policy, etc. While many algorithms that deal with these problems have been studied (Fujimoto et al., 2019; Kumar et al., 2020; Yu et al., 2020; Kidambi et al., 2020), the study of hierarchical algorithms that can deal with offline data is limited.

> **Problem:** How does one learn hierarchical policies using a given offline dataset with the same flexibility possible in the online setting?

To solve this problem, we propose a method to convert a given online hierarchical learning algorithm such that it can operate on a given offline dataset. While many works exist to learn hierarchical agents in the offline setting, our proposed approach offers a flexible and general learning algorithm to leverage existing online hierarchical algorithms for constructing offline algorithms. Based on the task and environment, different hierarchical RL algorithms may have different objectives and properties to be satisfied by the agent. Instead of redesigning each algorithm to work in the offline setting, our method allows one to apply the algorithms directly on the offline dataset, thereby preserving the characteristics of the resultant agents.

In addition to learning an agent from a given offline dataset, one might wish to augment the dataset with more trajectories to learn a better policy. This is especially important when the given dataset results in drastically sub-optimal policies, either due to it being formed using a sub-optimal policy, or because of lack of sufficient coverage of the environment.

Extending an offline dataset can be accomplished in a straightforward manner using any online algorithm for exploration, such as SAC (Haarnoja et al., 2018). However, one wishes to minimize the amount of exploration in the environment and further explore in such a way as to complement the offline data already available, due to cost concerns. In fact, such concerns are why an offline dataset is used in the first case, instead of training an agent online from a *tabula rasa* state. This motivates the development of strategies to explore the environment in such a way that minimizes the number of additional samples collected while maximizing the performance of the final agent.

In the realm of supervised learning, the issue of determining effective methods for data collection is referred to as Active Learning (Cohn et al., 1996; Settles, 2011; Bachman et al., 2017). In this scenario, the agent operates with a limited quantity of labeled data alongside a vast pool of unlabeled data, which incurs significant costs for annotation. The primary goal is to select a small subset of unlabeled data for labeling, thereby enhancing the performance of a model trained on this enriched labeled dataset.

This task becomes increasingly complex in the context of sequential decision-making problems, as the data is represented by samples and trajectories that remain unknown until the agent engages with the environment through an exploration policy. Consequently, rather than merely selecting which data points to label, the agent must decide where, how, and to what extent to explore the environment.

> **Problem:** How does one optimally augment a given offline dataset such that the resultant reinforcement learning agent achieves the best performance on the task under consideration, while minimizing the cost associated with additional online collection of data in the environment?

We propose a method to solve this problem that uses representation models to estimate epistemic uncertainty of the agent in different regions of the state space of the MDP, and uses these estimates to optimally start, explore, and stop online data collection. This reduces the cost incurred by the agent while still maximizing the final performance during deployment.

In the aforementioned settings, the decision-making problem is a fixed one, with a certain unchanging objective and a fixed rule by which the states evolve stochastically based on the actions taken by the agent. However, this is not the case in real life. During deployment of the agent, based on the context, the environment may undergo a variety of changes, ranging from continuous small changes to occasional drastic changes.

Consider the example of portfolio management in finance. An agent has access to a state consisting of various fundamental and technical indicators of a number of financial instruments

and has to decide whether to hold on to an instrument, sell it, short it, etc. However, the price movements may be affected by external forces such as government decisions like rate hikes or sudden market movements caused by events occurring in a different sector of the economy that are not being considered by the agent. Such external events might exert lasting influence over different time scales depending on the kind of events that occurred. A rate cut might affect stock prices for a few months, while a flash crash might cease its effect in a few hours.

As another example, consider a navigation problem in which an agent has control of a robot and needs to travel from one point to another. While the agent can learn to do this task in ideal conditions, the optimal actions to be taken by the agent might depend on external conditions such as rain, sudden traffic changes, or human movement patterns.

Problems:

(i) How can one effectively model the influence of external temporal events on the dynamics of MDPs?

(ii) What role do historical events play in estimating value functions, and how can one quantify them?

(iii) How does the agent learn policies when the environment is subject to external temporal influences?

In this thesis, we address the non-stationarity induced in the MDP due to the influence of an exogenous (non-Markovian) temporal event process. More formally, along with an MDP $\mathcal{M}$, we consider an external discrete-time temporal event process that is independent of $\mathcal{M}$ but affects its transition probabilities. Associated with each event is a probabilistic mark or feature vector. We investigate the properties of the event process and its effect on the feasibility and tractability of finding a good policy in this non-stationary environment.

What is this thesis about?

From the previous discussion, one can see that developing RL algorithms to solve sequential decision-making problems involves consideration of various constraints such as risk, lack of online access, and influence due to exogenous events. This is essential for deploying agents in practical situations.

In the first part of this thesis, we analyze the problem of designing risk-averse agents for combinatorial semi-bandits. We propose algorithms to tackle this problem and theoretically analyze and contextualize their performance and computational complexity.

In the next two parts of this thesis, we consider the problem where the agent is given an offline dataset of experiences and has little to no online access to the environment. In the first part, we study how this difficulty can be overcome to learn hierarchical policies. In the second part, we tackle the problem of minimally exploring the environment to optimally augment the given offline dataset.

In the final part of the thesis, we consider the problem of dealing with non-stationary scenarios in reinforcement learning. We analyze the feasibility of finding tractable solutions to a class of non-stationary MDPs, and analyze a variant of the policy iteration algorithm designed for this setting.

1.1 Revisiting the essentials

In this section, we give an intuitive description of various concepts related to the thesis.

1.1.1 What is Sequential Decision-Making?

In a sequential decision-making problem, an agent is placed in an environment and has to choose a decision or action a from an available action set $\mathcal{A}$ at each time step $t \in \{0, 1, \dots\}$. Based on the action a_t taken at time t and the properties of the environment at that time step, the agent obtains a reward r_t from the environment.

Based on the setting, and taking into account the effect of its actions on the environment as well as the future plans of the agent, a strategy has to be followed for choosing the actions that maximizes the cumulative rewards obtained by the agent.

When the time horizon is finite, this cumulative objective is generally taken as a sum of rewards $\sum_t r_t$ obtained. For an infinite planning horizon, a discounted sum of rewards $\sum_t \gamma^t r_t$ is considered, from some $\gamma \in (0, 1)$.

1.1.2 Multi-Armed Bandits

A multi-armed bandit is the simplest version of a sequential decision-making problem. At each time step, the agent is given the option of choosing one from a fixed set of K decisions or actions, also called arms. Each arm i corresponds to a probability distribution $\mathcal{P}_i$ on the reals. When an arm is chosen, the agent receives a sample from this distribution as a reward.

The goal of the agent is to strategize and choose arms so as to maximize the expected total amount of reward obtained.

In an ideal scenario in which the agent knows the exact distributions of the rewards of all arms beforehand, it can simply always choose the arm with the best expected reward. But in general, the agent does not know the rewards and thereby makes suboptimal decisions. The

expected difference in rewards obtained by a fully knowledgeable oracle and the agent under consideration is called the "regret". The goal of multi-armed bandit algorithms is to minimize the regret, which is equivalent to maximizing the rewards obtained.

Since the agent has no prior knowledge of the reward distributions of the arms, it has to estimate the distributions from the reward samples obtained. This leads to a class of algorithms that initially "explore" the reward landscape by choosing each arm a few times to collect some reward samples. Once sufficient samples are collected to make justified choices, the agent can then "exploit" by choosing arms with the highest presumed expected reward.

This exploration-exploitation trade-off can be solved in different ways. The simplest method, called the Explore-Then-Commit algorithm, is to initially pick each arm for a fixed number of times for exploration, and then stick to exploiting the best-performing arms. A more sophisticated way, called the Upper Confidence Bound algorithm (Auer et al., 2002a), is to implicitly solve this trade-off by making decisions based on the upper confidence bounds on expected reward from each arm, calculated using reward samples obtained till then. A brief overview of bandits and standard algorithms is provided in Section 2.1. For a rigorous and in-depth presentation of various algorithms and bounds on their performance, refer to Lattimore and Szepesvári (2020).

1.1.3 Markov Decision Processes

A Markov Decision Process is a generalization of a multi-armed bandit wherein the agent is placed in an environment that is in a "state" $s \in \mathcal{S}$ at each time step.

When the agent takes an action $a \in \mathcal{A}$, it obtains a reward $r(s, a)$, and the environment state transitions to a new state $s' \sim P(.|s, a)$, based on a transition kernel P that depends on the state and action.

In this work, we mainly consider infinite-horizon discounted MDPs, wherein this decision process repeats infinitely. While the goal of the agent is still to maximize rewards, since there are infinite rewards in this setting, the objective is to maximize a discounted sum of rewards, with some discount factor $\gamma \in (0, 1)$. That is, the goal of the agent is to determine a policy $\pi : \mathcal{S} \to \Delta\mathcal{A}$ of choosing actions that maximizes the expected discounted sum of rewards obtained, as follows:

$$\pi^* = \arg\min_{\pi \in \Pi} \mathsf{E}_{\substack{s_0 \sim \rho \\ a_t \sim \pi(.|s_t) \\ s_{t+1} \sim P(.|s_t, a_t)}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right],$$

where $\rho \in \Delta\mathcal{S}$ is some initial distribution of states and Π is the class of allowed policies.

In Section 2.2.1 of Chapter 2, we provide an overview of some algorithms to approximately solve this optimization problem. Comprehensive discussions of standard algorithms for various

settings of MDPs can be found in Sutton and Barto (1998) and Bertsekas (2012).

1.1.4 What do we mean by Risk?

In supervised learning and Bayesian decision theory, risk refers to the expected loss incurred by an algorithm based on a classification or decision made by the agent following the algorithm, with the expectation taken with respect to the data available as well as the stochasticity of the "query" on which the decision is made. However, in the context of this thesis, we consider a slightly different notion of risk used in the sequential decision-making literature, which is a way to generalize the performance of an agent beyond expectations of random variables.

For example, a straightforward way to incorporate volatility of the rewards into the objective is to consider the variance of the reward distributions along with the means. This leads to the Mean-Variance criterion, defined as the difference between the mean and variance of the obtained rewards, and the Sharpe Ratio, defined as the ratio of the mean and the standard deviation.

In this work, we study the Conditional Value-at-Risk (CVaR), which quantifies the expected worst case reward based on some parameter α that defines what is considered as "worst case". Intuitively, the CVaR of a random variable at risk level $\alpha \in (0, 1)$ is the expected value of the random variable conditioned on it being from the α-quantile of its probability distribution. Further technical details on risk metrics and how they are incorporated in multi-armed bandit problems are provided in Section 2.3.

1.1.5 Temporal Processes

A stochastic process is a set of random variables $\{X_t : t \in I\}$, where I is an index set. When t denotes time, (X_t) describes the evolution of a process and is referred to as a temporal process.

A temporal point process generally refers to a temporal process indexed on some interval $I \subseteq \mathbb{R}$. In this thesis, we study the properties of a Markov Decision Process under the influence of an exogenous temporal event process. While the temporal process considered is general, assumptions regarding its nature are motivated by the properties of a Hawkes process, which is a self-exciting counting process. That is, at any time t, it describes the number of "events" that have occurred till that time, and it is self-exciting since the occurrence of an event at a particular time affects the rate of occurrence of events in the future. Generally, this effect of each individual event on future events is considered to decay with time.

Since most reinforcement learning algorithms are formulated as Markov Decision Processes, in which an agent takes actions at discrete time steps, we consider discrete-time processes as the exogenous processes that perturb the MDP. Further, as an example, we study a discrete-time

analogue of the Hawkes process and its effect on the reinforcement learning problem. Technical details of such temporal processes can be found in Section 2.4.

1.2 Summary of contributions

We begin by studying risk in combinatorial bandit problems. We then move on to Markov Decision Processes and study algorithms that operate in the offline setting. Finally, we study more general non-stationary environments and how to tackle them.

1.2.1 Risk-Averse Combinatorial Semi-Bandits

Consider a multi-armed bandit with K arms and time horizon T. Let $\mathcal{A} \subseteq 2^{[K]}$ be a constraint set. At each time step, the agent is allowed to pick a "super-arm" $a \in \mathcal{A}$, and is awarded a reward that is the sum of rewards sampled from all arms in the super-arm. Suppose the maximum super arm size $L = \max_{a \in \mathcal{A}} |a|$.

We assume semi-bandit feedback for this combinatorial bandit, which means that at each time step, the agent only has knowledge of reward samples from those arms in the chosen super-arm. We propose and analyze algorithms that aim to minimize the CVaR_α regret from interacting with this stochastic combinatorial semi-bandit.

1. First, we consider the case where the rewards from each arm are Gaussian. Assuming knowledge of upper and lower bounds on the variance, we propose an algorithm to solve this problem whose cumulative CVaR regret is upper bounded by $\mathcal{O}\left(L^2 K \log T\right)$.

2. Next, we consider the case where the rewards from each arm are non-negative and bounded above by a known upper bound. We propose an algorithm that solves this problem and show that its regret is upper bounded by $\mathcal{O}\left(L^3 \log T + K\right)$.

3. We then study a discretization of the previous algorithm to reduce its worst-case computational complexity without incurring any additional regret.

4. Finally, we provide experimental validation of the proposed algorithms in simulated environments with Bernoulli and Gaussian rewards.

1.2.2 Hierarchical Reinforcement Learning with Offline Data

We consider the problem of learning hierarchical policies when the only data accessible to the agent is an offline dataset collected in the environment by an unknown, possibly sub-optimal behavior policy.

1. We propose a general framework that can be used to convert an online hierarchical reinforcement learning algorithm for learning options to learn solely from offline data.

2. We provide experimental validation of this framework on MuJoCo locomotion tasks in the standard and in the transfer setting.

3. We then validate the suitability of our framework for planning tasks by learning a hierarchical agent in robot gripper block-stacking tasks.

4. We provide ablations and analyses to show the relevance of each component of the framework to learn hierarchical agents that can leverage offline data for task transfer and high-level planning.

1.2.3 Active Learning in the Context of Offline Data

Next, we consider the problem of augmenting a given offline dataset in an efficient manner to minimize the cost to collect additional trajectories, as well as the deterioration of the learned policy due to abrupt out-of-distribution samples.

1. We propose a representation-aware epistemic uncertainty-based method for determining regions of the state space where the agent should collect additional trajectories.

2. We propose an uncertainty-based exploration policy for online trajectory collection that reuses the representation models.

3. Through extensive experimentation, we empirically demonstrate that our approaches can be widely applicable across a range of continuous control environments. Our active trajectory collection method reduces the need for online interactions by up to 75% when compared to existing fine-tuning approaches.

4. We also perform ablation experiments to demonstrate the importance of each component of our algorithm.

1.2.4 Reinforcement Learning under External Influence

Finally, we consider reinforcement learning in an environment that is not static, but is in fact constantly changing under the influence of external events. We study this problem in a general setting, where an MDP is perturbed by an external discrete-time temporal process, affecting its dynamics in a non-Markovian fashion. This process, being exogenous to the decision process, does not depend on the state of the environment under consideration or the actions of the agent. Our contributions are as follows:

1. We outline sufficient conditions that ensure the existence of a well-defined solution for this problem. Then, we establish criteria under which an approximate solution can be determined using only the current state and finite history of past events. We show that this is possible when the perturbations caused by events older than t time steps on the MDP transition dynamics and the event process itself are bounded in total variation by M_t and N_t respectively, and $\sum_t M_t, \sum_t N_t$ are convergent series. We analyze the trade-off between the performance of the agent and the length of history considered.

2. We propose a policy iteration algorithm and theoretically analyze its behavior. We show that policy improvement occurs in regions of the state space where the Bellman error is "not too low" based on M_t and N_t. Based on our analysis, we observe that the higher the approximation error due to non-stationarity, the smaller the region where the policy is guaranteed to improve.

3. We then analyze the sample complexity of policy evaluation in this setting. For the path-wise Least Squares Temporal Difference algorithm with linear function approximation, we extend results in the stationary setting (Lazaric et al., 2012) to study the impact of non-stationarity on the expected error of the learned value function. We show that with high probability, the expected error of the value function estimate decomposes into approximation error due to discarding old events, inherent error due to linear function approximation, and the standard error terms due to stochasticity and mixing that are present in the stationary case.

4. We further explain our theoretical results using the discrete-time Hawkes process and conduct experiments in a non-stationary pendulum environment to validate our results.

1.3 Organization of the thesis

Chapter 2: In this chapter, we discuss basic concepts in multi-armed bandits and reinforcement learning upon which the contributions of this thesis stand.

Chapter 3 We begin by tackling the problem of incorporating risk-aversion in a combinatorial multi-armed bandit. We propose and theoretically analyze bandit algorithms that minimize a notion of regret that takes into account the Conditional Value-at-Risk (CVaR), under Gaussian and bounded reward distributions. We present results of simulation experiments that justify our approach as opposed to existing baseline algorithms.

Chapter 4 In the next two chapters, we consider the problem of learning an agent that optimally operates in a Markov Decision Process (MDP) environment by mainly relying on an available fixed set of experiences that have been collected by an unknown policy. In this chapter, we propose a way to learn hierarchical agents that can incorporate planning at different time scales using solely this offline dataset. We empirically establish the validity of our algorithm on a variety of continuous control tasks.

Chapter 5 In this chapter, we consider the problem of minimally augmenting the given offline dataset so as to learn an optimal policy. Motivated by active supervised learning, we propose a framework to actively select regions of the state space from which to explore, along with an optimal exploration strategy. We benchmark and analyze our algorithm on a wide variety of continuous control tasks.

Chapter 6 This chapter deals with the issue of dealing with changing environments under the influence of exogenous stochastic processes. We formulate this problem and characterize the conditions under which it has an optimal as well as a tractable ϵ-suboptimal solution. We then propose and theoretically analyze the behavior of algorithms that take a step towards solving this problem. Finally, we present simulation experiments to demonstrate our results in practice.

Chapter 7 We make a few concluding remarks summarizing the importance of the problems studied in this thesis and our contributions towards solving them.

Chapter 2

Background and Preliminaries

In a sequential decision-making scenario, an agent is placed in an environment and has to take a sequence of decisions or actions. At each time step, it receives feedback from the environment based on the action taken. Based on this feedback, the agent has to carefully plan the actions to take and the sequence in which they are taken, so as to optimize some performance criterion over a time horizon.

2.1 Multi-Armed Bandits

A special case of sequential decision-making is formalized by the framework of multi-armed bandits. In this setting, an agent is set in a decision-making environment for T time steps, called the time horizon. At each time step, it is allowed to choose between K arms, $[K] = \{1, 2, \ldots, K\}$.

Each arm $i \in [K]$ has an associated probability distribution $\mathcal{D}_i$ for its reward. When an agent chooses arm i, it obtains a reward $r \sim \mathcal{D}_i$ sampled from its corresponding reward distribution. Generally, $\mathcal{D}_i$ is assumed to belong to some class of probability distributions, such as Gaussian, Bernoulli, or simply having some bounded support $[a, b] \subset \mathbb{R}$.

The aim of the agent is to maximize the expected reward obtained over the course of the entire time horizon $\{1, 2, \ldots, T\}$.

Let $\mu_i \in \mathbb{R}$ be the expected reward obtained from the reward distribution $\mathcal{D}_i$ of arm i. Let i^* be the arm that has the highest expected reward.

$$i^* = \underset{i \in [K]}{\operatorname{argmax}}\, \mu_i.$$

An agent that has prior knowledge of the reward distributions $\mathcal{D}_i$ will always choose this arm. We denote the resultant expected reward with $\mu^* = \mu_{i^*}$. However, since this information is not

usually known, any algorithm has a certain regret due to choosing the wrong arm with a lower expected reward. This leads to the definition of a performance metric known as the "regret", as follows:

$$\mathcal{R}(T) = \mathsf{E}\left[\sum_{t=1}^{T} (\mu^* - \mu_{i_t})\right], \tag{2.1}$$

where i_t is the arm chosen by the agent at time t, μ^* is the expected reward from the optimal arm, and the expectation is over the stochasticity of the obtained rewards that lead to stochasticity in the choices of the agent, along with any inherent stochasticity in the decision-making process of the agent. The regret is just a complement of the rewards, in the sense that maximizing the expected reward is equivalent to minimizing the regret.

However, one motivation for using regret instead of reward is that the above expression of the regret can be rearranged to give the following informative expression that is also more amenable to analysis.

$$\mathcal{R}(T) = \sum_{i \in [K]} \mathsf{E}\left[T_i(T)\right] \Delta_i, \tag{2.2}$$

where $T_i(t)$ is the number of times a sub-optimal arm i has been chosen by the agent till time t, and Δ_i is the expected sub-optimality due to choosing arm i, which is

$$\Delta_i = \mu^* - \mu_i.$$

This expression is more useful for analyzing the regret of the algorithm, since one can analyze how many times a sub-optimal arm can be chosen by the agent, along with the corresponding sub-optimality, combining them to obtain bounds on the overall regret.

Algorithms

In supervised learning, or even decision-making in general, the agent has to make decisions based on all the available data or evidence. However, the fundamental problem of sequential decision-making is that the decision made by the agent affects the performance metric as well as the data obtained itself. This leads to a trade-off between exploration, wherein an agent has to make a decision solely to obtain more or better information to make better decisions in the future, and exploitation, wherein the agent trusts its available data enough that it can actually make apparently correct decisions and reap the rewards.

Various algorithms have been studied in the literature to solve this exploration-exploitation dilemma in different ways. One straightforward method is to simply explore with a fixed small probability, and exploit the current best arm otherwise. This algorithm, called the ϵ-greedy

Algorithm 1 ϵ-greedy algorithm

Input: Number of arms K, Time horizon T, Exploration probability ϵ
// Sample each arm once
for each arm $i \in [K]$ **do**
  Choose arm i to receive reward $r_i \sim \mathcal{D}_i$
  Initialize mean $\hat{\mu}_{i,1} \leftarrow r_i$
end for
// ϵ-greedy exploration
for t in $\{K+1, \ldots, T\}$ **do**
  With probability ϵ
    Choose arm i randomly from $[K]$
  With probability $1-\epsilon$
    Choose the arm i with best $\mu_{i,t}$
  Update the empirical reward mean $\hat{\mu}_i$ for selected arm.
end for

algorithm, is given in Algorithm 1, wherein $\hat{\mu}_{i,t}$ is the empirical average reward obtained from arm i till time t. While this algorithm eventually finds the best arm because of choosing each arm many times due to the ϵ probability, it still suffers from linear regret because of choosing sub-optimal arms even after that.

$$\mathcal{R}_{\epsilon\text{-greedy}}(T) = \sum_{t=1}^{T} \left(\mathsf{E}\left[\mu^* - \mu_{i_t}\right]\right) = \sum_{t=1}^{T} \left(\mathsf{E}\Delta_{i_t}\right) \geq \sum_{t=1}^{T} \epsilon\left(1-\epsilon\right)\Delta_{\min} = \epsilon(1-\epsilon)\Delta_{\min} T,$$

where $\Delta_{\min} = \min\left\{\Delta_i (\neq 0) : i \in [K]\right\}$ is the minimum regret suffered at any time due to choosing a sub-optimal arm. A regret that is linear in the number of time steps is not ideal since it means that the algorithm is not optimally using its access to a greater number of samples as time passes.

From the above analysis, it is clear that at some point, it becomes more optimal to discard known sub-optimal arms from consideration and only choose the best known arm. A simple version of this idea is formalized by the Explore-Then-Commit (ETC) algorithm, given in Algorithm 2. While this algorithm does perform better in the latter stages of the time horizon due to pure exploitation, it is still sub-optimal in the initial stages because of a fixed number of pulls for each arm, even if an arm is known to be sub-optimal. Further, the number of initial arm pulls m is a hyperparameter that needs to be decided by the agent in advance, which may not be possible to do optimally.

The fundamental problem of the exploration-exploitation tradeoff is still not solved in the above two algorithms that consider explicit exploration and exploitation time steps. A frame-

Algorithm 2 `ETC` algorithm

Input: Number of arms K, Time horizon T, Number of initial arm pulls m
// `Exploration phase`
for each arm $i \in [K]$ **do**
  Play the arm i for m consecutive time steps
end for
// `Exploitation phase`
Determine best arm $i^* = \underset{i \in [K]}{\text{argmax}}\, \hat{\mu}_{i,mK}$
for t in $\{mK + 1, \ldots, T\}$ **do**
  Choose arm i^*
end for

work that solves this tradeoff implicitly is the class of Upper Confidence Bound (UCB) algorithms. The fundamental premise behind UCB algorithms is the idea of "Optimism in the face of Uncertainty". At each time step, a UCB agent chooses an arm that has the highest potential to be the optimal arm, quantified by the upper confidence of the mean of the reward distribution. The upper confidence term is the sum of the current estimate of the expected reward and an additional index term that captures the uncertainty due to only having access to a limited number of samples to make a decision. As the number of samples obtained from an arm increases, the uncertainty reduces, and the upper confidence of the mean moves asymptotically close to the actual mean of the reward distribution. Such an algorithm is given in Algorithm 3.

Algorithm 3 `UCB1` algorithm (Auer et al., 2002a)

Input: Number of arms K, Time horizon T
// `Sample each arm once`
for each arm $i \in [K]$ **do**
  Choose arm i to receive reward $r_i \sim \mathcal{D}_i$
  Initialize mean $\hat{\mu}_{i,1} \leftarrow r_i$
  Initialize the number of samples $n_{i,t} = 1$
end for
// `UCB Phase`
for t in $\{K + 1, \ldots, T\}$ **do**
  Determine for each arm i the index term $c_i = \sqrt{\frac{2 \log t}{n_{i,t}}}$
  // `Choose the arm with best UCB`
  Choose $i_t = \underset{i \in [K]}{\text{argmax}}\, \hat{\mu}_i + c_i$
  Update the empirical reward mean $\hat{\mu}_{i_t}$ for selected arm.
  Update number of samples for selected arm $n_{i_t} \leftarrow n_{i_t} + 1$
end for

This algorithm directly avoids the pitfalls of the previous algorithms. When an arm gives very little reward after many pulls, it will not have a high UCB and will thus be ignored from then on. However, an arm with low empirical reward with a smaller number of pulls will still be given a chance by virtue of having a high UCB, since the rewards are stochastic and the first few reward samples might have been low by chance. This avoids having to choose the hyperparameter m of the ETC algorithm, since the number of exploratory samples is variable and depends on the actual distribution of rewards for the respective arms.

2.2 Markov Decision Processes

A Markov Decision Process (MDP) is defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, p, \rho, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, p is the probability transition function, ρ is the initial state distribution and γ is the discount factor.

At each time t, the agent is at some known state s_t and has to choose an action $a_t \in \mathcal{A}$. Based on the chosen action, it receives a reward $r(s_t, a_t)$ and the state of the environment changes to some new state s_{t+1} based on the probability distribution $p(.|s_t, a_t)$ on $\mathcal{S}$. The objective of the agent is to maximize the cumulative discounted rewards obtained from the environment.

More precisely, suppose the agent follows a rule $\pi : \mathcal{S} \to \Delta\mathcal{A}$ to choose its action at each state s. That is, at each state s, the agent chooses an action a according to the probability distribution on $\mathcal{A}$ defined by $\pi(s)$, denoted by $\pi(.|s)$. Then the objective of the agent is to choose a policy π^* that is the solution of the following optimization problem:

$$\pi = \underset{\pi \in \Pi}{\operatorname{argmax}}\, \mathsf{E}_{s_0 \sim \rho, s_t, a_t \sim (p^t, \pi)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right],$$

where the states and actions at each time step are a stochastic function of all the previous actions taken by the agent and the environment's transition probabilities, and Π is the set of all allowed policies.

Value Functions and Bellman Equations

For solving such a problem, a general approach is to define functions called value functions that give the performance of a policy. The value function $V^\pi : \mathcal{S} \to \mathbb{R}$ for a policy π at state s is defined as

$$V^\pi(s) = \mathsf{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \,\middle|\, s_0 = s \right],$$

the expected discounted sum of rewards obtained by the agent when starting from state s by following policy π.

Similarly, the action-value function $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, also known as the Q-function, is defined as

$$Q^\pi(s,a) = r(s,a) + \gamma \mathsf{E}_{s' \sim p(.|s,a),\pi} \left[\sum_{t=1}^{\infty} \gamma^t r(s_t, a_t) \Big| s_1 = s' \right]$$

the expected discounted sum of rewards obtained by the agent when starting from state s by taking action a in the first step and following the policy π from then onward. It is clear from the definition that Q and V are related, with a difference of just the first term due to choosing the first action in different ways. Therefore, the Q function can be rewritten as

$$\begin{aligned} Q^\pi(s,a) &= r(s,a) + \gamma \mathsf{E}_{s' \sim p(.|s,a)} \left[V^\pi(s') \right] \\ &= r(s,a) + \gamma \mathsf{E}_{s' \sim p(.|s,a), a' \sim \pi(.|s')} \left[Q^\pi(s', a') \right]. \end{aligned}$$

The above equation, which defines the Q-value function recursively in terms of itself, is called the Bellman equation.

While the above value functions are defined for any arbitrary policy π, under some regularity conditions on $\mathcal{S}$ and $\mathcal{A}$, there exists an optimal policy π whose action-value function satisfies the so-called optimal Bellman equation

$$Q(s,a) = r(s,a) + \gamma \mathsf{E}_{s' \sim p(.|s,a)} \left[\max_{a' \in \mathcal{A}} Q(s', a') \right]. \tag{2.3}$$

The above equations are generally written in terms of the Bellman operator $\mathcal{T}^\pi$ and the optimal Bellman operator $\mathcal{T}^*$ on the space of real-valued functions on $\mathcal{S}$ or $\mathcal{S} \times \mathcal{A}$, defined as

$$\begin{aligned} \mathcal{T}^\pi[f](s) &= r(s, \pi(s)) + \gamma \mathsf{E}_{s' \sim P(.|s,a)} f(s'), \\ \mathcal{T}^*[f](s) &= \max_a \left\{ r(s,a) + \gamma \mathsf{E}_{s' \sim P(.|s,a)} f(s') \right\}, \end{aligned}$$

for $f : \mathcal{S} \to \mathbb{R}$. Since the domain of these operators contains real-valued functions defined on the state space, they can act on the value functions. Analogous Bellman and optimal Bellman operators can be defined on the space $\mathcal{S} \times \mathcal{A}$ that operates on Q functions.

2.2.1 Algorithms

Policy Iteration

Given that a value function evaluates and quantifies how good a policy is, one can use it to improve a given policy. This is essentially the idea of Algorithm 4, called the Policy Iteration algorithm.

Algorithm 4 Policy Iteration algorithm

Deterministic initial policy $\pi_0 : \mathcal{S} \to \mathcal{A}$
for t in $\{0, 1, 2, \dots, \}$ until convergence **do**
  // `Policy Evaluation`
  Evaluate π_t by determining V^{π_t}
  // `Policy Improvement`
  $\pi_{t+1}(s) \leftarrow \underset{a \in \mathcal{A}}{\text{argmax}}\ r(s, a) + \gamma \mathsf{E}_{s' \sim P(.|s,a)} V^{\pi_t}(s')$.
end for

If the policy evaluation step is exact, the policy improvement step is guaranteed to actually improve the policy. Further, when the state and action spaces are finite, this algorithm converges in a finite number of steps to the optimal policy. Else, provided some regularity conditions are satisfied, it is guaranteed to converge asymptotically to the optimal policy (Bertsekas, 2012).

This algorithm is not suitable as is for practical applications, since there is an optimization routine that has to run on the expectation of a function over an unknown probability distribution. But the general ideas can be extended to Q functions to obtain Q learning and actor-critic algorithms.

Q-Learning

A different iterative scheme can be obtained by considering the optimal Bellman operator on Q functions instead. The reinforcement learning problem boils down to finding the Q-function that satisfies the optimal Bellman equation, and the corresponding deterministic policy

$$\pi(s) = \underset{a}{\text{argmax}}\ Q(s, a)$$

induced by this Q-function is an optimal policy. For finding a solution to the optimal Bellman equation, an iterative scheme can be used that starts from an initial Q estimate and repeatedly updates it in the direction of minimizing the difference between the left-hand side and the right-hand side of the optimal Bellman equation. That is, starting with $Q^{(0)}$, the Q-function is updated at each iteration $k \geq 0$ as

$$Q^{(k+1)}(s, a) \leftarrow Q^{(k)}(s, a) + \alpha_k \Big(r + \gamma \max_{a' \in \mathcal{A}} Q^{(k)}(s', a') - Q^{(k)}(s, a) \Big), \tag{2.4}$$

where α_k is the learning rate during step k, and (s, a, r, s') is a transition tuple sampled from the environment.

Under certain assumptions on the learning rate schedule and the distribution of transition samples from the environment, the above iterative scheme can be shown to converge to the

optimal Q function, thus giving us a corresponding optimal policy (Watkins and Dayan, 1992).

Actor-Critic Methods

It can be seen that for each step in the above algorithms, the Q function update in (2.4) and the policy improvement step in Algorithm 4 contain a maximization term over all possible actions in the action space $\mathcal{A}$. While this can be computed easily for a finite action space, it is expensive to do an exact computation for arbitrary countable or continuous action spaces.

To deal with this issue, a corresponding function $\mu : \mathcal{S} \to \mathcal{A}$ is used that implicitly keeps track of an approximate solution for the optimization problem in policy iteration. This function corresponds to a deterministic policy under consideration in the current iteration and is referred to as an actor. The objective is the Q function of the current policy π_t, and is referred to as the critic.

Using such an abstraction, the iterative scheme can be broken down into two alternate updates:

$$\mu^{(k+1)}(s) = \operatorname*{argmax}_{a \in \mathcal{A}} Q(s, a), \tag{2.5}$$

$$Q^{(k+1)} = \operatorname*{argmin}_{Q} \mathsf{E}_{(s,a,s')} \left[\left(r(s, a) + \gamma Q\left(s', \pi(s')\right) - Q(s, a) \right)^2 \right], \tag{2.6}$$

where (2.5) is the Q-function variant of the policy improvement step in the policy iteration algorithm, and (2.6) achieves policy evaluation by minimizing the difference between two sides of the Bellman equation involving the Q function. Replacing the exact solutions to the optimization problems in the above equations with approximate updates consisting of a few gradient steps gives rise to actor-critic algorithms. For complex environments, the actor and critic functions can be represented by neural network function approximators, and the policy can be stochastic as well. There are a multitude of such actor-critic methods in the literature (Lillicrap et al., 2016; Haarnoja et al., 2018; Fujimoto et al., 2018) that solve these problems in different ways.

2.3 Risk

In machine learning and statistical learning literature, "risk" refers to expected loss. Consider a supervised learning task wherein a dataset

$$\mathcal{D} = \{(x_i, y_i) \mid x_i \in \mathcal{X}, y_i \in \mathcal{Y}, i \in [N]\}$$

of samples is given to an algorithm, to be modeled by an unknown function $f : \mathcal{X} \to \mathcal{Y}$. A loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is used as a performance metric to quantify the quality of any prediction

function. Given the true value y and a prediction $\hat{y}$ based on some model, $\ell(y, \hat{y})$ signifies how far the prediction was from the actual ground truth.

But this is just for one specific sample. For defining the quality of prediction for all possible values of $\mathcal{X}$, one assumes that there exists some probability distribution $P(X, Y)$ on $\mathcal{X} \times \mathcal{Y}$. In this case, the risk, defined as the expected loss (Castro, 2007), is given by

$$R(f) := \mathsf{E}_{(x,y)\sim P(X,Y)} \left[\, \ell(y, f(x)) \, \right].$$

Many techniques, along with theoretical guarantees, exist to minimize this risk based on an approximation using available samples, along with an intelligent choice of the model class $\mathcal{F}$ from which the functions f should be taken, as well as possible regularization techniques, etc. This forms the core of statistical learning theory (Hastie et al., 2009).

In this thesis, we consider a related, but different concept of risk that is more general than the one defined above, which is used in various fields such as finance. Essentially, it involves going beyond the expectation of loss metrics to consider other distributional properties as well for quantifying the performance of various algorithms.

Let D be some domain and $\Delta(D)$ be a class of probability distributions on D. A risk measure is defined as a function $U : \Delta(D) \to \mathbb{R}$ that assigns a real number to each probability distribution on D.

A simple risk measure is the expectation of the distribution itself, i.e,

$$U(P) = \mathsf{E}_{X\sim P} X, \quad \text{for all } P \in \Delta(D).$$

This is the common risk measure used in machine learning. A slightly more general risk measure is the Mean-Variance model (Sani et al., 2012), which is defined for risk tolerance ρ as

$$U(P) = \mathsf{Variance}(P) - \rho \, \mathsf{Mean}(P),$$

a linear combination of the variance and mean of the distribution defining a risk-reward trade-off.

While the variance of a distribution does give a meaningful sense of risk because it quantifies the spread of a distribution, it is inherently symmetric. The spread of a loss above the mean is treated the same way as the spread below the mean. However, intuition suggests that higher losses above the mean should indicate higher risk, while lower losses below the mean should not contribute to a higher risk value.

This is incorporated by the risk measure VaR_α, the Value-at-Risk at risk level α, defined for

a random variable X as

$$\mathsf{Var}_\alpha(X) = \inf_{x \in \mathsf{Supp}(X)} \{\mathsf{P}\left(X \leq x\right) \geq \alpha\},$$

where $\mathsf{Supp}(X)$ is the support of the random variable, and $\mathsf{P}(\cdot)$ denotes the probability of a given event. Here, the random variable X denotes the reward or the negative of the loss obtained by an algorithm. Intuitively, the Value-at-Risk at risk level α is the α-quantile of the reward distribution, or the $(1-\alpha)$-quantile of the loss distribution. This takes care of skewed distributions where the loss could be much higher than the average loss.

Still, VaR_α only considers a quantile of the distribution, and ignores the properties of the distribution above this quantile. Even if the quantile value is small, the shape of the distribution beyond the quantile could lead to a huge loss with some probability less than α.

These concerns are addressed by the risk measure CVaR_α, the Conditional Value-at-Risk, also called the Expected Shortfall (ES) (Acerbi and Tasche, 2002), defined for a random variable X as follows.

$$\mathsf{CVaR}_\alpha(X) = \frac{1}{\alpha} \int_0^\alpha \mathsf{VaR}_\alpha(X)\, \mathrm{d}p,$$

the average of VaR_α values beyond the α-quantile. Essentially, for continuous random variables, it is the expected value of X conditioned on the value of X being at most its α-quantile, namely $\mathsf{VaR}_\alpha(X)$. For discrete distributions, a term needs to be subtracted from this expectation to account for the extra probability mass at Var_α that goes beyond the quantile α.

We consider this concept of risk in the context of multi-armed bandits. The analogue to loss here is the regret of the bandit algorithm. The regret of a bandit algorithm is defined as the expected difference between the rewards obtained by an agent with full knowledge of the underlying distributions and the actual algorithm.

This notion can be extended to a risk measure in multiple ways. Consider a bandit with K arms and time horizon T, with the reward distribution of each arm i being $\mathcal{D}_i$. An agent interacting with the bandit obtains rewards $r_1, \ldots, r_t$ that are random and depend on its policy. The objective is to optimize the risk $U(\cdot)$ of the obtained rewards.

Consider policies of the form $\pi = (\pi_1, \pi_2, \pi_3, \ldots,)$, where π_t is a policy that chooses an action at time t based on all the rewards obtained till time $t-1$. Let $\pi^* = (\pi_1^*, \pi_2^*, \ldots)$ be the oracle policy that knows the underlying distribution over all arm rewards and is optimal with respect to U. The regret of the algorithm corresponding to policy π can be defined as

$$\mathcal{R}_T(\pi) = \mathsf{E}\left[U\left(\hat{F}_T^{\pi^*}\right) - U\left(\hat{F}_T^{\pi}\right)\right],$$

where $\hat{F}_t^\pi$ is the empirical distribution of rewards obtained using policy π till time t.

This metric considers the CVaR_α of the empirical distribution of all the rewards obtained by a policy, which is a cumulative metric that cannot be decomposed into the sum of quantities at each time step or corresponding to each arm, analogous to how the standard regret defined in (2.1) could be rewritten as (2.2), which accumulates individual terms for each arm based on the number of times that arm was chosen. This makes the analysis of this measure of regret very challenging.

A slightly more reasonable regret measure is the pseudo-regret defined as follows.

$$\mathcal{R}_T(\pi) = \max_{i \in [K]} \mathsf{CVaR}_\alpha(\mathcal{D}_i) - \mathsf{E}\left[\mathsf{CVaR}_\alpha\left(\hat{F}_T^\pi\right)\right].$$

Here, the oracle term is replaced by the CVaR of the arm with the best CVaR, which is simpler than considering a more powerful non-stationarity oracle. This is analogous to the regret considered in the case of adversarial bandits (Auer et al., 2002b), where the oracle is given the option to choose just one arm for the entire time horizon. However, it still considers a cumulative CVaR for the agent, which cannot be broken down across time steps.

Another variant is the proxy-regret (Khajonchotpanya et al., 2021), which considers the difference between the CVaRs of the optimal arm and the chosen arm, in line with the standard notion of expected regret in the risk-free case.

$$\mathcal{R}_T(\pi) = T \cdot \max_{i \in [K]} \mathsf{CVaR}_\alpha(\mathcal{D}_i) - \mathsf{E}\left[\sum_{t=1}^{T} \mathsf{CVaR}_\alpha\left(\mathcal{D}_{i^t}\right)\right],$$

where i^t is the arm chosen by the agent at time t.

2.4 Temporal Processes

A temporal point process $(X_t)_{t \in I}$ is a set of random variables X_t indexed by time t in an interval $I \subseteq \mathbb{R}$, which is generally taken to be $[0, \infty)$. A counting process is a temporal process that, at each time t, describes the number of some "events" that have occurred till that time.

A continuous time counting process $(S_t)_{t \geq 0}$ is defined by its conditional intensity function $\lambda(t)$, defined as

$$\lambda(t) = \lim_{h \downarrow 0} \frac{\mathsf{E}\left[S_{t+h} - S_t | \mathcal{H}_t\right]}{h},$$

the expected rate of new events conditioned on the history $\mathcal{H}_t$ of events till time t.

A Hawkes process is a self-exciting temporal point process, which means that the probability of an event occurring in some time period depends on the occurrence of events in the past.

Formally, the conditional intensity function of a Hawkes process is given by

$$\lambda(t) = \lambda + \sum_{t'<t} \mu(t - t'),$$

where λ is the base intensity, t' are the times at which previous events have occurred, and μ is a function that describes the effect of previous events on the current intensity. A monotonically decreasing μ means that the occurrence of an event increases the probability of occurrence of more events in the near future, and this effect slowly fades according to the rate of decay of μ.

Since most reinforcement learning algorithms consider discrete-time MDPs, we too consider a discrete-time process as the exogenous process that perturbs the MDP. A discrete-time analogue of the Hawkes process was proposed by Seol (2015) as follows.

Let $(E_t)_{t\in\mathbb{N}}$ be a sequence of random variables taking values in $\{0, 1\}$, with E_t sampled from $\mathsf{Bernoulli}\,(p_t)$, where the intensity p_t at time t depends on the realizations of E_t at the previous time steps as

$$\begin{aligned} p_1 &= \alpha_0, \\ p_t &= \alpha_0 + \sum_{t'=1}^{t-1} \alpha_{t-t'} E_{t'} \quad \text{for } t > 1, \end{aligned}$$

where (α_t) converges sufficiently quickly to zero. The sum $S_t = \sum_{t'=1}^{t} E_t$ is a self-exciting counting process that is the discrete-time counterpart of the continuous-time Hawkes process. Here, α_0 is analogous to the base intensity λ in the continuous-time case, while the sequence $(\alpha_t)_{t\geq 1}$ is analogous to the kernel function μ that determines the effect of previous events based on the time passed. While the continuous-time kernel is specified as a function of time, the discrete-time kernel can be specified as either a general sequence or in some parametric form.

Chapter 3

Risk-Averse Combinatorial Semi-Bandits

In a standard multi-armed bandit problem, the agent has to choose one arm from a fixed set of arms at each time step. However, many applications (Chen et al., 2013; Gai et al., 2010; Gai et al., 2012) require the agent to select a combination of arms at each time step, sometimes satisfying certain constraints. Such problems have been studied as the multi-armed bandit with multiple plays (Anantharam et al., 1987) and the combinatorial bandit (Gai et al., 2012).

In a combinatorial bandit, a set of constraints is given to the agent, and at each time step, the agent has to choose a combination of arms, called a super-arm, that satisfies the given constraints. A reward is given to the agent as a stochastic function of the super-arm chosen. In the semi-bandit setting, individual rewards from the arms belonging to the selected super arm are also known to the agent. In this chapter, we study risk-averse combinatorial semi-bandits and present some algorithms to tackle this problem and derive results on their regret. These results are also reported in Ayyagari and Dukkipati (2023).

3.1 Problem Formulation

Consider a bandit with K arms and time horizon T. Each allowed super arm is an element $a \in \mathcal{A} \subseteq 2^{[K]}$. In other words, $\mathcal{A}$ defines the constraints on the possible combinations of arms that can be chosen by the agent at each time step. Let $L = \max_{a \in \mathcal{A}} |a|$ be the maximum possible size of an allowed super arm.

Each arm $i \in [K]$ has an associated probability distribution $\mathcal{D}_i$ for its reward. When a super arm $a \in \mathcal{A}$ is selected, a reward is sampled from each arm $i \in a$ and revealed to the agent. The total reward received by the agent is the sum of these sampled rewards.

Note on notation: We use the symbols $\oplus$ and $\bigoplus$ to denote the binary and n-ary sum of probability distributions, and so denote the distribution of the reward of super arm a as $\mathcal{D}_a$ or

$\mathcal{D}_{\bigoplus_{i\in a}\mathcal{D}_i}$. Depending on the context, for any distribution $\mathcal{D}$ with some subscript/superscript, we denote its corresponding cumulative distribution function and probability density/mass function as F and f, respectively, with the same subscript/superscript.

For any random variable $X \sim \mathcal{D}$ with probability distribution $\mathcal{D}$ and cumulative distribution function F_X, the Conditional Value at Risk, for risk level $\alpha \in (0, 1)$, is defined as

$$\mathsf{CVaR}_\alpha(X) = \frac{1}{\alpha}\int_0^\alpha \mathsf{VaR}_p(X)\mathrm{d}p,$$

where VaR_p is the Value-at-Risk at risk level p. Value-at-Risk is defined as

$$\mathsf{VaR}_\alpha(X) = x_\alpha = \inf\{x \in \mathsf{Supp}(\mathcal{D}) : F_X(x) \geq \alpha\}.$$

Intuitively, for continuous probability distributions, VaR_α is just the α quantile of distribution, and CVaR_α is the expected value of the random variable conditioned on its value being at most VaR_α. It is the same for discrete distributions, except for an additional term that subtracts any extra probability mass at the VaR that goes beyond the cutoff α.

The goal of the agent is to choose the super arm with the best Conditional Value-at-Risk at every time step. Sub-optimal choices induce regret, and the goal is to minimize the cumulative CVaR regret, defined by

$$\mathcal{R}_{\mathsf{CVaR}_\alpha}(T) = \mathsf{E}\left[\sum_{t=1}^{T}\left(\mathsf{CVaR}_\alpha(a^*) - \mathsf{CVaR}_\alpha(a_t)\right)\right], \tag{3.1}$$

where $\mathsf{CVaR}_\alpha(a)$ is the CVaR of the reward distribution $\mathcal{D}_a$ of super arm a, a_t is the super arm chosen by the agent at time t, and a^* is the super arm with the highest CVaR. The above regret can also be written as

$$\mathcal{R}_{\mathsf{CVaR}_\alpha}(T) = \sum_{a\in\mathcal{A}} \mathsf{E}\left[T_a(T)\right]\Delta_a,$$

where $T_a(T)$ is the number of times super arm a has been chosen till time T, and $\Delta_a = \mathsf{CVaR}_\alpha(a^*) - \mathsf{CVaR}_\alpha(a)$ is the CVaR gap of super arm a. Further, let $\Delta_{\min}$ and $\Delta_{\max}$ denote the minimum and maximum nonzero Δ_a among all super arms $a \in \mathcal{A}$.

3.2 Algorithms

The naive way to solve a combinatorial bandit problem is to treat each super arm as an arm, thereby reducing the problem to a standard multi-armed bandit problem. However, this approach ignores the dependence between super arms and results in regret that depends linearly

on the number of super arms, which itself may depend exponentially on the number of arms. It has been shown that an algorithm and regret analysis that takes into account this dependence can yield a regret upper bound that is only polynomial in the number of arms (Gai et al., 2012).

The crux of such an algorithm is the way in which samples from individual arms are used to construct estimates of the relevant properties of the super arm reward distributions. This depends on the nature of the arm reward distributions. In this work, we propose risk-aware algorithms for two cases: (i) normally distributed and (ii) bounded arm rewards.

3.2.1 Gaussian Rewards

Assume that each arm i gives Gaussian rewards with mean μ_i and variance σ_i^2, and the rewards of all arms are independent. This implies a super arm $a \in \mathcal{A}$ has a Gaussian reward distribution with mean $\mu_a = \sum_{i \in a} \mu_i$ and variance $\sigma_a^2 = \sum_{i \in a} \sigma_i^2$. The CVaR of this super arm is then given by

$$\mathsf{CVaR}_\alpha(\mathcal{D}_a) = \mu_a - \frac{\sigma_a}{\alpha} \varphi\left(\Phi^{-1}(\alpha)\right),$$

where φ and Φ are the probability density function and cumulative distribution function of the standard normal distribution. Hence, to estimate the CVaR of a super arm, it is sufficient to estimate the mean and standard deviation of the super arm.

Since both the mean and the variance are unknown, for obtaining suitable finite-sample guarantees on our estimates, we assume that the agent knows an upper and lower bound on the variance of all arms, i.e., there exist known constants $M, N > 0$ such that $N^2 < \sigma_i^2 < M^2$ for every arm $i \in [K]$.

This allows us to construct upper confidence bounds on the mean of each arm and lower confidence bounds on the variance of each arm, and in turn, use these to create upper confidence bounds on the CVaR of all super arms. During each time step, we select the super arm with the highest CVaR upper confidence bound, as detailed in Algorithm 5.

Theorem 1. *The expected cumulative regret of Algorithm 5 over time horizon T satisfies the following upper bound:*

$$\begin{aligned}\mathcal{R}_{\mathsf{CVaR}_\alpha}(T) \lesssim & \left(2 + \frac{2}{3}\pi^2\right) K \Delta_{\max} \\ & + \mathcal{O}\left(\frac{M^2 \sqrt{L} K \log T \Delta_{\max}}{\Delta_{\min}}\right) \cdot \max\left\{\frac{96 L \sqrt{L}}{\Delta_{\min}}, \mathcal{O}\left(\frac{\varphi(\Phi^{-1}(\alpha))}{\alpha N}\right)\right\},\end{aligned}$$

where K is the number of arms of the bandit, L is the maximum number of arms in any super

Algorithm 5 Algorithm CVaR_α-CUCB-G

Require: M, N

Initially, pick each super arm so that all the arms have at least two rewards obtained

for each time t till T **do**

for each arm $i \in [K]$ **do**

Estimate sample mean $\hat{\mu}_i$ & sample variance $\hat{s}_i^2$

// `Construct corresponding confidence bounds`

$T_{i,t-1}$: Number of samples from arm i

$\tilde{\mu}_i \leftarrow \hat{\mu}_i + M\sqrt{\frac{6\log t}{T_{i,t-1}}}$

$D_{i,t-1} \leftarrow 2M^2\left(\frac{3\log t}{T_{i,t-1}-1} + \sqrt{\frac{3\log t}{T_{i,t-1}-1}}\right)$

$\tilde{s}_i^2 \leftarrow \max\left\{\left(\hat{s}^2_{i,T_{i,t-1}} - D_{i,t-1}\right), N^2\right\}$

end for

// `Construct UCB for` CVaR `of each super arm` a

$\mathrm{UCB}_a \leftarrow \sum_{i\in a}\tilde{\mu}_i - \frac{\sqrt{\sum_{i\in a}\tilde{s}_i^2}}{\alpha}\varphi(\Phi^{-1}(\alpha))$

Pick the super arm with the best UCB.

end for

arm, M and N are the (known) upper and lower bounds of the standard deviation of the arms. $\Delta_{\max}$ and $\Delta_{\min}$ are the maximum and minimum difference between the CVaR *of the optimal super arm and the* CVaR *of any suboptimal super arm.*

The first term in the regret upper bound is due to $2K$ initialization rounds and the probability that the confidence intervals are not accurate in the later rounds. The second term is due to the number of rounds required for the confidence intervals of the super arms to be sufficiently precise enough to prevent the selection of suboptimal super arms. Since the confidence interval for the CVaR is constructed by estimating the mean and variance of individual arms, this term depends on the maximum of two expressions as given, which signify the number of rounds required to obtain sufficient precision for the estimation of the mean and variance, respectively.

3.2.2 Bounded Rewards

In this setting, we assume that the rewards from each of the arms are non-negative and bounded above by a known upper bound. Without any loss in generality, we can assume that the rewards of each arm fall in the interval $[0, 1]$.

In the previous subsection, the Gaussian assumption for the rewards of the arms significantly simplifies the estimation of CVaR of the super arms since the CVaR of each super arm reduces to a simple function of the mean and standard deviation of the corresponding individual arms. But for general nonparametric distributions, obtaining confidence intervals for the CVaR of a

Algorithm 6 Algorithm CVaR_α-SDCB

Initially, pick each super arm so that all the arms have at least one reward obtained
for each time t till T **do**
  for each arm $i \in [K]$ **do**
    $\hat{F}_{i,t-1}$: Empirical reward distribution
    $T_{i,t-1}$: Number of samples from arm i
    $C_{i,t-1} \leftarrow \sqrt{\frac{3\log(t)}{2T_{i,t-1}}}$
    $\tilde{F}_i(x) \leftarrow \begin{cases} (\hat{F}_i(x) - C_{i,t-1})^+, & x < 1 \\ 1, & \text{otherwise} \end{cases}$
  end for
  Calculate empirical CDF for each super arm a as:
    $\tilde{F}_a \leftarrow F_{\bigoplus_{i\in a}\tilde{F}_i}$
  Calculate $\mathsf{CVaR}_\alpha(\tilde{F}_a)$ for each super arm a
  Pick super arm with best CVaR
end for

super arm is much less straightforward since it cannot be calculated as a simple function of the CVaR's of the constituent arms.

For simply estimating the CVaR of a super arm, it is sufficient to calculate the empirical CVaR from the obtained rewards. However, for minimizing the CVaR regret, we need to construct upper confidence bounds on the super arm CVaR. This can be done by constructing a probability distribution that is close but stochastically dominates the super arm reward distribution. This stochastically dominant probability distribution, in turn, can be constructed by constructing stochastically dominant distributions for each of the individual arms and calculating the convolution of those distributions.

Thus, if $\hat{F}_i$ is the empirical cumulative distribution function formed by all the samples from arm i, we construct a corresponding stochastically dominant distribution $\tilde{F}_i$ by subtracting a constant throughout the domain of F below 1, the known upper bound for the actual distribution being estimated. This can be used to construct a stochastically dominant distribution for each super arm a as $\tilde{F}_a = \bigoplus_{i\in a}\tilde{F}_i$. This procedure is listed in Algorithm 6.

Theorem 2. *The regret for Algorithm 6 satisfies*

$$\mathcal{R}_{\mathsf{CVaR}_\alpha}(T) \leq C\frac{L^3}{\alpha^4}\log T \sum_{i\in a_B}\frac{1}{\Delta_{i,\min}} + \left(1+\frac{\pi^2}{3}\right)K\Delta_{\max},$$

where a_B is the set of arms contained in at least one suboptimal super arm, and $\Delta_{i,\min} = \min\{\Delta_a(\neq 0) : i \in a\}$.

Computational Complexity

The above algorithm requires computing a stochastically dominant probability distribution for each super arm in $\mathcal{A}$, which involves computing the probability distribution of a sum of at most L discrete probability distributions.

For distributions f_1, f_2, determining $f_1 \oplus f_2$ requires performing a convolution of f_1 and f_2 and involves at most $|\text{Supp}(f_1)||\text{Supp}(f_2)|$ computations, where Supp(.) indicates the support. Further, $|\text{Supp}(f_1 \oplus f_2)| \leq |\text{Supp}(f_1)||\text{Supp}(f_2)|$ with equality occurring in the worst case. Following this argument, it is clear that the number of computations required for just the final step of calculating the stochastically dominant distribution for each a is

$$\prod_{i \in a} |\text{Supp}(\tilde{F}_{i,t-1})| \leq \left(\max_{i \in a} |\text{Supp}(\tilde{F}_{i,t-1})| \right)^L.$$

This quantity is exponential in $L(\leq K)$ and might cause the algorithm to become computationally expensive for large L, or for a large time horizon that may cause $\max_{i \in a} |\text{Supp}(\tilde{F}_{i,t-1})|$ to become large for continuous probability distributions. To mitigate this, we propose a discretized algorithm.

3.2.3 Discretized Algorithm

The above problem of exponentially increasing support can be solved by discretizing the distributions and allowing the random variables involved to take only certain values. More specifically, we choose some small real number $\epsilon > 0$, and at each time t, we "round up" each distribution $\tilde{F}_{i,t-1}$ to a new distribution $F'_{i,t-1}$ by moving the probability mass at each point $x \in \text{Supp}(\tilde{F}_{i,t-1})$ to the smallest point $x' \geq x$ that is an integral multiple of ϵ, i.e, $x' = \lceil \frac{x}{\epsilon} \rceil \epsilon$. When probability mass from multiple points is moved to the same multiple of ϵ, the individual mass values are added up to obtain the total probability mass at the final point. The following lemma quantifies the resultant effect on the super arm CVaR.

Lemma 1. *Let F'_i be the probability distribution obtained by discretizing the distribution $\tilde{F}_i$ of arm i. For each super arm a, let $\tilde{F}_a = \bigoplus_{i \in a} \tilde{F}_a$ and $F'_a = \bigoplus_{i \in a} F'_a$. Then,*

$$\mathsf{CVaR}_\alpha(\tilde{F}_a) \leq \mathsf{CVaR}_\alpha(F'_a) \leq \mathsf{CVaR}_\alpha(\tilde{F}_a) + \frac{\epsilon(L+1)}{\alpha}.$$

Based on the above result, for our problem of minimizing the CVaR regret, we choose parameter $\epsilon = \frac{\alpha}{(L+1)T}$. The regret of the resultant algorithm D-CVaR$_\alpha$-SDCB is bounded as

Algorithm 7 Algorithm D-CVaR_α-SDCB

Require: T

Initially, pick each super arm so that all the arms have at least one reward obtained

for each time t till T **do**

for each arm $i \in [K]$ **do**

$\hat{F}_{i,t-1}$: Empirical reward distribution

$T_{i,t-1}$: Number of samples from arm i

$C_{i,t-1} \leftarrow \sqrt{\frac{3\log(t)}{2T_{i,t-1}}}$

$\tilde{F}_i(x) \leftarrow \begin{cases} (\hat{F}_i(x) - C_{i,t-1})^+, & x < 1 \\ 1, & \text{otherwise} \end{cases}$

Discretize $\tilde{F}_i$ to F'_i

end for

Calculate empirical CDF for each super arm a as:

$F'_a \leftarrow F_{\bigoplus_{i \in a} F'_i}$

Calculate $\mathsf{CVaR}_\alpha(F'_a)$ for each super arm a

Pick super arm with best CVaR

end for

follows.

Theorem 3. *The algorithm D-CVAR$_\alpha$-SDCB has* CVaR *regret that satisfies*

$$\mathcal{R}_{\mathsf{CVaR}_\alpha}(T) \leq 2 + \left(1 + \frac{\pi^2}{3}\right) K\Delta_{\max} + C\frac{L^3}{\alpha^4} \log T \sum_{i \in a_B} \frac{1}{\Delta_{i,\min}}.$$

Therefore, using a suitable precision ϵ results in no additional regret due to the discretization. However, this choice of ϵ requires the knowledge of the time horizon T, unlike the previous two algorithms. The proofs of the theorems are given in Section 3.A.

3.2.4 Remarks

Analysis of the regret of CVaR_α-CUCB-G requires two-sided confidence bounds on both the mean and standard deviation of the arm reward distributions, which is unusual for standard multi-armed bandit problems. Although bandits with unknown means and variances have been studied, the optimizing objective still consists of only the expectation of the rewards, and hence, prior techniques cannot be applied in our case. Auer et al. (2002a) used a concentration inequality for χ^2 random variables that is a conjecture that they verified numerically. For constructing confidence intervals as part of our analysis, we use the concentration inequalities that are a corollary of Lemma 1 in Laurent and Massart (2000). However, the size of such a

confidence interval for the variance of the Gaussian random variable itself depends on the true unknown variance. To deal with this, we assume knowledge of upper and lower bounds on the variance of each arm's reward distribution.

3.3 Numerical Results

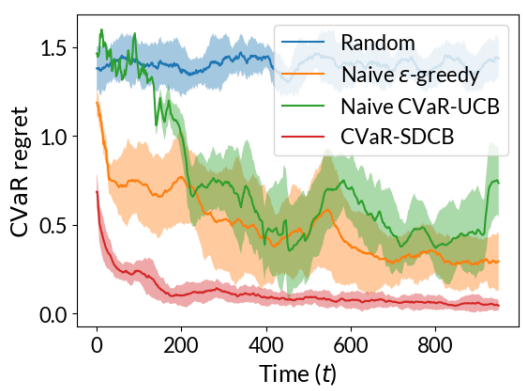
We conducted numerical experiments to compare the performance of algorithms on simulated combinatorial bandit problems with bounded and Gaussian rewards with $\alpha = 0.05$. In the first case, we consider a bandit with $K = 20$ arms for $T = 1000$ time steps. Each arm's reward is Bernoulli, with the mean randomly sampled uniformly from $[0, 1]$. For $\mathcal{A}$, we randomly sample 30 allowed super arms of size $L = 10$. We compare our algorithm CVaR_α-SDCB with (i) a random algorithm that uniformly chooses $a \in \mathcal{A}$, (ii) a naive ϵ-greedy algorithm that chooses a random super arm with probability $\epsilon = 0.1$ and the current best super arm with probability $1 - \epsilon$, and (iii) a naive extension of CVaR-UCB (Tamkin et al., 2019) that considers each of the allowed super arms as an arm of the bandit. Figures 3.1a, 3.1c and 3.1e show the CVaR regret, the cumulative CVaR regret and the cumulative CVaR respectively as a function of the time step. It is clear that our algorithm significantly outperforms the others, showing that leveraging the combinatorial nature of the problem does indeed result in better results in practice.

For Gaussian rewards, we use the same values for T, K, L, but the mean for each arm has been randomly sampled from $[-10, 10]$ and the standard deviation from $[1, 5]$. 150 random super arms of size L form $\mathcal{A}$. Our algorithm is compared with the same random and ϵ-greedy baselines as before. However, other optimism-based algorithms like CVaR-UCB, etc., assume that the rewards are bounded and hence cannot be used in this setting as baselines. Therefore, the third baseline shown is a naive extension of the pessimistic MARAB algorithm (Galichet et al., 2013), which uses a lower bound of the CVaR of each arm for risk-averse exploration.

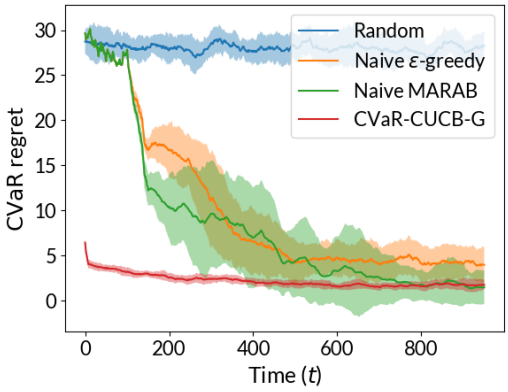
For this setting, it can be seen from Figures 3.1b and 3.1d that even though the individual regret terms at the end of the horizon are almost the same for MARAB, the cumulative regret, which is the ultimate objective being optimized for, is still superior for CVaR-CUCB-G. The same is the case for the third metric in Figure 3.1f, which is the empirical CVaR of all rewards obtained by the algorithm till each time step. This shows that even a pessimistic algorithm could not avoid taking risky actions in the beginning stages of the episodes, thereby performing poorly compared to our algorithm.

3.4 Related Work

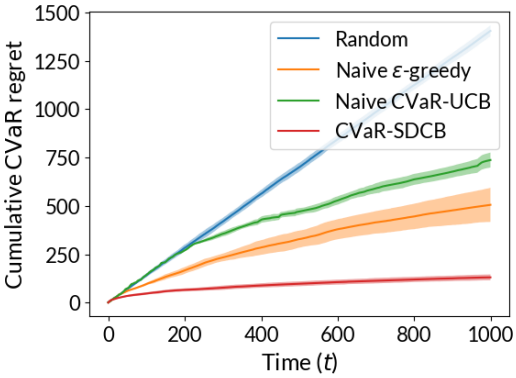
The multi-armed bandit problem with multiple plays was first studied by Anantharam et al. (1987), who provided an algorithm and a lower bound on the regret for parameterized rewards.



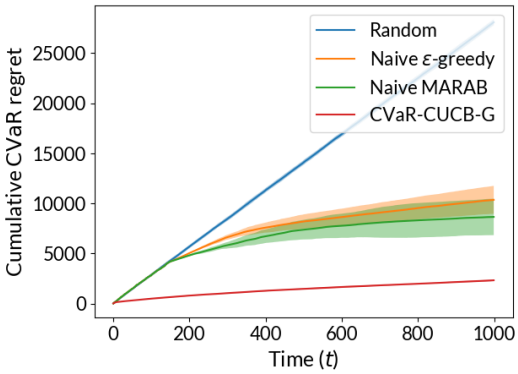
(a) CVaR regret



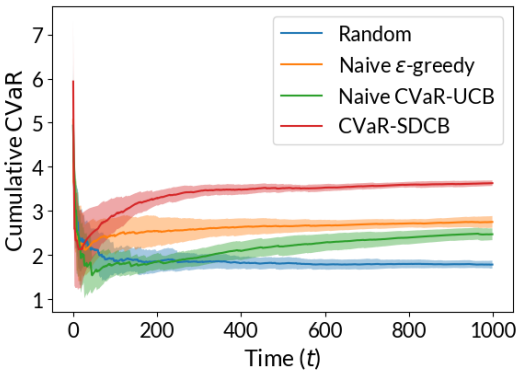
(b) CVaR regret



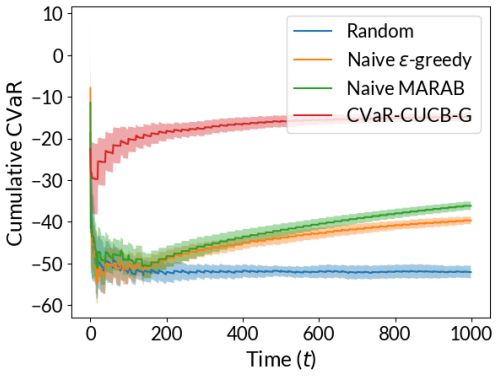
(c) Cumulative CVaR regret



(d) Cumulative CVaR regret



(e) Cumulative CVaR



(f) Cumulative CVaR

Figure 3.1: Comparison of our algorithms (in red) with baselines on two simulated combinatorial multi-armed bandit problems. The plots show the average of the respective metrics over 15 runs, with the shading representing ± as the standard deviation. Figures 3.1a, 3.1c and 3.1e correspond to a bounded reward setting, and Figures 3.1b, 3.1d and 3.1f correspond to Gaussian rewards with unknown means and variances. Figures 3.1a and 3.1b are smoothed by taking a moving average with a window of size 50.

Gai et al. (2010) formulated the combinatorial bandit problem for the problem of optimally allocating channels to users in a cognitive radio network. This has been extended by Chen et al. (2013) to more general non-linear reward functions that are monotone and smooth. However, this previous approach only applies when the expected reward from a super arm is fully characterized by the means of the individual arm reward distributions. Because of this assumption, the super arm selection can be done just by constructing and using the upper confidence bounds (UCB) for each of the arm rewards means.

The above assumption is removed by Chen et al. (2016), who proposed SDCB (Stochastically Dominant Confidence Bound), which can also deal with bounded monotone reward functions that depend on the entire probability distributions of the arm rewards and not just the means. SDCB constructs a new probability distribution for each arm that acts as a stochastically dominant confidence bound for the unknown arm reward distribution and uses these to determine the optimal super arm. We adapt this technique to tackle the case of general bounded rewards.

Kveton et al. (2015) provided tight regret bounds for the case of bounded linear rewards, which are further improved by Combes et al. (2015) for the special case of Bernoulli rewards. For general bounded rewards, Merlis and Mannor (2019) proposed an algorithm that constructs confidence bounds for the arms that also use the empirical variance of the observed samples. Their regret bound is independent of the size of the super arms and instead depends on the smoothness parameters of the non-linear reward function.

The combinatorial bandit problem has also been tackled using a Thompson sampling approach by Wang and Chen (2020), whose algorithm achieves the same theoretical regret as the UCB-based algorithm CUCB (Chen et al., 2013), and matches the regret lower bound of Kveton et al. (2015) for linear reward functions.

Risk awareness has first been studied for multi-armed bandits under the mean-variance criterion (Sani et al., 2012) and the MIN and CVaR criteria (Galichet et al., 2013) for bounded rewards. An algorithm with provable guarantees on the CVaR regret was proposed for bounded rewards by Galichet (2015).

Bhat and L.A. (2019) provide concentration results for CVaR estimation, and thereby a UCB-like algorithm for optimizing CVaR regret, for the case of sub-Gaussian arm rewards. A distribution oblivious algorithm that works with rewards that are unbounded or have heavy tails has been proposed and analyzed by Kagrecha et al. (2019).

An algorithm, U-UCB, that can deal with general risk metrics, including CVaR, has been proposed by Cassel et al. (2018). They have also derived performance guarantees for their algorithm (Cassel et al., 2023) with respect to a stronger form of regret than in other works that deal with CVaR. Our work is the first to develop and study such risk-aware algorithms in

a combinatorial setting.

3.5 Outlook

Combinatorial bandit problems are encountered in a wide variety of real-world applications. They also arise naturally from online resource allocation problems with threshold-based rewards (Verma et al., 2019). In this work, we consider the problem of designing risk-averse algorithms for tackling combinatorial bandits with semi-bandit feedback. More specifically, we proposed algorithms for optimizing the CVaR of the rewards obtained by an agent from a combinatorial semi-bandit, for the cases of Gaussian and bounded arm rewards. We provided upper bounds for the CVaR regret of these two algorithms, and for the bounded reward case, we also discussed the implications of discretization on the regret and computational complexity of the corresponding algorithm.

A discretization approach similar to ours was used by Chen et al. (2016) to decrease the worst-case space and time complexity of maintaining $\hat{F}_i$ from $\Theta(T)$ and $\Theta(T^2)$ to $\Theta(\sqrt{T})$ and $\Theta(T^{3/2})$ respectively. However, in our work, we use discretization mainly for tackling the computation of the probability distributions corresponding to each of the super arms. This computation has to be done for each of the allowed super arms separately. The number of allowed super arms $|\mathcal{A}|$ is part of the problem specification and can be exponential in the number of arms in the worst case, rendering the algorithms very computationally expensive. However, this is unavoidable since determining the optimal super arm is an NP-hard problem (Maehara, 2015) and does not admit a polynomial-time multiplicative approximation algorithm, even with additional assumptions on $\mathcal{A}$, such as being a matroid.

While some works have proposed polynomial-time online risk-averse algorithms in combinatorial settings (Ohsaka and Yoshida, 2017; Soma and Yoshida, 2021), the problem they solve is a relaxation of our problem in which the agent is allowed to choose a portfolio of super arms instead of a single super arm. In that setting, the reward from a portfolio is a convex combination of rewards from the super arms in that portfolio. Note that in our setting, while the agent can choose to pick different super arms with different probabilities, the effective reward distribution would be a mixture distribution and not equivalent to the portfolio reward distribution. Further work is needed to bridge the gap between these two different approaches to sequential decision-making for combinatorial problems.

3.A Appendix: Proofs of Results

3.A.1 Proof of Theorem 1

Confidence Intervals

We have Hoeffding's inequality relating the empirical mean and true mean of n Gaussian samples,

$$\mathsf{P}\left(\hat{\mu} - \mu > \epsilon\right) < \exp\left(\frac{-n\epsilon^2}{2\sigma^2}\right).$$

A similar inequality holds for the upper deviation as well. For our analysis, we need the probability of error to be at most $t^{-3} = \exp\left(-3\log t\right)$ for each arm. So, equating $\frac{-n\epsilon^2}{2\sigma^2} = -3\log t$ and replacing the unknown σ with known upper bound M, the required ϵ is

$$C_{i,t,n} = M\sqrt{\frac{6\log t}{n}}.$$

For bounding the variance, we use the following concentration inequalities for χ^2_k random variables with mean k (Laurent and Massart, 2000).

$$\begin{aligned}\mathsf{P}(X - k \geq 2\sqrt{kx} + 2x) &\leq \exp(-x),\\ \text{and } \mathsf{P}(k - X \geq 2\sqrt{kx}) &\leq \exp(-x).\end{aligned}$$

The sample variance $\hat{s}^2_n$ can be written as $\hat{s}^2_n = \frac{\sigma^2}{n-1}X$ for some χ^2_{n-1} random variable X. This gives

$$\begin{aligned}\mathsf{P}\left(\hat{s}^2_n - \sigma^2 \geq 2\sigma^2\sqrt{\frac{x}{n-1}} + 2\sigma^2\frac{x}{n-1}\right) &\leq \exp(-x),\\ \text{and } \mathsf{P}\left(\sigma^2 - \hat{s}^2_n \geq 2\sigma^2\sqrt{\frac{x}{n-1}}\right) &\leq \exp(-x).\end{aligned}$$

Similar to the above, we need to replace the unknown variance with its known upper bound and choose x such that the probability is bounded by $\exp(-3\log t)$. Combining this with the known lower and upper bounds on the variance gives us the confidence interval $(\sigma^2_{l,n}, \sigma^2_{u,n})$, where

$$\begin{aligned}\sigma^2_{l,n} &= \max\left\{\hat{s}^2_n - 2M^2\left[\frac{3\log t}{n-1} + \sqrt{\frac{3\log t}{n-1}}\right], N^2\right\},\\ \text{and } \sigma^2_{u,n} &= \min\left\{\hat{s}^2_n + 2M^2\sqrt{\frac{3\log t}{n-1}}, M^2\right\}.\end{aligned}$$

Regret

At every time step, the algorithm chooses the super arm with the highest upper confidence bound on its CVaR, constructed using upper confidence bounds on the means and lower confidence bounds on the standard deviations of the individual arms in the super arm.

Let S be the set of sub-optimal super arms. We adapt the proof technique by Chen et al. (2013) and maintain two counts $T_{i,t}$ and $N_{i,t}$ for each arm i. $T_{i,t}$ is the number of times the arm i has been pulled as part of any super arm, till (and including) time t. $N_{i,t}$ is set to 2 for all arms after the initialization phase. After that, at any time t, if a sub-optimal super arm $a \in S$ is pulled, then among all the arms in a, the arm with the lowest count of N_i has its count incremented by 1, i.e. $N_{i,t} = N_{i,t-1} + 1$ for exactly one of the arms in a. Ties are broken arbitrarily.

Clearly $N_{i,t} \leq T_{i,t}$ for all i and t. Also, when regret occurs due to sub-optimal super arm selection, N_i is incremented for some arm. So, tracking the regret becomes the problem of keeping track of the increments of N_i, leading to the following:

$$\begin{aligned}\mathcal{R}_{\mathsf{CVaR}_\alpha}(T) &= \mathsf{E}\left[\sum_{t=1}^{T} \mathbb{I}\left\{a_t \in S\right\} \Delta_{a_t}\right] \leq 2K + \mathsf{E}\sum_{t=1}^{T}\sum_{i=1}^{K} \mathbb{I}\left\{a_t \in S, N_{i,t} > N_{i,t-1}\right\} \Delta_{a_t} \\ &\leq (2K + K\ell)\Delta_{\max} + \mathsf{E}\sum_{t=1}^{T}\sum_{i=1}^{K} \mathbb{I}\left\{a_t \in S, N_{i,t} > N_{i,t-1}, N_{i,t-1} > \ell\right\} \Delta_{a_t},\end{aligned}$$

where a_t is the super arm chosen at time t. The second inequality holds by writing the first ℓ increments of each N_i as a separate term. Since each increment occurs for exactly one arm and N_j of all other arms in a_t are at least N_i, this can further be bounded by

$$\begin{aligned}&(2K + K\ell)\Delta_{\max} + \mathsf{E}\sum_{t=1}^{T} \mathbb{I}\left\{a_t \in S,\ N_{i,t-1} > \ell \text{ for all } i \in a_t\right\} \Delta_{a_t} \\ &\leq (2K + K\ell)\Delta_{\max} + \mathsf{E}\sum_{t=1}^{T} \mathbb{I}\left\{a_t \in S,\ T_{i,t-1} > \ell \text{ for all } i \in a_t\right\} \Delta_{a_t}.\end{aligned}$$

We introduce an event E_t defined to be true if the actual values of the mean and standard deviations of the individual arms fall within the respective confidence intervals at time t, i.e.,

$$E_t = \bigcap_{i\in[K]} \left\{|\hat{\mu}_{i,t} - \mu| \leq 2M\sqrt{\frac{6\log t}{T_{i,t-1}}}\right\} \cap \bigcap_{i\in[K]} \left\{\sigma_i^2 \leq \hat{s}^2_{i,T_{i,t-1}} + 2M^2\sqrt{\frac{3\log t}{T_{i,t-1}-1}}\right\}$$

$$\cap \bigcap_{i\in[K]} \left\{ \hat{s}^2_{i,T_{t-1}} - 2M^2 \left[\frac{3\log t}{T_{i,t-1}-1} + \sqrt{\frac{3\log t}{T_{i,t-1}-1}} \right] \leq \sigma_i^2 \right\}.$$

Each term in the summation above is bounded above by

$$\mathsf{E}\left[\, \mathbb{I}\left\{E_t, a_t \in S,\ T_{i,t-1} > \ell \text{ for all } i \in a_t\right\}\right] \Delta_{\max} + \mathsf{P}\{-E_t\}\Delta_{\max}.$$

For bounding the second term in this, union bound over the concentration inequalities gives

$$\mathsf{P}(-E_t) \leq \sum_{i=1}^{K} \sum_{T_{i,t-1}=1}^{t} 4t^{-3} \leq 4Kt^{-2}.$$

The first term is 1 if and only if a sub-optimal super arm a_t is selected despite the actual CVaR's falling inside their confidence bounds, and the number of arm reward samples $T_{i,t-1} \geq \ell$. So, the sub-optimal super arm a_t can be selected over the optimal super arm a^* only if

$$\sum_{i\in a^*} \left[\hat{\mu}_{i,T_{i,t-1}} + C_{i,t,T_{i,t-1}}\right] - \frac{\sigma_l^{(a^*)}}{\alpha}\varphi(\Phi^{-1}(\alpha)) < \sum_{i\in a_t} \left[\hat{\mu}_{i,T_{i,t-1}} + C_{i,t,T_{i,t-1}}\right] - \frac{\sigma_l^{(a_t)}}{\alpha}\varphi(\Phi^{-1}(\alpha)).$$

Since E_t holds and a_t is sub-optimal, the lower confidence bound for the CVaR of a_t satisfies

$$\begin{aligned}
\sum_{i\in a_t} \left[\hat{\mu}_{i,T_{i,t-1}} - C_{i,t,T_{i,t-1}}\right] - \frac{\sigma_u^{(a_t)}}{\alpha}\varphi(\Phi^{-1}(\alpha)) &\leq \mathsf{CVaR}(a_t) < \mathsf{CVaR}(a^*) \\
&= \sum_{i\in a^*} \mu_{i,T_{i,t-1}} - \frac{\sigma^{(a^*)}}{\alpha}\varphi(\Phi^{-1}(\alpha)) \\
&\leq \sum_{i\in a^*} \left[\hat{\mu}_{i,T_{i,t-1}} + C_{i,t,T_{i,t-1}}\right] - \frac{\sigma_l^{(a^*)}}{\alpha}\varphi(\Phi^{-1}(\alpha)) \\
&< \sum_{i\in a_t} \left[\hat{\mu}_{i,T_{i,t-1}} + C_{i,t,T_{i,t-1}}\right] - \frac{\sigma_l^{(a_t)}}{\alpha}\varphi(\Phi^{-1}(\alpha)),
\end{aligned}$$

which implies that $\Delta_{a_t} = \mathsf{CVaR}(a^*) - \mathsf{CVaR}(a_t)$ is bounded by

$$\begin{aligned}
&\sum_{i\in a_t} \left[\hat{\mu}_{i,T_{i,t-1}} + C_{i,t,T_{i,t-1}}\right] - \frac{\sigma_l^{(a_t)}}{\alpha}\varphi(\Phi^{-1}(\alpha)) - \sum_{i\in a_t} \left[\hat{\mu}_{i,T_{i,t-1}} - C_{i,t,T_{i,t-1}}\right] + \frac{\sigma_u^{(a_t)}}{\alpha}\varphi(\Phi^{-1}(\alpha)) \\
&\qquad = \sum_{i\in a_t} 2\, C_{i,t,T_{i,t-1}} + \frac{\sigma_u^{(a_t)} - \sigma_l^{(a_t)}}{\alpha}\varphi(\Phi^{-1}(\alpha)).
\end{aligned}$$

This bound depends on the number of samples $T_{i,t-1}$. Since $T_{i,t-1} > \ell$, for sufficiently large ℓ the confidence intervals become too small, and the above inequality cannot hold, making the corresponding term in the regret bound 0. For this to happen, it is sufficient that

$$C_{i,t,T_{i,\ell}} < \frac{\Delta_{\min}}{4L} \text{ and } \sigma_u^{(a_t)} - \sigma_l^{(a_t)} < \frac{\alpha\Delta_{\min}}{2\varphi(\Phi^{-1}(\alpha))}.$$

Since $C_{i,t,\ell} = M\sqrt{\frac{6\log t}{\ell}}$, for the first inequality above,

$$M\sqrt{\frac{6\log t}{\ell}} < \frac{\Delta_{\min}}{4L}, \text{ or } \ell > \frac{96M^2L^2\log T}{\Delta_{\min}^2}.$$

To obtain the final regret bound, we need ℓ such that

$$\sigma_u^{(a_t)} - \sigma_l^{(a_t)} < \frac{\alpha\Delta_{\min}}{2\varphi(\Phi^{-1}(\alpha))},$$

the indicator term involving the event E_t vanishes. The confidence intervals of the super arms are defined in terms of those from the constituent arms. For confidence intervals constructed using n samples from a Gaussian distribution, we have the following lower and upper confidence bounds on the variance, respectively.

$$\sigma_l^2 = \hat{s}_n^2 - 2M^2\left[\frac{3\log t}{n-1} + \sqrt{\frac{3\log t}{n-1}}\right], \text{ and}$$
$$\sigma_u^2 = \hat{s}_n^2 + 2M^2\sqrt{\frac{3\log t}{n-1}},$$

Writing the confidence intervals of the variance of the super arms a_t as $\sigma_u^{(a_t)} = \|(\sigma_{u,i})_{i\in a_t}\|_2$ and $\sigma_l^{(a_t)} = \|(\sigma_{l,i})_{i\in a_t}\|_2$, where

$$\sigma_{u,i}^2 = \min\left\{\left(s_{i,T_{i,t-1}}^2 + 2M^2\sqrt{\frac{3\log t}{T_{i,t-1}-1}}\right), M^2\right\} \text{ and}$$
$$\sigma_{l,i}^2 = \max\left\{\left(s_{i,T_{i,t-1}}^2 - 2M^2\left[\frac{3\log t}{T_{i,t-1}-1} + \sqrt{\frac{3\log t}{T_{i,t-1}-1}}\right]\right), N^2\right\},$$

$$\text{gives us } \sigma_u^{(a_t)} - \sigma_l^{(a_t)} \leq \sqrt{\sum_{i\in a_t}(\sigma_{u,i} - \sigma_{l,i})^2}, \tag{3.2}$$

a bound based on the confidence interval sizes of the constituent arms. Using

$$\sqrt{b}-\sqrt{a}=\int_a^b \frac{1}{2\sqrt{x}}\mathrm{d}x \leq \frac{1}{2\sqrt{a}}\int_a^b \mathrm{d}x = \frac{b-a}{2\sqrt{a}}, \text{ we obtain}$$

$$\begin{aligned}\sigma_{u,i}-\sigma_{l,i} \leq \frac{1}{2\sqrt{N^2}}\Bigg(&\min\left\{\left(s^2_{i,T_{i,t-1}}+2M^2\sqrt{\frac{3\log t}{T_{i,t-1}-1}}\right), M^2\right\} \\ &- \max\left\{\left(s^2_{i,T_{i,t-1}}-2M^2\left[\frac{3\log t}{T_{i,t-1}-1}+\sqrt{\frac{3\log t}{T_{i,t-1}-1}}\right]\right), N^2\right\}\Bigg) \\ \leq \frac{M^2}{N}&\left(\frac{3\log t}{T_{i,t-1}-1}+2\sqrt{\frac{3\log t}{(T_{i,t-1}-1)}}\right).\end{aligned}$$

So, from (3.2), and the fact that a_t has at most L arms and $T_{i,t-1} > \ell$,

$$\sigma_u^{(a_t)}-\sigma_l^{(a_t)} \leq \sqrt{L}\frac{M^2}{N}\left(\frac{3\log t}{\ell-1}+2\sqrt{\frac{3\log t}{(\ell-1)}}\right)$$

So, for making sure that the confidence bounds are sufficiently small so as to preclude sub-optimal choices (provided E_t is true), ℓ should be chosen such that

$$\begin{aligned}\sqrt{L}\frac{M^2}{N}\left(\frac{3\log t}{\ell-1}+2\sqrt{\frac{3\log t}{(\ell-1)}}\right) &< \frac{\alpha\Delta_{\min}}{2\varphi(\Phi^{-1}(\alpha))}, \\ \text{i.e, } \sqrt{\frac{3\log t}{\ell-1}}\left[\sqrt{\frac{3\log t}{\ell-1}}+2\right] &< \frac{\alpha N\Delta_{\min}}{\sqrt{L}M^2\varphi(\Phi^{-1}(\alpha))}.\end{aligned}$$

This means that ℓ should be at least of the order of

$$\mathcal{O}\left(\frac{\sqrt{L}M^2\varphi(\Phi^{-1}(\alpha))\log T}{\alpha N\Delta_{\min}}\right).$$

Combining both constraints on ℓ and the bound on the probability of event $\{-E_t\}$, and using

$\sum_t \frac{1}{t^2} \leq \frac{\pi^2}{6}$ gives

$$\mathcal{R}_{\mathsf{CVaR}_\alpha}(T) < \left(2 + \frac{2\pi^2}{3}\right) K\Delta_{\max} + \mathcal{O}\left(\frac{\sqrt{L}M^2 K \log T\Delta_{\max}}{\Delta_{\min}}\right) \max\left\{\frac{96L\sqrt{L}}{\Delta_{\min}}, \mathcal{O}\left(\frac{\varphi(\Phi^{-1}(\alpha))}{\alpha N}\right)\right\}.$$

3.A.2 Proof of Theorem 2

The regret of the algorithm can be rewritten as

$$\mathcal{R}_{\mathsf{CVaR}_\alpha}(T) = \mathsf{E}\left[\sum_{t=1}^{T} \mathbb{I}\{\mathcal{E}_t\}\Delta_{a_t}\right] + \mathsf{E}\left[\sum_{t=1}^{T} \mathbb{I}\{\neg\mathcal{E}_t\}\Delta_{a_t}\right],$$

where a_t is the super arm chosen at time t, Δ_a is the CVaR gap, and

$$\mathcal{E}_t = \left\{\text{There exists arm } j \in [K] \text{ s.t } \sup_{x\in[0,1]} \left|\hat{F}_{j,T_{j,t-1}}(x) - F_j(x)\right| \geq C_{j,t-1,T_{j,t-1}}\right\}$$

is the "bad" event that the empirical distribution and true distribution of some arms are not close at some point.

The DKW inequality states that the cumulative distribution function F of a random variable is close to an empirical cumulative distribution function $\hat{F}_n$ obtained using n samples with probability given as follows:

$$\mathsf{P}\left(\sup_{x\in\mathbb{R}} |F_n(x) - F(x)| > \epsilon\right) \leq 2\, e^{-2n\epsilon^2}.$$

We use this to bound the first term of the regret. For $C_{i,t-1,T_{i,t-1}} = \sqrt{\frac{3\log t}{2T_{i,t-1}}}$,

$$\begin{aligned}\mathsf{P}\left(\mathcal{E}_t\right) &\leq \sum_{j=1}^{K}\sum_{s=1}^{t-1} \mathsf{P}\left(\sup_{x\in[0,1]} \left|\hat{F}_{j,s}(x) - F_j(x)\right| \geq C_{j,t-1,s}\right) \\ &\leq \sum_{j=1}^{K}\sum_{s=1}^{t-1} 2\exp\left(-2sC_{j,t-1,s}^2\right) = \sum_{j=1}^{K}\sum_{s=1}^{t-1} \frac{2}{t^3} \leq \frac{2K}{t^2}.\end{aligned}$$

For the second term, since $\mathcal{E}_t$ does not occur, we have, for every arm i,

$$\tilde{F}_i(x) < F_i(x) < \tilde{F}(x) + C_{i,t-1,T_{i,t-1}}, \qquad \text{for each arm } i \in [K],$$
$$\text{which implies that } \tilde{F}_a(x) < F_a(x) < \tilde{F}_a(x) + 2\sum_{j\in a} C_{j,t-1,T_{j,t-1}} \qquad \text{for all super arms } a \in \mathcal{A}.$$

Using Proposition F.1 of Cassel et al. (2023) on CVaR stability that relates differences in cumulative distribution functions with a difference in the CVaR, we obtain

$$\mathsf{CVaR}(F_a) < \mathsf{CVaR}(\tilde{F}_a) < \mathsf{CVaR}(F_a) + L \cdot w\left(2\sum_{i\in a} C_{i,t-1,T_{i,t-1}}\right),$$

$$\text{where } w(x) = b.(x + x^2), \text{ and } b = \frac{4}{\alpha \min\{\alpha, 1-\alpha\}}.$$

Now, since only sub-optimal super arms a_t that have been chosen at time t contribute to the regret, we have

$$\begin{aligned}
0 < \Delta_{a_t} &= \mathsf{CVaR}(F_{a^*}) - \mathsf{CVaR}(F_{a_t}) \\
&< \mathsf{CVaR}(\tilde{F}_{a^*}) - \mathsf{CVaR}(\tilde{F}_{a_t}) + L.w\left(2\sum_{i\in a_t} C_{i,t-1,T_{i,t-1}}\right) \\
&\leq Lb\left(2\sum_{j\in a_t} C_{j,t-1,T_{j,t-1}} + \left(2\sum_{j\in a_t} C_{j,t-1,T_{j,t-1}}\right)^2\right) \\
&\leq \phi_L^{-1}\left(2\sum_{j\in a_t} C_{j,t-1,T_{j,t-1}}\right),
\end{aligned}$$

where $\phi_L^{-1}(x) = \max\{2Lbx, 2Lbx^2\}$, and $\phi_L(y) = \min\{\frac{y}{2Lb}, \sqrt{\frac{y}{2Lb}}\}$. So, we have

$$0 < \phi_L(\Delta_{a_t}) < 2\sum_{i\in a_t} C_{i,t-1,T_{i,t-1}} = 2\sum_{i\in a_t}\sqrt{\frac{3\log t}{2T_{i,t-1}}}.$$

$$\begin{aligned}
\text{So, } \mathcal{R}_{\mathsf{CVaR}_\alpha}(T) &= \mathsf{E}\left[\sum_{t=1}^{T} \mathbb{I}\{\Delta_{a_t} \neq 0\}\Delta_{a_t}\right] \\
&\leq K\Delta_{\max} + \mathsf{E}\left[\sum_{t=K+1}^{T} \mathbb{I}\{\neg\mathcal{E}_t\}\Delta_{a_t}\right] + \Delta_{\max}\sum_{t=K+1}^{T} \mathbb{P}\{\mathcal{E}_t\}
\end{aligned}$$

$$\leq K\Delta_{\max} + \mathsf{E}\left[\sum_{t=K+1}^{T} \mathbb{I}\{\neg\mathcal{E}_t\}\Delta_{a_t}\right] + \Delta_{\max}\sum_{t=K+1}^{T}\frac{2k}{t^2}$$
$$\leq \left(1+\frac{\pi^2}{3}\right)K\Delta_{\max} + \mathsf{E}\left[\sum_{t=K+1}^{T}\mathbb{I}\left\{0<\phi_L(\Delta_{a_t})<2\sum_{i\in a_t}\sqrt{\frac{3\log t}{2T_{i,t-1}}}\right\}\Delta_{a_t}\right].$$

Following Chen et al. (2016), we define the event

$$\mathcal{H}_t = \left\{0<\phi_L(\Delta_{a_t})<2\sum_{i\in a_t}\sqrt{\frac{3\log t}{2T_{i,t-1}}}\right\},$$

and two decreasing sequences of constants: $1=\beta_0>\beta_1>\ldots$ and $\alpha_1>\alpha_2>\ldots$, such that $\lim_{k\to\infty}\alpha_k=0, \lim_{k\to\infty}\beta_k=0$, and

$$\sqrt{6}\sum_{k=1}^{\infty}\frac{\beta_{k-1}-\beta_k}{\sqrt{\alpha_k}}\leq 1, \qquad \sum_{k=1}^{\infty}\frac{\alpha_k}{\beta_k}<267.$$

For $t\in\{K+1,\ldots,T\}$, let

$$m_{k,t} = \begin{cases} \alpha_k\frac{L^2\log T}{\left(\phi_L(\Delta_{a_t})\right)^2} & \Delta_{a_t}>0, \\ +\infty & \Delta_{a_t}=0, \end{cases}$$
$$\text{and } A_{k,t}=\{i\in a_t \text{ s.t. } T_{i,t-1}\leq m_{k,t}\}.$$

Define the event

$$\mathcal{G}_{k,t}=\{|A_{k,t}|\geq\beta_k L\}.$$

Lemma 2. *In the t'th round, if $\mathcal{H}_t$ occurs, then there exists $k\in\mathbb{N}$ such that event $\mathcal{G}_{k,t}$ happens.*

Proof. Assume $\mathcal{H}_t$ happens and none of $\mathcal{G}_{k,t}$ happens. Let $A_{0,t}=a_t$ and $\bar{A}_{k,t}=a_t\setminus A_{k,t}$. Then, since $\lim_{k\to\infty}m_{k,t}=0$, following Lemma 5 in Chen et al. (2016),

$$\sum_{i\in a_t}\frac{1}{\sqrt{T_{i,t-1}}}<\sum_{k=1}^{\infty}\frac{(\beta_{k-1}-\beta_k)L}{\sqrt{m_{k,t}}}.$$

$$\text{This implies } \phi_L(\Delta_{a_t})<2\sum_{i\in a_t}\sqrt{\frac{3\log t}{2T_{i,t-1}}}\leq\sqrt{6\log T}\sum_{i\in a_t}\frac{1}{\sqrt{T_{i,t-1}}}$$

$$< \sqrt{6 \log T} \sum_{k=1}^{\infty} \frac{(\beta_{k-1} - \beta_k) L}{\sqrt{m_{k,t}}}$$
$$= \sqrt{6} \sum_{k=1}^{\infty} \frac{\beta_{k-1} - \beta_k}{\sqrt{\alpha_k}} \phi_L(\Delta_{a_t}) < \phi_L(\Delta_{a_t}),$$

which is a contradiction, hence the Lemma. □

$\mathcal{G}_{k,t}$ is the event that at least $\beta_k K$ arms in a_t do not have "enough" ($> m_{k,t}$) number of samples. Let $\mathcal{G}_{i,k,t}$ be the corresponding event for a specific arm $i \in a_t$. That is, $\mathcal{G}_{i,k,t} = \mathcal{G}_{k,t} \wedge \{i \in a_t, T_{i,t-1} \leq m_{k,t}\}$. So, when $\mathcal{G}_{k,t}$ occurs, $\mathcal{G}_{i,k,t}$ occurs for at least $\beta_k K$ arms, i.e,

$$\mathbb{I}\{\mathcal{G}_{k,t}, \Delta_{a_t} > 0\} \leq \frac{1}{\beta_k L} \sum_{i \in a_B} \mathbb{I}\left\{\mathcal{G}_{i,k,t}, \Delta_{a_t} > 0\right\},$$
$$\text{and so } \sum_{t=K+1}^{T} \mathbb{I}\{\mathcal{H}_t\} \Delta_{a_t} \leq \sum_{i \in a_B} \sum_{k=1}^{\infty} \sum_{t=K+1}^{T} \mathbb{I}\left\{\mathcal{G}_{i,k,t}, \Delta_{a_t} > 0\right\} \frac{\Delta_{a_t}}{\beta_k L}$$

For each arm $i \in a_B$, let the arm be contained in N_i sub-optimal super arms $a^B_{i,1}, a^B_{i,2}, \ldots, a^B_{i,N_i}$. Let $\Delta_{i,l} = \Delta_{a^B_{i,l}}$ for $l \in [N_i]$, and without any loss in generality

$$+\infty = \Delta_{i,0} \geq \Delta_{i,1} \geq \Delta_{i,2} \geq \cdots \geq \Delta_{i,N_i} = \Delta_{i,min}.$$

Then, following Chen et al. (2016), we have

$$\sum_{t=K+1}^{T} \mathbb{I}\{\mathcal{H}_t\} \Delta_{a_t} \leq \sum_{i \in a_B} \sum_{k=1}^{\infty} \sum_{t=K+1}^{T} \mathbb{I}\left\{\mathcal{G}_{i,k,t}, \Delta_{a_t} > 0\right\} \frac{\Delta_{a_t}}{\beta_k L}$$
$$\leq 267 L \log T \sum_{i \in a_B} \sum_{j=1}^{N_i} \left(\frac{1}{\left(\phi_L(\Delta_{i,j})\right)^2} - \frac{1}{\left(\phi_L(\Delta_{i,j-1})\right)^2} \right) \Delta_{i,j}.$$

Finally, for each $i \in a_B$, we have

$$\sum_{j=1}^{N_i} \left(\frac{1}{\left(\phi_L(\Delta_{i,j})\right)^2} - \frac{1}{\left(\phi_L(\Delta_{i,j-1})\right)^2} \right) \Delta_{i,j} = \frac{\Delta_{i,N_i}}{\left(\phi_L(\Delta_{i,N_i})\right)^2} + \sum_{j=1}^{N_i - 1} \frac{1}{\left(\phi_L(\Delta_{i,j})\right)^2} \left(\Delta_{i,j} - \Delta_{i,j+1}\right)$$
$$\leq \frac{\Delta_{i,N_i}}{\left(\phi_L(\Delta_{i,N_i})\right)^2} + \int_{\Delta_{i,N_i}}^{\Delta_{i,1}} \frac{1}{\left(\phi_L(x)\right)^2} \mathrm{d}x.$$

Now, $\phi_L(x) = \min\{\frac{x}{2Lb}, \sqrt{\frac{x}{2Lb}}\}$. For $x \in [\Delta_{i,N_i}, \Delta_{i,1}]$, $x \leq L$, and so $\frac{x}{2Lb} \leq \frac{1}{2b} < 1$, therefore,

$\phi_L(x) = \frac{x}{2Lb}$, and

$$\sum_{j=1}^{N_i} \left(\frac{1}{\left(\phi_L(\Delta_{i,j})\right)^2} - \frac{1}{\left(\phi_L(\Delta_{i,j-1})\right)^2} \right) \Delta_{i,j} \leq 4L^2b^2 \frac{\Delta_{i,N_i}}{\left(\Delta_{i,N_i}\right)^2} + 4L^2b^2 \int_{\Delta_{i,N_i}}^{\Delta_{i,1}} \frac{1}{x^2} \mathrm{d}x < \frac{8L^2b^2}{\Delta_{i,\min}}.$$

$$\text{Therefore, } \sum_{t=K+1}^{T} \mathbb{I}\{\mathcal{H}_t\} \Delta_{a_t} \leq 2136 L^3 b^2 \log T \sum_{i \in a_B} \frac{1}{\Delta_{i,\min}}, \text{ resulting in}$$

$$\mathcal{R}_{\mathsf{CVaR}_\alpha}(T) \leq C \frac{L^3}{\alpha^4} \log T \sum_{i \in a_B} \frac{1}{\Delta_{i,min}} + \left(1 + \frac{\pi^2}{3}\right) K \Delta_{\max}.$$

3.A.3 Proof of Theorem 3

Discretization (Proof of Lemma 1)

Let $(f_i)_{i\in[L]}$ be L probability mass functions and f' be the corresponding discrete distributions obtained by discretizing the support to points spaced ϵ apart by rounding up. The probability mass at each point x in the support of f_i is shifted to $\lceil \frac{x}{\epsilon} \rceil \epsilon$. Multiple points might be shifted to the same multiple of ϵ, but for the sake of analysis, we keep track of all the original points and their probability masses individually, even after merging.

Let $x_1, x_2, \ldots x_L$ be points in the support of $f_1, f_2, \ldots, f_L$ respectively. Then $\sum_{i\in[L]} x_i \in \mathsf{Supp}\left(\bigoplus_{i\in[L]} f_i\right)$, with probability mass $\prod_{i\in[L]} f_i(x_i)$ due to these L specific points. Let $x'_1, x'_2, \ldots x'_L$ be the points obtained by ϵ-rounding up the corresponding points. These points are in the support of $f'_1, f'_2, \ldots, f'_L$ respectively, and $\sum_{i\in[L]} x'_i \in \mathrm{Supp}\left(\bigoplus_{i\in[L]} f'_i\right)$. Now,

$$\begin{aligned}
\sum_{i\in[L]} x'_i - \sum_{i\in[L]} x_i &= \sum_{i=1}^{L} \left\lceil \frac{x_i}{\epsilon} \right\rceil \epsilon - \sum_{i=1}^{L} x_i \\
&< \left[\sum_{i=1}^{L} \left\lceil \frac{x_i}{\epsilon} \right\rceil - \left(\left\lceil \sum_{i=1}^{L} \frac{x_i}{\epsilon} \right\rceil - 1 \right) \right] \epsilon \\
&\leq \left[\sum_{i=1}^{L} \left\lceil \frac{x_i}{\epsilon} \right\rceil - \left(\left\lceil \sum_{i=1}^{L-1} \frac{x_i}{\epsilon} \right\rceil + \left\lceil \frac{x_L}{\epsilon} \right\rceil - 1 - 1 \right) \right] \epsilon \\
&\leq \left[\sum_{i=1}^{L} \left\lceil \frac{x_i}{\epsilon} \right\rceil - \left(\left\lceil \sum_{i=1}^{L-j} \frac{x_i}{\epsilon} \right\rceil + \sum_{i=L-j+1}^{L} \left(\left\lceil \frac{x_i}{\epsilon} \right\rceil - 1 \right) - 1 \right) \right] \epsilon \\
&\leq \left[\sum_{i=1}^{L} \left\lceil \frac{x_i}{\epsilon} \right\rceil - \left(\sum_{i=1}^{L} \left(\left\lceil \frac{x_i}{\epsilon} \right\rceil - 1 \right) - 1 \right) \right] \epsilon = (L+1)\epsilon.
\end{aligned}$$

That is, the sum of the samples after discretization is no further than distance $(L+1)\epsilon$ to the right of the original sum. Now, we study how discretization affects CVaR. For discrete X,

$$\mathsf{CVaR}_\alpha(X) = \frac{1}{\alpha}\left[\sum_{x \le x_\alpha} x f_X(x) - \left(\sum_{x \le x_\alpha} f_X(x) - \alpha\right) x_\alpha\right],$$

where $x_\alpha = \mathsf{VaR}_\alpha(X) = \inf\{x : F_X(x) \ge \alpha\}$. Now, we consider the CVaR of the discretized distribution X'. Including the multiplicative factor $L+1$ inside ϵ, since the probability mass shifted rightwards by at most ϵ, we have $x_\alpha \le x'_\alpha < x_\alpha + \epsilon$.

Now, for determining the CVaR difference, we divide the points in the support of X based on whether they contribute to CVaR of X and X' (before and after the shift). Points contribute to the CVaR only if they are at most the VaR.

A: Points that contribute to the CVaR of X

B: Points that contribute to the CVaR of X'

So, the set of all points is

$$(A \setminus B) \cup (B \setminus A) \cup (A \cap B) \cup (A^c \cap B^c).$$

Considering the contributions of $A \cap B$,

$$\begin{aligned}\sum_{x \in A \cap B} x' f_{X'}(x') - x'_\alpha \sum_{x \in A \cap B} f_{X'}(x') &\le \sum_{x \in A \cap B} ((x+\epsilon) f_X(x)) - x_\alpha \sum_{x \in A \cap B} f_{X'}(x') \\ &= \sum_{x \in A \cap B} x f_X(x) - x_\alpha \sum_{x \in A \cap B} f_X(x) + \epsilon \sum_{x \in A \cap B} f_X(x).\end{aligned}$$

The points in $B \setminus A$, those that contribute to X' but not X, are those that are not greater than x'_α, but are greater than x_α. Their contribution is at most zero because

$$\sum_{x > x_\alpha, x' \le x'_\alpha} (x' f_{X'}(x')) - x'_\alpha \sum_{x > x_\alpha, x' \le x'_\alpha} f_{X'}(x') \le x'_\alpha \sum_{x > x_\alpha, x' \le x'_\alpha} (f_{X'}(x')) - x'_\alpha \sum_{x > x_\alpha, x' \le x'_\alpha} f_{X'}(x').$$

So, $\mathsf{CVaR}_\alpha(X')$ can be written as

$$\begin{aligned}&\frac{1}{\alpha}\Bigg[\sum_{x \in A \cap B} x' f'_X(x') + \sum_{x \in B \setminus A} x' f'_X(x') - \left(\sum_{x' \in A \cap B} f'_X(x') + \sum_{x' \in B \setminus A} f'_X(x') - \alpha\right) x'_\alpha\Bigg] \\ &\le \frac{1}{\alpha}\Bigg[\sum_{x \in A \cap B} x f_X(x) - x_\alpha \sum_{x \in A \cup B} f'_X(x') - \alpha x'_\alpha + \epsilon \sum_{x \in A \cap B} f_X(x)\Bigg]\end{aligned}$$

$$= \frac{1}{\alpha}\Bigg[\sum_{x\in A} x f_X(x) - x_\alpha \sum_{x\in A} f'_X(x') - \alpha x'_\alpha - \sum_{x\in A\setminus B} x f_X(x) + x_\alpha \sum_{x\in A\setminus B} f'_X(x') + \epsilon \sum_{x\in A\cap B} f_X(x)\Bigg].$$

Now, for $x \in A \setminus B$, $x + \epsilon \geq x' > x'_\alpha$, so $-x < -x'_\alpha + \epsilon$, giving

$$\begin{aligned}\mathsf{CVaR}_\alpha(X') &\leq \frac{1}{\alpha}\Bigg[\sum_{x\in A} x f_X(x) - x_\alpha \sum_{x\in A} f'_X(x') - \alpha x'_\alpha - x'_\alpha \sum_{x\in A\setminus B} f_X(x) + x_\alpha \sum_{x\in A\setminus B} f'_X(x') \\ &\qquad + \epsilon \sum_{x\in A\cap B} f_X(x) + \epsilon \sum_{x\in A\setminus B} f_X(x)\Bigg] \\ &\leq \frac{1}{\alpha}\left[\sum_{x\in A} x f_X(x) - x_\alpha \sum_{x\in A} f_X(x) - \alpha x_\alpha\right] + \frac{\epsilon}{\alpha}\sum_{x\in A} f_X(x) \leq \mathsf{CVaR}_\alpha(X) + \frac{\epsilon}{\alpha}.\end{aligned}$$

$$\text{Therefore, } \mathsf{CVaR}_\alpha(\tilde{F}_a) \leq \mathsf{CVaR}_\alpha(F'_a) \leq \mathsf{CVaR}_\alpha(\tilde{F}_a) + \frac{\epsilon(L+1)}{\alpha}.$$

Regret

We proceed similar to the proof of Theorem 2, writing the regret as

$$\mathcal{R}_{\mathsf{CVaR}_\alpha}(T) = \mathsf{E}\left[\sum_{t=1}^{T} \mathbb{I}\{\mathcal{E}_t\}\Delta_{a_t}\right] + \mathsf{E}\left[\sum_{t=1}^{T} \mathbb{I}\{\neg\mathcal{E}_t\}\Delta_{a_t}\right],$$

with a_t, Δ_a and $\mathcal{E}_t$ as defined previously. For the first term, we know from the previous analysis that $\mathsf{P}(\mathcal{E}_t) \leq \frac{2K}{t^2}$. For the second term, since $\mathcal{E}_t$ does not occur, we have

$$\begin{aligned}\tilde{F}_a(x) < \quad F_a(x) \quad < \tilde{F}_a(x) + 2\sum_{j\in a} C_{j,t-1,T_{j,t-1}}, \\ \text{and hence, } \mathsf{CVaR}(F_a) < \mathsf{CVaR}(\tilde{F}_a) < \mathsf{CVaR}(F_a) + L.w\Big(2\sum_{j\in a} C_{j,t-1,T_{j,t-1}}\Big).\end{aligned}$$

Now, since a sub-optimal super arm has been chosen based on the discretized distributions F'_a,

$$\begin{aligned}\Delta_{a_t} &= \mathsf{CVaR}(F_{a^*}) - \mathsf{CVaR}(F_{a_t}) \\ &< \mathsf{CVaR}(\tilde{F}_{a^*}) - \mathsf{CVaR}(\tilde{F}_{a_t}) + Lw\left(2\sum_{j\in a} C_{j,t-1,T_{j,t-1}}\right) \\ &\leq \mathsf{CVaR}(F'_{a^*}) - \mathsf{CVaR}(F'_{a_t}) + \frac{\epsilon(L+1)}{\alpha} + Lw\left(2\sum_{j\in a_t} C_{j,t-1,T_{j,t-1}}\right)\end{aligned}$$

$$\leq \frac{\epsilon(L+1)}{\alpha} + \phi_L^{-1}\left(2\sum_{j\in a_t} C_{j,t-1,T_{j,t-1}}\right),$$

which implies either $\frac{\Delta_{a_t}}{2} \leq \phi_L^{-1}\left(2\sum_{j\in a_t} C_{j,t-1,T_{j,t-1}}\right)$ or $\frac{\Delta_{a_t}}{2} \leq \frac{\epsilon(L+1)}{\alpha}$.

When $\frac{\Delta_{a_t}}{2} \leq \phi_L^{-1}\left(2\sum_{j\in a_t} C_{j,t-1,T_{j,t-1}}\right)$, we have

$$0 < \phi_L\left(\frac{\Delta_{a_t}}{2}\right) < 2\sum_{i\in a_t} C_{i,t-1,T_{i,t-1}} = 2\sum_{i\in a_t}\sqrt{\frac{3\log t}{2T_{i,t-1}}}.$$

So, the regret is bounded by

$$K\Delta_{\max} + \Delta_{\max}\sum_{t=K+1}^{T}\frac{2k}{t^2} + \mathsf{E}\left[\sum_{t=K+1}^{T}\mathbb{I}\left\{0 < \phi_L(\Delta_{a_t}) < 2\sum_{i\in a_t}\sqrt{\frac{3\log t}{2T_{i,t-1}}}\right\}\Delta_{a_t}\right]$$
$$+ \mathsf{E}\left[\sum_{t=K+1}^{T}\mathbb{I}\left\{\frac{\Delta_{a_t}}{2} \leq \frac{\epsilon(L+1)}{\alpha}\right\}\Delta_{a_t}\right].$$

From the proof of Theorem 2, for some constant C,

$$\mathsf{E}\left[\sum_{t=K+1}^{T}\mathbb{I}\left\{0 < \phi_L(\Delta_{a_t}) < 2\sum_{i\in a_t}\sqrt{\frac{3\log t}{2T_{i,t-1}}}\right\}\Delta_{a_t}\right] \leq CL^3b^2\log T\sum_{i\in a_B}\frac{1}{\Delta_{i,\min}}.$$

Therefore,

$$\mathcal{R}_{\mathsf{CVaR}_\alpha}(T) \leq \left(1+\frac{\pi^2}{3}\right)K\Delta_{\max} + CL^3b^2\log T\sum_{i\in a_B}\frac{1}{\Delta_{i,\min}} + 2\epsilon\frac{(L+1)T}{\alpha}.$$

Letting $\epsilon = \frac{\alpha}{(L+1)T}$ proves Theorem 3.

Chapter 4

Hierarchical Reinforcement Learning with Offline Data

Standard reinforcement learning algorithms with a single policy perform poorly on tasks in complex environments involving sparse rewards, diverse behaviors, or long-term planning. This motivates the study of algorithms that incorporate temporal abstraction by training a hierarchy of policies that plan over different time scales. The options framework (Sutton et al., 1999) has been introduced to implement such temporal abstraction by learning low-level options that act as extended actions controlled by a high-level policy.

Consider a Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, r, p, \rho, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, P is the probability transition function, ρ is the initial state distribution and γ is the discount factor.

In the options framework, the agent consists of a set Ω of options. Each option $\omega \in \Omega$ is defined by a tuple $(\mathcal{I}_\omega, \pi_\omega, \beta_\omega)$, where $\mathcal{I}_\omega$ is the set of initial states from which ω can be given control, π_ω is the actual policy corresponding to ω, and β_ω is the termination function that decides when the option ω relinquishes control and passes it back to the high-level policy. In most approaches, the initiation set $\mathcal{I}$ is the entire state space.

Extending the standard actor-critic approach, options framework approaches consider option-value functions $Q_\Omega(s, \omega), Q_U(s, \omega, a)$ and $V_\Omega(s)$, and the option-value function upon arrival $U(s', \omega)$ that incorporates the probability of the previous option ω terminating at state s' immediately upon arrival. These functions lead to analogous Bellman and policy gradient equations for updating the critics and actors, respectively.

The main challenge in applying these algorithms to real-world problems is that they suffer from high sample complexity to train multiple levels of the hierarchy, which is impossible in

many online settings. Motivated by this, we propose an offline hierarchical RL method that can learn options from existing offline datasets collected by other unknown agents. This is a very challenging problem due to the distribution mismatch between the learned options and the policies responsible for the offline dataset, and to our knowledge, this is the first work in this direction. We propose a meta-algorithm by which an online hierarchical reinforcement learning algorithm can be trained on an offline dataset of transitions collected by an unknown behavior policy. The results presented in this chapter can also be found in Ayyagari et al. (2024).

Standard hierarchical algorithms require the agent to repeatedly interact with the environment to obtain diverse transitions. However, one might wish to avoid collecting any transitions and instead use a dataset of existing transitions collected by an unknown behavior policy. This setting is called Offline Reinforcement Learning.

There are many issues with learning optimal policies from such a dataset. Evaluating a policy using transitions collected by a different policy leads to a distributional shift. Learning an optimal policy is prevented by limited coverage of the dataset due to the behavior policy being sub-optimal or narrow.

Many approaches have been proposed to deal with these issues, most of which incorporate pessimism into the learning process in various ways. In our work, we consider Model-Based Offline Reinforcement Learning (MOReL) (Kidambi et al., 2020), in which a pessimistic MDP (P-MDP) that is an approximation of the true MDP is learned from the dataset and a model-based Natural Policy Gradient algorithm is then used to learn an optimal policy in the P-MDP. The performance of a policy in the pessimistic MDP is an approximate lower bound on its performance in the unknown real MDP.

4.1 Proposed Approach

Consider an offline dataset $\mathcal{D}$ created by exploring the environment using an unknown behavior policy π_b. The dataset is a collection of transitions and is of the form

$$\mathcal{D} = \{(s_i, a_i, r_i, s'_i, d_i)\}_{i=1}^{N},$$

where $s_i \in \mathcal{S}$ is some state of the environment, a_i is the action taken by π_b at s_i, r_i is the reward obtained, s'_i is the next state the environment transitioned into, and d_i is a binary variable indicating whether this transition resulted in the termination of the episode.

Such a dataset can be directly used to learn a standard reinforcement learning agent, sometimes even naively, because such an agent learns its own policy π whose behavior can be inferred from the response of the environment to π_b.

However, when a hierarchy of policies needs to be learned, such as in the Option-Critic architecture (Bacon et al., 2017), the performance of a high-level policy depends on the current set of options, and it is not possible to directly learn this value since the options keep changing and the dataset does not have information about how any current option would perform long-term. This is as opposed to the fact that the dataset contains information on the reward obtained by a low-level action taken by a flat policy.

So, to convert an arbitrary online hierarchical reinforcement learning algorithm into an offline algorithm, there is a compelling reason to learn the MDP model, more so than in the case of learning a flat agent. For this reason, we learn a P-MDP, a pessimistic approximation of the MDP that terminates with a reward penalty in regions of the state space where it is uncertain about the true MDP.

The pessimism of the P-MDP explicitly restricts the agent from exploring regions of the environment's state-action space that are not represented in the offline dataset and are hence unknown. Due to the penalty term, the agent is incentivized to plan ahead and stay within the confines of the offline dataset.

Another approach for restricting the agent from taking out-of-distribution actions is to explicitly learn the state-action distribution and only allow the agent to sample actions in the support of this distribution. We also incorporate this approach by learning a Conditional Variational Auto-Encoder (CVAE) to model the latent space of actions conditioned on the state, and allowing the low-level policy to only pick an action from this latent space.

4.1.1 Standard Setting

First, we consider the standard offline reinforcement learning setting, where a dataset $\mathcal{D}$ of transitions is given as described above and the goal is to use solely this dataset to learn and perform optimally in the corresponding online environment during deployment.

Given an offline dataset $\mathcal{D}$ as described above, the first step is to construct a pessimistic MDP (P-MDP) that is a pessimistic approximate model of the real MDP. This is done by learning an ensemble $\{T_k, R_k\}_{k=1}^{K}$ of dynamics models and reward functions. Each model T_k takes as input a state-action pair and predicts the change in the environment state due to the action, while R_k predicts the reward. The termination condition of the MDP is assumed to be known.

The T_k are trained on $\mathcal{D}$ until convergence, and a discrepancy measure $M : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is considered that acts as a proxy to the epistemic uncertainty of the learned dynamics models. It takes a state-action pair as input and outputs the disagreement regarding the resultant next state among the dynamics models in the ensemble. For our experiments, we use the variance

of the predictions as the discrepancy measure.

The P-MDP model is defined as follows. For sampling an episode from the P-MDP, one dynamics model T_k and reward model R_k are randomly chosen from among the ensemble. At each step, for current state s and a given action a, the next state is calculated as $s' = s + T_k(\bar{s}, a)$ and the discrepancy is calculated as $M(s, a) = \mathsf{Var}\left(\{T_j(\bar{s}, a)\}_{j=1}^{K}\right)$, where $\bar{s}$ is the normalized state. The reward is calculated as $R_k(\bar{s}, a)$.

If $M(s, a)$ is above a certain threshold, the episode is terminated and a penalty is given to the agent. The threshold and penalty are hyperparameters that depend on the environment under consideration. This termination and penalty induce pessimism and deter the agent from wandering far from the support of the given offline dataset distribution.

The second step is to train a Conditional Variational Auto-Encoder (CVAE), consisting of an encoder E and a decoder D. Each state-action pair (s, a) in $\mathcal{D}$ is passed through the encoder to obtain the encoding $\bar{a} = E(\bar{s}, a)$. For a given state, an action can be sampled in $\mathcal{A}$ by sampling an action $\bar{a} \sim \mathcal{N}(0, I)$ in the latent space and passing it through the decoder, giving $a = D(\bar{s}, \bar{a})$.

The CVAE is trained in the standard way, with the loss to be minimized being the sum of the reconstruction error and the KL divergence between the latent actions and the standard normal distribution.

Now, an arbitrary hierarchical RL algorithm H can be trained using a combination of the CVAE and the P-MDP as follows. When at state s, the low-level policy takes an action $\bar{a}$ in the state-conditioned latent action space of the CVAE. This is passed through the decoder to obtain the real action $a = D(s, \bar{a})$, which is then passed to the learned P-MDP. The P-MDP returns the next state and reward to H, which uses the transition as it sees fit.

In essence, the algorithm H is operating in the approximate pessimistic MDP constructed from the offline dataset and is planning in the latent action space of the CVAE. It treats this setting as if it is an online environment and plans and learns as in the online setting without any modifications. The overall structure is given in Figure 4.1.

4.1.2 Transfer setting

An advantage of learning reusable low-level skills in a hierarchical agent is the transfer of such skills to solve a task different from the one the agent was trained on. If the low-level skills are reusable, then retraining the hierarchical agent on the new task will be more sample-efficient than retraining a flat agent.

In our work, we consider the setting where an offline dataset is given for a specific task in an environment, but the agent has online access to the environment and needs to learn a different

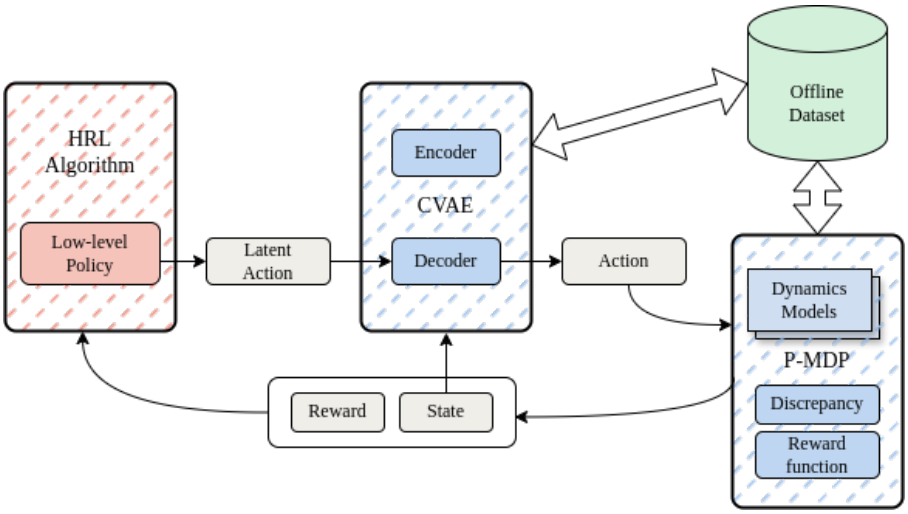


Figure 4.1: Meta-algorithm for learning an offline version of a hierarchical online algorithm. The CVAE and environment models are learned from the offline dataset. These are then used to train the HRL algorithm, which operates in a pessimistic approximation of the actual environment and the latent action space of the CVAE.

task. We first train the hierarchical agent offline as described previously and then fine-tune it online on the new task.

The actions to be taken to solve the new task need not be the same actions that solve the original task. Therefore, while the agent learns to plan in the latent action space during the offline training, for online fine-tuning, after that, the decoder is discarded, and the agent is made to plan using the actual actions in the environment.

4.1.3 Goal-Conditioned Setting

In many problem settings, it is desirable to learn a general agent that can reach any goal given to it, rather than retrain a different agent for each task that needs to be done. This has led to the study of algorithms that learn goal-conditioned policies that take the current state and intended goal as input and output an action based on that.

In a hierarchical setting, the final intended goal is given to the high-level policy, which in turn creates a low-level sub-goal and passes it to the low-level policy to reach. The transitions in the offline dataset also contain the goal information.

For dealing with such a scenario using our algorithm, we learn a goal-conditioned CVAE, in which both the encoder and decoder are passed the goal along with the state as the conditioning input to learn the latent action space. The P-MDP model is left unchanged.

4.2 Experiments

4.2.1 Standard Setting

For the standard setting, we consider the OpenAI Gym MuJoCo (Brockman et al., 2016) continuous control environments, HalfCheetah-v2, and Hopper-v2. The agent controls a 2D figure, and the task is to make the figure move in the forward direction while remaining upright. We consider the "Medium" and "Medium-Expert" datasets in the D4RL benchmark (Fu et al., 2020) as the offline datasets on which our algorithm and baselines are run.

For the base online hierarchical algorithm, we use the Multi-updates Option Critic (MOC) algorithm (Klissarov and Precup, 2021). Here, a stochastic high-level policy chooses the option to execute, and the options are parameterized by neural networks with shared initial and hidden layers.

For the baselines, we consider Proximal Policy Optimization (PPO) (Schulman et al., 2017) and Soft Actor-Critic (SAC) (Haarnoja et al., 2018) algorithms run on top of our CVAE + P-MDP framework. This gives an insight into whether this framework preserves the advantages of using a hierarchical agent over a flat agent. The results are shown in Figure 4.2. The plots in all the figures show the results averaged over five runs, with the shaded region denoting the standard deviation. Further details of training are provided in Section 4.2.6.

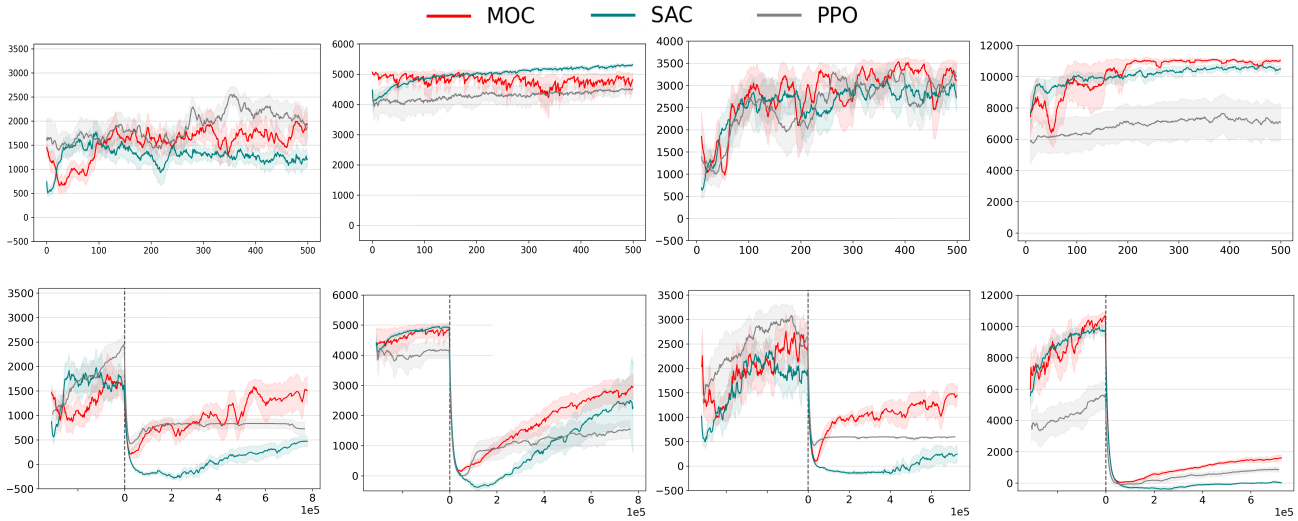
Table 4.1 gives the results of our framework compared with other offline algorithms in the literature, which are Model-Based Offline Reinforcement Learning (MOReL) (Kidambi et al., 2020), Model-Based Offline Policy Optimization (MOPO) (Yu et al., 2020), and Conservative Q-Learning (CQL) (Kumar et al., 2020).

The performance of our algorithm is comparable to that of existing algorithms except on Hopper Medium. It outperforms the other methods in the HalfCheetah-v2 environment, and in the case of the HalfCheetah Medium-Expert dataset, where MOReL attains a maximum unnormalized reward of around 8000, it can be seen that the additional CVAE results in our method obtaining rewards more than $10,000$.

4.2.2 Transfer Setting

For the task transfer setting, we consider the same MuJoCo locomotion environments as before. The offline dataset is the same as before, but during the online fine-tuning, the task is changed to a new one wherein the agent is rewarded for going in the backward direction, as opposed to the forward direction in the original task.

In the standard setting, we see that the hierarchical agent using MOC is performing as well as SAC and even PPO in some cases. However, when the task is switched and the trained agents



(a) Hopper Medium (b) HalfCheetah Medium (c) Hopper Medium-Expert (d) HalfCheetah Medium-Expert

Figure 4.2: First row: Rewards obtained in Gym MuJoCo locomotion environments in the standard offline reinforcement learning setting w.r.t. the number of iterations of training. The captions specify the environment and the offline dataset on which the algorithms are learned. All three algorithms are trained on the CVAE + P-MDP framework. Second row: Results for the transfer task in Gym MuJoCo environments. The dip in the reward corresponds to the start of online training on the different tasks, and the x-axis corresponds to the number of samples obtained online.

are updated to learn a new task online, we see in Figure 4.2 that MOC clearly outperforms the baselines. This shows that learning a hierarchy results in better sample complexity when the agent is asked to transfer its skills from one task to another. This also means that our framework preserves the advantages of a hierarchical agent over flat agents in the offline setting.

4.2.3 Goal-Conditioned Setting

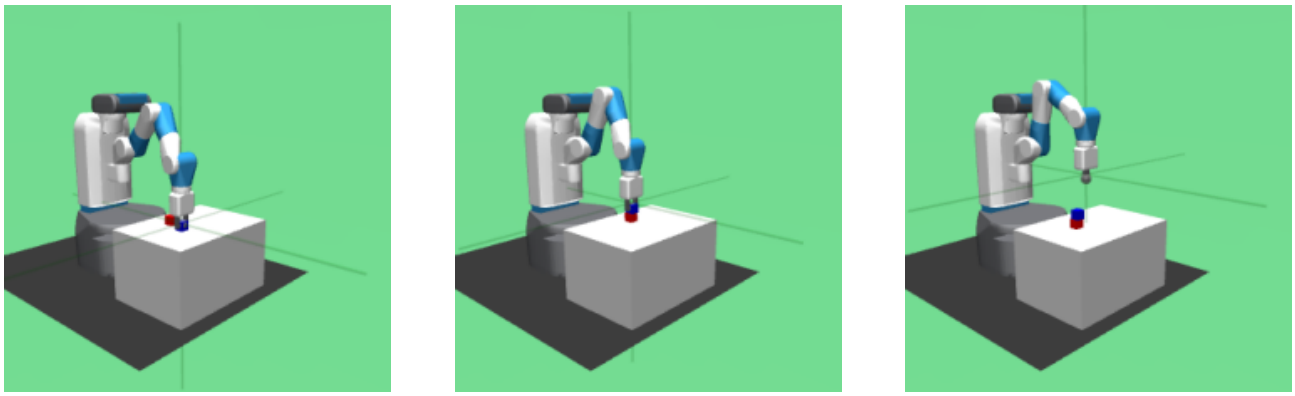
For the goal-conditioned setting, we consider two block-stacking tasks wherein the agent controls a robotic gripper and has to handle and manipulate colored block-shaped objects based on the goal given.

In the first task, the agent has to learn to achieve the following three goals: (i) grasping the blue block, (ii) placing it on the red block, and (iii) returning the gripper to the original position after successfully completing the previous tasks. It is given a 0 reward when the goal is reached, -1 otherwise. The observations given to the agent consist of information about the positions and orientations of two blocks, and the goals are in the form of one-hot vectors.

Similarly, the second task involves three blocks and six goals, with goals (i)-(iii) involving

Table 4.1: Normalized scores on D4RL benchmark. Our algorithm scores are the running averages over five seeds. Results of MOReL, MOPO, and CQL are reported from their respective papers.

Environment	Dataset	Offline MOC (Ours)	MOReL	MOPO	CQL
Hopper	Medium	76.8	**95.4**	28.0	86.6
Hopper	Medium-Expert	107.4	108.7	23.7	**111**
HalfCheetah	Medium	**45.04**	42.1	42.3	44.4
HalfCheetah	Medium-Expert	**90.6**	53.3	63.3	62.4



(a) Goal (i) (b) Goal (ii) (c) Goal (iii)

Figure 4.3: The three high-level goals in the robotics environment: (i) grasping the blue block, (ii) placing it on the red block, and (iii) returning to the end position. These actions have to be taken sequentially, necessitating a hierarchical agent with a high-level planner.

setting the blue block on the red block and goals (iv)-(vi) for setting the green block on the red block.

For these tasks, we consider the Universal Options Framework (UOF) algorithm that learns a goal-conditioned high-level policy and a goal-conditioned low-level policy. The output of the high-level policy is the low-level goal given to the low-level policy. The same sparse $\{-1, 0\}$ reward is given to the low-level policy as the low-level reward, depending on whether it reached the low-level goal.

A fixed number of low-level goals are defined *a priori* that correspond to the high-level goals and are a function of the current state. The high-level policy chooses one of these as its high-level action and passes it to the low-level policy.

The high-level policy is trained using a deep-learning based goal-conditioned version of the Intra-Option Learning algorithm called DIOL. It is also provided with abstract demonstrations,

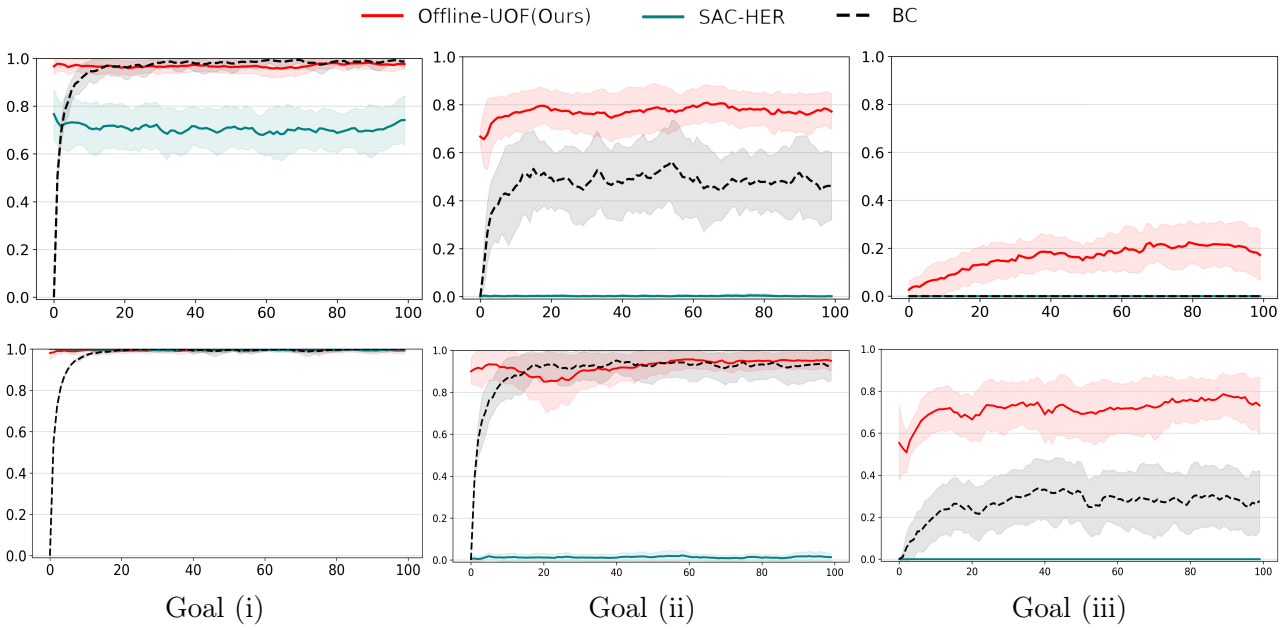


Figure 4.4: Results on the first robotic-gripper block-stacking task for UOF along with SAC-HER and BC baselines, trained on the CVAE + P-MDP framework. The first and second rows depict the performance when trained on the Medium and Medium-Expert datasets, respectively. Each plot shows the fraction of times the agents were able to reach the corresponding goal against the number of training iterations.

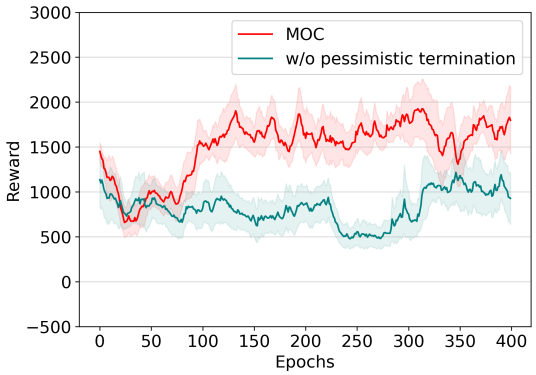
which are sequences of high-level actions to take to achieve the desired high-level goal. The low-level policy is learned using Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2016) and Hindsight Experience Replay (HER) (Andrychowicz et al., 2017) to deal with sparse low-level rewards.

We ran the UOF algorithm in the online environment and used two partially trained models as the behavior policies to create the offline datasets “Medium" and “Medium-Expert" on which to benchmark the offline algorithms. As baselines, we consider the SAC algorithm with HER and behavior cloning as a function of the current state and low-level goal. All the algorithms use the low-level goals in the offline dataset to learn goal-conditioned policies in the latent space of the goal-conditioned CVAE.

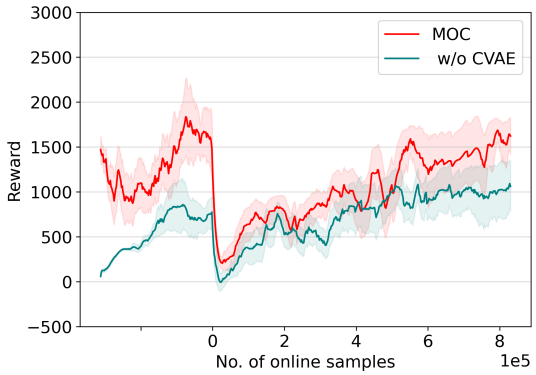
For evaluation, each agent is given a goal to reach. For UOF, the goal is in the form of a one-hot vector. For the baselines, the goal is in the same format as the low-level goals of UOF. Figure 4.4 and Table 4.2 show the results of the experiments on the Medium and Medium-Expert datasets, respectively, as a fraction of evaluation episodes during which the agent could reach the goal, plotted against the number of training epochs. It is clear that while all agents can reach the initial goals, the baselines struggle to reach the later goals due to a

Table 4.2: Performance on the second robotic block stacking task involving six goals. It can be seen that while the baselines have some success in the first of a sequence of tasks, i.e., tasks (i) and (iv), only our algorithm successfully learns tasks (ii, iii), and (v, vi) that occur later in their respective sequence.

Algorithm	Goal (i)	(ii)	(iii)	(iv)	(v)	(vi)
Our algorithm	0.90	**0.71**	**0.41**	0.93	**0.81**	**0.46**
SAC+HER	0.33	0.03	0	0.20	0.01	0
BC	**0.99**	0.50	0	**0.99**	0.20	0



(a) Ablation of pessimistic termination on Hopper Medium.



(b) Ablation of CVAE for Hopper task switch.

Figure 4.5: Results for the ablation of pessimistic termination and CVAE, respectively, in the Hopper environment with the Medium offline dataset.

lack of planning. This demonstrates that a high-level policy guiding a low-level policy to the desired goal performs better than a single goal-conditioned policy.

4.2.4 Ablations

To analyze the effectiveness of each component of our framework, we perform ablation studies in the goal-conditioned setting. We conduct experiments to validate the necessity and effectiveness of the CVAE, the goal-conditioning of the CVAE, and the discrepancy-based termination of the P-MDP. The results are shown in Figures 4.5 and 4.6.

The necessity and effectiveness of the CVAE is a valid question since the MOReL algorithm does not need a CVAE to train an agent with a P-MDP. There might be an additional concern that limiting the actions that can be taken by the agent to the distribution learned by the CVAE might hinder generalization.

However, we find that it is not the case. Figure 4.6 shows the results of the UOF algorithm

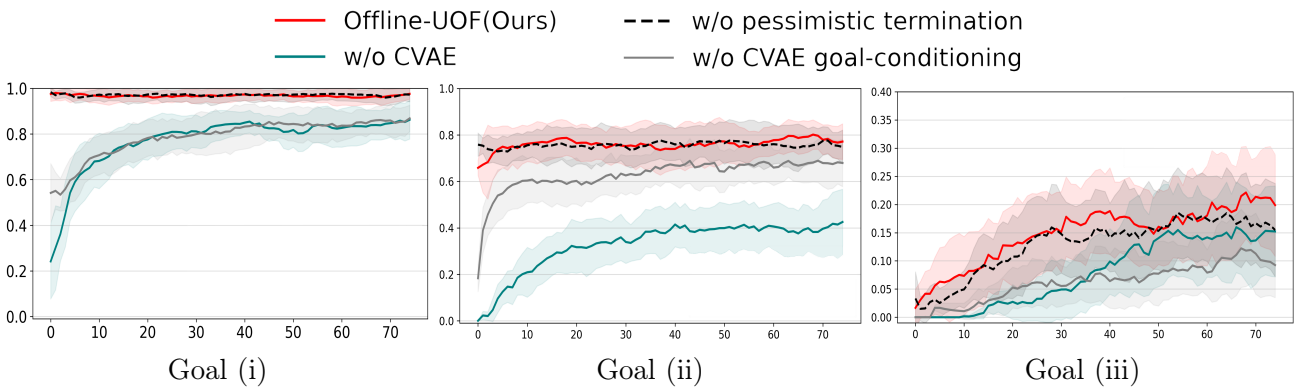


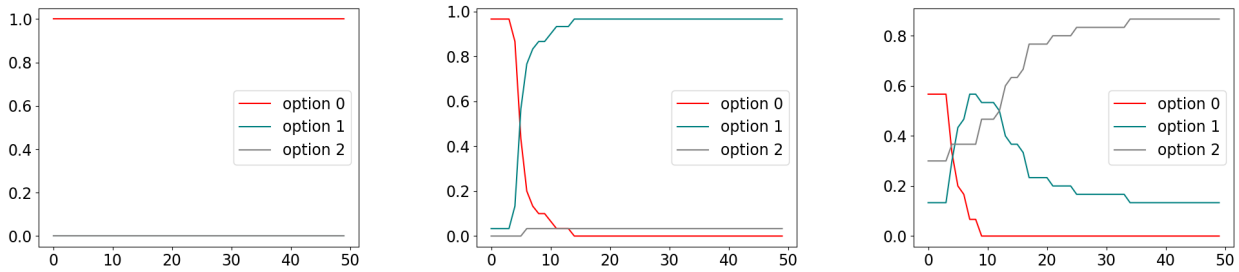
Figure 4.6: Results for the ablation experiments in the robotic gripper block-stacking tasks, showing the fraction of times the agent reached the specified goal versus the number of training iterations.

on the offline dataset with and without the CVAE. It is clear that the CVAE not only enables learning a better policy, but it enables faster convergence as well, since the decoder acts as an initial pre-trained policy from which the agent can start and improve. The same can be observed in the transfer setting as well, as seen in Figure 4.5b, which shows the performance in the Hopper transfer task after training offline on the Hopper Medium dataset.

Another aspect of the algorithm in the goal-conditioned setting is the goal-conditioning of the CVAE itself. The same latent action taken by the low-level policy results in different actual actions executed in the environment based on the desired goal of the low-level policy and the current state.

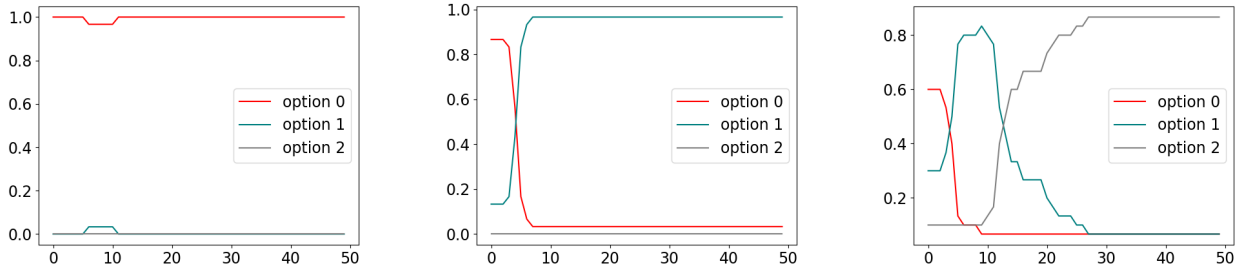
To study the value of this approach, we conducted an ablation to compare the performance with that of an agent in which the CVAE is conditioned only on the state. It can be seen from Figure 4.6 that removing the goal-conditioning of the CVAE while retaining goal-conditioned high and low-level policies still results in a worse agent.

Additionally, we also conducted experiments without the pessimistic termination of the MDP when an uncertain state is encountered. We see that this component does not affect the performance significantly, possibly because all the episodes are restricted to only 25 time steps for this task, and so early termination does not significantly affect the behavior of the learned policies. In environments with longer trajectories, pessimistic termination is an important component in the offline setting, as seen in Figure 4.5a, which shows the results of this ablation in the Hopper-v2 environment trained on the Hopper Medium dataset.



(a) Goal (i) (b) Goal (ii) (c) Goal (iii)

Figure 4.7: Fraction of times each high-level action is chosen by the trained UOF agent at each time step after being trained using the Medium dataset.



(a) Goal (i) (b) Goal (ii) (c) Goal (iii)

Figure 4.8: Fraction of times each high-level action is chosen by the trained UOF agent at each time step after being trained using the Medium-Expert dataset.

4.2.5 Options learned

In the first block stacking task, due to the way the goals are defined, reaching a goal requires reaching all the previous goals also in order. The three possible high-level actions of the agent are defined corresponding to the three goals of the environment. So if the options are learned properly as intended, given a high-level goal, the agent should choose each of the previous high-level actions until the corresponding high-level goal is reached, and move on to the next high-level goal, and so on. For example, for reaching goal (iii), the agent should choose high-level action (i) to reach goal (i), then choose action (ii) to reach (ii), and then choose action (iii) to reach the intended goal.

It turns out that this is actually the case for the agent learned using UOF on the offline dataset. This can be seen in Figures 4.7 and 4.8, which show the fraction of times a certain high-level action is chosen by the agent at a particular time step, when trained on the Medium

and Medium-Expert datasets respectively till convergence and evaluated 30 times. The x-axis shows the time passed, and the y-axis shows the fraction of times each option is active at that time step.

In Figure 4.7, for goal (iii), it can be seen that option 0 is chosen initially. Then the likelihood of choosing option 1 peaks, followed by option 2, to finally reach the goal. This order of choosing the options becomes even more evident and consistent in Figure 4.8 since the training has been done on a better dataset.

4.2.6 Details of Experiments

Robotics Environment

We consider the first two block-stacking tasks described in Yang et al. (2021), wherein the agent controls a robotic gripper to manipulate given colored blocks. The first task consists of three goals to (i) catch the blue block, (ii) set it on the red block, and (iii) return to the initial position. Similarly, the second task consists of six goals corresponding to placing a blue block or a green block on the red block.

For the hierarchical algorithm, we have two levels of goals. Each of the three goals is given as a high-level goal to the high-level policy. Additionally, the high-level policy has three high-level actions or options it can take, corresponding to the three goals. Each of these three high-level actions is mapped to a corresponding low-level goal by a known function based on the current state. This low-level goal is represented by the absolute Cartesian coordinates of the block and the gripper, along with the gripper width. For completing the n'th higher-level goal, the agent must complete the previous $n-1$ goals in proper sequence. So the high-level policy needs to learn to give these low-level goals to the low-level policy appropriately.

As opposed to the D4RL dataset, each sample transition in the offline dataset for the robotics environment also consists of a low-level goal. Thus, each sample in the static dataset $\mathcal{D}$ can be represented as $(s_i, a_i, g_i, s_{i+1}, r_i, d_i)$. We create two offline datasets of $1M$ samples each for our experiments by running the UOF algorithm online for 20 and 50 epochs, respectively, and collecting samples using these partially trained models to interact with the environment. We refer to these two offline datasets as the "Medium" and "Medium-Expert" datasets, respectively.

The high-level policy in UOF takes the one-hot high-level goal as its input, while the low-level policy can take any low-level goal as input in the format described earlier. For the Behavior-Cloning (BC) and the SAC-HER agents, there exists no hierarchy. So, the low-level goals in the dataset from the UOF algorithm are used as the goals for the baseline algorithms.

Dynamics Model The dynamics model for the robotics environment consists of just a transition model. The reward function is not learned but defined to be 0 when the goal is reached and -1 otherwise. The transition model comprises a Multi-Layer Perceptron (MLP) with two hidden layers of 512 ReLU activated nodes each. We use an ensemble of 5 models to compute the discrepancy in the next state prediction. The states are normalized, and the models predict the residual for the next state. Since normalized states are used, the dynamics model is learned using the Adam optimizer to minimize the l_1 loss.

CVAE architecture The proposed CVAE consists of an encoder and a decoder network, each of which comprises 2 hidden layers with 750 ReLU activated nodes. The low-level goal is passed along with the state as input to the CVAE components. The latent dimension is the same as that of the action dimension. Similar to the dynamics model, we use Adam optimizer and l_1 Loss. Once the CVAE is learned, the encoder is discarded, and just the decoder part of the CVAE is used to convert the latent actions of the agent to the real actions to be taken in the P-MDP or the environment during training and evaluation of the RL agent, respectively.

Hyperparameters For training the dynamics models and the CVAE, we use a train-validation split of 85 : 15. The learning rate and the number of hidden neurons in the MLP are decided by a grid-search technique on the basis of the respective validation losses. The search is done using Ray Tune (Liaw et al., 2018) to randomly select and evaluate 10 out of all possible combinations. The ranges on which the hyperparameters are searched are given in Table 4.3.

Table 4.3: Grid search range for hyperparameters of the transition models and CVAE networks.

Hyperparameter	Search Range
Learning rate	$\{10^{-3}, 10^{-4}, 10^{-5}, 2 \times 10^{-4}, 5 \times 10^{-4}\}$
Transition model hidden layer size	$\{128, 256, 512, 1024\}$
CVAE hidden layer size	$\{256, 512, 720, 1024\}$

At any state, when a specific action is taken, a discrepancy is calculated as the variance of the next state prediction by the transition functions of the model ensemble. As described previously, when the discrepancy at a particular state-action pair exceeds a certain threshold, the episode is terminated with a high negative reward.

The threshold can be defined in one of two ways. Once the ensemble is trained and fixed, the discrepancy value is calculated for each state-action pair in the offline dataset. Using these values, the threshold can be calculated as either a quantile of these values or a fraction of the maximum discrepancy value on the dataset.

Table 4.4: Hyperparameters for dynamics model and CVAE in the robotics environment.

Hyperparameter	Value
Dynamics Model	MLP$(512, 512)$
Activation	ReLU
Batch size	256
Learning rate	10^{-4}
Discrepancy quantile	99
Negative reward	50
No. of models in ensemble	5
CVAE	MLP$(720, 720)$
Activation	ReLU
Batch size	128
Learning rate	10^{-4}

Which of the above two methods to use, and the corresponding quantile or fraction value, are both hyperparameters, as well as the reward penalty given when the discrepancy exceeds this threshold and the episode is terminated.

The discrepancy threshold is set by evaluating the UOF agent in the original environment using the CVAE. For this environment, we set the threshold as a quantile of the discrepancy values induced by the ensemble on the offline dataset. The final hyperparameters are given in Table 4.4. The hyperparameters for SAC-HER are the default hyperparameters in Stable Baselines 3.

The SAC-HER algorithm is taken from Stable-Baselines3 (Hill et al., 2018), and the UOF code has been adopted from its official implementation (Yang et al., 2021).

Locomotion Environments

For the locomotion environments, we consider two MuJoCo continuous control tasks, Hopper-v2 and HalfCheetah-v2, in OpenAI Gym (Brockman et al., 2016). The agent controls a 2D figure, and the task is to make the figure move in the forward direction while remaining upright. We consider the "Medium" and "Medium-Expert" datasets in the D4RL benchmark (Fu et al., 2020) as the offline datasets on which our algorithm and baselines are run. The Medium dataset comprises 1M transitions and the Medium-Expert dataset comprises 2M. After transfer, the Gym environment is modified such that the environment gives rewards when the agent moves in the backward direction.

Dynamics Model Our dynamics model consists of a transition model as well as a reward model, each of which comprises an MLP with two hidden layers of 512 ReLU-activated nodes. We use an ensemble of 5 models to compute the discrepancy in the next state prediction. The dynamics model and the reward model are both learned using the Adam optimizer and the l_1 Loss.

Table 4.5: Grid search range for hyperparameters of the transition models and CVAE networks.

Hyperparameter	Search Range
Learning rate	$\{10^{-3}, 10^{-4}, 10^{-5}, 2 \times 10^{-4}, 5 \times 10^{-4}\}$
Transition model hidden layer size	$\{128, 256, 512, 1024\}$
Reward model hidden layer size	$\{128, 256, 512, 1024\}$
CVAE hidden layer size	$\{256, 512, 720, 1024\}$

CVAE The CVAE consists of an encoder and a decoder, each of which comprises an MLP with two layers of 750 ReLU-activated nodes. The latent dimension is the same as the action dimension of the environment under consideration. We use Adam optimizer with l_1 loss to learn the parameters.

Hyperparameters For training the dynamics models, reward model, and CVAE, we use a train-validation split of 90 : 10. As in the robotics environment, the learning rate and the number of hidden neurons in the networks are decided using grid search based on the validation loss. The range in which the hyperparameters are searched is given in Table 4.5.

Once the CVAE and the dynamics and reward functions are learned, the discrepancy fraction or quantile of the ensemble, along with the negative reward penalty, are tuned by training an SAC policy on the CVAE + P-MDP and evaluating it using the CVAE and the real gym environment, where the actions taken are obtained by passing latent actions through the CVAE decoder. The hyperparameters thus obtained are given in Tables 4.6 and 4.7.

After fixing the discrepancy threshold and penalty, the hyperparameters of the MOC and PPO algorithms are set by training on the CVAE + P-MDP model and evaluating using the CVAE and the real gym environment. The hyperparameters used are given in Table 4.8.

The learning rate and η in MOC are tuned using Ray-Tune. Other hyperparameters are taken from the MOC paper (Klissarov and Precup, 2021). Similarly, for PPO, all the hyperparameters are tuned except for vf_coef, ent_coef and PPO steps, which are taken from Hugging Face. PPO and SAC have been used from Stable-Baselines3 (Hill et al., 2018), and the MOC code has been adopted from its official implementation (Klissarov and Precup, 2021).

Table 4.6: CVAE and P-MDP hyperparameters for Hopper

Parameter	Dataset	
	Medium	Medium-Expert
Dynamics model	MLP (512,512)	MLP (512,512)
Activation	ReLU	ReLU
Batch size	256	256
Learning rate	10^{-4}	10^{-4}
Discrepancy fraction	1.08	0.08
Negative reward penalty	20	30
Number of models in ensemble	5	5
CVAE	MLP(720,720)	MLP(720,720)
Activation	ReLU	ReLU
Batch size	256	128
Learning rate	10^{-4}	10^{-4}

4.3 Related Work

The concept of options to incorporate temporal abstraction into reinforcement learning agents was introduced by Sutton et al. (1999). They studied this setting in the framework of Semi-Markov Decision Processes (SMDPs), in which each option is considered as an extended action that takes a variable number of time steps to execute. The Option-Critic (OC) architecture (Bacon et al., 2017) is a method to discover and learn options that are parameterized by deep neural networks and learned using policy gradient updates.

Over the years, many variants of the Option-Critic have been studied that aim to improve certain aspects of the algorithm. Interest Option Critic (Khetarpal et al., 2020) generalizes initiation sets to learnable interest functions. Residual Soft Option Critic (Zhu et al., 2021) extend OC by incorporating rewards based on the mutual information between different options and an entropy term. Smith et al. (2018) consider the options that gave rise to a trajectory as latent variables and updates all the options to maximize the reward.

The use of a P-MDP for offline learning was introduced by Kidambi et al. (2020), who used a model-based Natural Policy Gradient (NPG) approach to train the agent in the P-MDP. Such ideas regarding pessimism or conservatism when dealing with out-of-distribution states or actions are incorporated by many offline algorithms in different ways. Model-based Offline Policy Optimization (MOPO) (Yu et al., 2020) also trains an arbitrary RL algorithm in an approximate MDP, penalizing the agent based on the aleatoric uncertainty of the current model

Table 4.7: CVAE and P-MDP hyperparameters for HalfCheetah

Parameter	Dataset	
	Medium	Medium-Expert
Dynamics model	MLP (512,512)	MLP (512,512)
Activation	ReLU	ReLU
Batch size	256	256
Learning rate	10^{-4}	10^{-4}
Discrepancy fraction	0.1	0.0101
Negative reward penalty	20	20
Number of models in ensemble	5	5
CVAE	MLP(720,720)	MLP(720,720)
Activation	ReLU	ReLU
Batch size	256	128
Learning rate	10^{-4}	10^{-4}

at every time step. Lu et al. (2022) empirically compares different choices of uncertainty penalization along with other design choices in model-based offline RL. Model-Based Offline Options (MO2) (Salter et al., 2022) learns an option-transition model that predicts termination state distribution from a state-option-action tuple to optimize behavior cloning and predictability objectives.

The use of a CVAE trained on the offline dataset to implicitly constrain possible actions was introduced by Fujimoto et al. (2019) as a component of the Batch Constrained deep Q-learning (BCQ) algorithm. However, in BCQ, the actions output by the CVAE decoder are perturbed by a separate perturbation model learned using DDPG, and the actions are selected in a model-free manner by sampling multiple actions and choosing the best based on the Q-values. The Policy in Latent Space (PLAS) (Zhou et al., 2021) algorithm is another model-free algorithm that trains an agent in the latent space of a CVAE and uses an optional perturbation layer after the decoder for out-of-distribution generalization. Nasiriany et al. (2019) and Li et al. (2022) employ a VAE and a CVAE, respectively, to learn latent represenTransitioning from simple simulations to complex practical applications requires one to solve a host of new challenges, including a costly collection of data and the incorporation of high-level reasoning. Ajay et al. (2021) learn a continuous space of low-level skills by treating a skill as a latent variable corresponding to a sub-trajectory in the offline dataset and the low-level policy as a function of the state and the skill.

Table 4.8: Hyperparameters for MOC and PPO

Hyperparameter	Hopper-v2	HalfCheetah-v2
MOC		
lr	$10^{-4}, 10^{-4}, 10^{-4}$	$10^{-4}, 10^{-4}, 10^{-4}$
clipping value	0.2	0.2
η	0.7	0.7
γ	0.99	0.99
λ	0.95	0.95
PPO		
steps	2048	2048
lr	9.80828×10^{-05}	0.0003
clipping value	0.1	0.2
vf_coef	0.835671	0.835671
ent_coef	0.00229519	0.00229519

4.4 Outlook

In recent times, Reinforcement Learning has increasingly been used to learn complex tasks for developing agents deployable in the real world. Transitioning from simple simulations to complex practical applications requires one to solve a host of new challenges, including costly collection of data and incorporation of high-level reasoning.

The problem of a lack of access to online samples from the environment is mitigated by offline algorithms designed to work on datasets that have been collected previously by an unknown policy. High-level complex reasoning is achieved by the use of hierarchical policies that incorporate temporal abstraction, operating at different time scales across various levels of the hierarchy. However, learning such policies is highly sample-intensive, and there is limited research on a broad approach to incorporate hierarchy into an agent using just an offline dataset.

We solve this problem by proposing a general approach to convert any online hierarchical algorithm to operate solely on an offline dataset. Our approach consists of constructing a new MDP that is a pessimistic approximation of the real environment, and whose action space is the state-conditional latent action space of the behavior policy learned from the offline dataset using a CVAE.

We demonstrated the validity of our method on various continuous control tasks. First, we built upon MOC (Klissarov and Precup, 2021), an online algorithm that learns a two-level hierarchy in the form of options, to tackle MuJoCo locomotion tasks. We showed that

it outperforms or is comparable to other state-of-the-art flat offline algorithms. Further, we showed that the hierarchy learned is valid and preserves the advantages of learning the hierarchy, just as in an online setting, by applying the algorithm in a transfer task and demonstrating that a hierarchical policy learned using our framework learns to solve a target task online faster and better than a flat agent.

Next, we incorporated our method into the Universal Options Framework (UOF) (Yang et al., 2021) to learn to perform goal-conditioned tasks in a robotic gripper-based block-stacking environment. We showed that this outperforms flat goal-based algorithms such as Hindsight Experience Replay and Behavior Cloning. We performed further analysis in the form of ablation studies to demonstrate the importance of each component of our algorithm.

While there are algorithms that can deal with complex tasks using hierarchical or goal-conditioned policies in the offline setting, our algorithm differs in its generality. For any environment or task, along with desired behavior for an agent, e.g., interpretability or generalization, a suitable existing online algorithm known to work for a similar setting can be chosen and adapted using our approach to solve the task offline.

Chapter 5

Active Offline Reinforcement Learning

Real-world applications of machine learning often involve sequential decision-making problems, wherein the agent has to not only react to the current state of the environment but also plan for the future on how its actions affect the evolution of the environment. Training such an agent requires extensive data on how its actions affect the environment in diverse regions of the state space of the environment.

Strategies for collecting new data in such sequential settings are particularly complex due to the multitude of factors that have to be taken into consideration by the exploration agent. Choices that have to be made during exploration include where to collect new data, which path to follow, and how much to collect.

In many real-world cases, there may exist constraints on the regions of the environment from which an agent can explore and gain additional experience. For instance, consider the problem of determining the optimal treatment regimen in clinical trials. We have access to existing data from related trials and wish to collect additional data that is more relevant and informative by providing the treatment to new patients. However, each new possible patient has a different treatment and medical history and can be at a different stage of diagnosis. Using a limited budget, we have to choose the optimal subset of participants to provide the treatment so as to optimally complement existing data.

Given budget constraints, there is an additional dilemma of continuing along a long trajectory versus collecting multiple short trajectories at various regions. For example, consider the problem of collecting additional data to improve the navigation capability of self-driving vehicles. Data collection is redundant in regions where representative samples are already present in the dataset at hand, in which case any trajectory that enters such a region can be cut short, and a new trajectory can be started in a more informative location.

In the context of supervised learning, this problem of choosing how to collect more data

is studied as Active Learning. The agent has access to a small amount of labeled data and a huge dataset of unlabeled data that is expensive to annotate. The objective is to pick a small subset of unlabeled data to be labeled to maximize the performance of a model trained on this augmented labeled dataset.

This becomes more challenging in the case of sequential decision-making problems since the data is in the form of samples and trajectories that are unknown until an agent interacts with an environment via an exploration policy. So here, instead of choosing what points to label, the agent has to choose where, how, and how much to explore the environment.

Since collecting data is expensive in many cases, we wish to reuse existing related data previously collected by someone with some unknown exploration policy. Learning a policy using solely such an offline dataset is studied as the problem of offline reinforcement learning.

In our work, we wish to study strategies for further active exploration of the environment in the context of this offline dataset. We consider the case where a small exploration budget is given to the agent, and the data should be collected efficiently so as to complement the offline dataset and avoid redundant data. Details of our experiments and results can also be found in Dukkipati et al. (2025).

5.1 Problem Formulation

Consider an MDP defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \rho, \gamma)$, where $\mathcal{S}$ is the set of possible states of an environment, $\mathcal{A}$ is the continuous space of actions that can be taken by the agent, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta\mathcal{S}$ is the transition function, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, ρ is the initial state distribution and $\gamma \in [0, 1)$ is the discount factor.

Let $V^\pi(s)$ be the expected sum of discounted rewards obtained by an agent following policy $\pi : \mathcal{S} \to \Delta\mathcal{A}$ from starting state s. The goal of reinforcement learning is to find a policy that maximizes V^π. We are given an offline dataset $\mathcal{D}$ of the form

$$\mathcal{D} = \{(s_i, a_i, s'_i, r_i, d_i)\}_{1 \le i \le N} ,$$

where s'_i is a next state sample due to taking action a at state s_i, r_i is the resultant reward, and d_i denotes whether it resulted in episode termination.

First, we explain the problem in a supervised learning setting. Here, given an unlabeled set of data points $\mathcal{D}_{\text{Unl}}$ and a small set of labeled points $\mathcal{D}_{\text{lab}}$, the agent has to choose which points in $\mathcal{D}_{\text{Unl}}$ to label, in the context of what is already available as labeled data. More precisely, consider a supervised learning algorithm Alg that takes a labeled dataset $\mathcal{D}_{\text{lab}}$ and returns a trained model $\pi_{\mathsf{Alg}}(\mathcal{D}_{\text{lab}})$. Further, any model π has to finally perform well on some unknown

distribution, quantified by some performance measure V^{π}. At each stage, the goal of the active learning agent is to choose a point $\tau \in \mathcal{D}_{\text{Unl}}$ to label so as to optimize V of the model learned on this new dataset, i.e.,

$$\tau^* = \underset{\tau \in \mathcal{D}_{\text{Unl}}}{\operatorname{argmax}} V^{\pi_{\mathsf{Alg}}(\mathcal{D}_{\text{lab}} \cup \{\tau\})}.$$

The aim here is to pose this problem in an MDP setting. Due to the sequential nature of the problem, the unlabeled data points in supervised learning correspond to unknown trajectories. Further, a trajectory (which translates to a data point in supervised learning) is not available to be chosen here but can only be observed as a stochastic function of some exploration strategy employed by the active learning agent in the context of constraints imposed by the environment. We formalize these aspects below.

Consider the following MDP $\mathcal{M}_{\text{Act}} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \hat{\rho}, \gamma)$, with a different initial state distribution $\hat{\rho}$ for the active trajectory collection phase. At the start of each episode, a set of candidate initial states $\mathcal{C} = \{s_i \sim \hat{\rho} : 1 \leq i \leq N\}$ is sampled from $\hat{\rho}$. The agent can choose to start exploring the environment from any subset of these candidates and can also stop exploring at any time, if necessary, to preserve its budget.

Thus, the active learning agent $\mu = (\mathcal{I}, \pi, \beta)$ has three components: (i) an initial state selection function $\mathcal{I} : \mathcal{S} \to \mathbb{R}$ that decides the utility of collecting a trajectory from a given state, (ii) an exploration policy $\pi : \mathcal{S} \to \Delta\mathcal{A}$ that maps states to probability distributions over actions, and (iii) a termination function $\beta : \mathcal{S} \to \{0, 1\}$ that decides whether to terminate the current trajectory.

Such an agent μ induces a distribution $\mathfrak{T}^{\mu}$ over trajectories due to the stochasticity of the initial state distribution and the exploration policy. The objective of the agent is to collect new samples within a budget in the context of the current dataset $\mathcal{D}$, so as to maximize the performance in the original MDP $\mathcal{M}$ of the policy π_{Alg} trained on this augmented dataset using the offline algorithm Alg, i.e.,

$$\mu = \underset{\mu = (\mathcal{I}, \pi, \beta)}{\operatorname{argmax}} \mathsf{E}_{\tau \sim \mathfrak{T}^{\mu}} \left[V^{\pi_{\mathsf{Alg}}(\mathcal{D} \cup \{\tau\})} \right].$$

For example, for the medical trials problem discussed above, candidate initial states are generated by $\hat{\rho}$ corresponding to patients with varied medical histories till that point. The agent should choose a subset of these based on the state and existing data. Similarly, in the navigation setting, when a known state is reached while exploring the environment, the termination function β stops the current trajectory so as to prevent the collection of redundant samples. The procedure for active exploration is listed in Algorithm 8.

Algorithm 8 Active Offline Reinforcement Learning

Require: Offline Dataset $\mathcal{D}$
Offline RL algorithm Alg
Interaction budget B

Ensure: $T = \{\tau_t\}$: Trajectories actively collected from the MDP $\mathcal{M}_{\text{Act}}$

Initialize $T = \{\ \}$
while $B > 0$ **do**
  Learn policy $\mu = \mu^{\mathcal{D}} = (\mathcal{I}, \pi, \beta)$ based on $\mathcal{D} \cup T$
  // `Obtain candidate states from the environment`
  $\mathcal{C} \sim \hat{\rho}\,(\mathcal{M}_{\text{Act}})$
  // `Choose the best state`
  $s_{\text{init}} \leftarrow \underset{1 \le i \le |\mathcal{C}|}{\text{argmax}}\, \mathcal{I}(s_i)$
  // `Collect trajectories from` $s_{\texttt{init}}$
  $\tau = \phi$
  $s = s_{\text{init}}$
  while $B > 0$ and $\beta(s) = 0$ **do**
    $a \sim \pi(s)$
    $s' \sim \mathcal{T}(s, a)$
    $r = r(s, a)$
    $\tau \leftarrow \tau \cup \{(s, a, s', r)\}$
    $B \leftarrow B - 1$
    $s \leftarrow s'$
  end while
  $T \leftarrow T \cup \tau$
end while
$\pi_{\mathsf{Alg}} = \mathsf{Alg}\,(\mathcal{D} \cup T)$

5.2 Algorithm

A practical implementation of actively collecting trajectories in addition to offline data and learning an optimal agent consists of two components: (i) The base offline reinforcement learning algorithm Alg that learns a policy given a dataset of transitions, and (ii) The active collection strategy $\mu = (\mathcal{I}, \pi, \beta)$.

In this work, we consider existing offline algorithms for the first component. For the second active component, the goal is to collect transitions that are diverse and underrepresented in the given offline dataset. To solve this, we propose an epistemic uncertainty-based method, where we learn an ensemble of representation models for encoding states and state-action pairs and use them to estimate the uncertainty of the agent in state and state-action space.

5.2.1 Representation-based Uncertainty Estimation

We consider representation models of the form $\mathcal{E} = (\mathcal{E}^s, \mathcal{E}^a)$ to encode state and state-action representations, where $\mathcal{E}^s$ and $\mathcal{E}^a$ encode state and action information, respectively. $\mathcal{E}$ has the following modes of operation:

(a) Given a state s as input, the latent embedding is obtained by passing it through the state encoder, i.e. $\mathcal{E}(s) \equiv \mathcal{E}^s(s)$, and

(b) Given a state-action pair (s, a) as input, the latent embedding is obtained by adding the embeddings of the state and action obtained through their respective encoders, i.e. $\mathcal{E}(s, a) \equiv \mathcal{E}^s(s) + \mathcal{E}^a(a)$.

To obtain meaningful embeddings using this method that are consistent with the structure of the environment, we enforce the following two modeling objectives for aligning the latent representations learned by $\mathcal{E}$:

(i) Clustering: Similar states (or observations) must be clustered in the latent space, and

(ii) Transition Dynamics Modeling: The embedding of a state-action pair must align with the latent representation of the corresponding next state.

Consider a transition tuple $(s, a, s', r, d) \sim \mathcal{D}$. To satisfy the clustering objective, we use s as the anchor sample, s' as the positive sample, and any other observation s'' sampled from $\mathcal{D}$ is considered as the negative sample. Embedding vectors $\mathbf{v}$, $\mathbf{v}^+$, and $\mathbf{v}^-$ are obtained by passing s, s', and s'' respectively through the state encoder $\mathcal{E}^s$. Moreover, to satisfy the transition dynamics modeling objective, we enforce the encoding $\hat{\mathbf{v}}^+ = \mathcal{E}(s, a)$ to be close to $\mathbf{v}^+$.

We train an ensemble $\{\mathcal{E}_k\}_{k=1}^K$ of such models to maximize the following augmented noise-contrastive loss:

$$\mathcal{L} = \log(\sigma(\mathbf{v} \cdot \mathbf{v}^+)) + \log(1 - \sigma(\mathbf{v} \cdot \mathbf{v}^-)) - \lambda \left\| \hat{\mathbf{v}}^+ - \mathbf{v}^+ \right\|^2,$$

where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid function and λ is a hyperparameter.

We consider epistemic uncertainty for both initial state selection and exploration. A natural strategy to estimate the same is to use the ensemble $\{\mathcal{E}_k\}_{k=1}^K$ of state representation models and calculate the amount of dissimilarity among the latent representations of said models, i.e., disagreement, for a given state or state-action pair.

Let $\mathbf{v}_k$ denote the latent representation of a state or state-action pair in model $\mathcal{E}_k$. So, for a given state or state-action pair, we have K vectors $\{\mathbf{v}_k\}_{k=1}^K$. We construct a similarity matrix $\mathbb{S}$ as follows:

$$\mathbb{S}_{i,j} = \|\mathbf{v}_i - \mathbf{v}_j\|^2, \quad \text{for all } i, j \in \{1, 2, ..., K\}. \tag{5.1}$$

We use the value of the largest element in $\mathbb{S}$ as our proxy for epistemic uncertainty of the model with respect to the environment.

5.2.2 Active Offline Reinforcement Learning Algorithm

We wish to solve the problem of learning an optimal RL agent by effectively making use of a given offline dataset while minimally actively exploring the environment in the context of available data. To achieve this, we propose a two-phase algorithm.

In the first phase of our algorithm, we learn an ensemble of representation models and an RL agent using solely the offline dataset without any exploration. Each representation model is independently learned using the techniques described in Section 5.2.1. The RL agent is learned using a suitable offline reinforcement learning algorithm depending on the environment.

The second phase is that of active exploration, which consists of two components: (i) Initial state selection, and (ii) Trajectory collection from thereon.

Given candidate initial states $\mathcal{C} = \{s_i\}$ from $\hat{\rho}$, those initial states are chosen that result in maximum uncertainty based on the representation model ensemble $\{\mathcal{E}_k\}_{k=1}^K$, calculated as

$$\text{Uncertainty}(s_i) = \max_{k,k'} \|\mathcal{E}_k^s(s_i) - \mathcal{E}_{k'}^s(s_i)\|^2.$$

For trajectory collection starting at the chosen states s_k, we consider the current policy π. At the beginning of the active phase, this is just the policy learned on the offline dataset with

the appropriate offline RL algorithm. In later epochs, it is the latest policy under consideration. At each step, M actions are sampled from $\pi(.|s)$ for the current state s, resulting in M state-action pairs. The policy π could be deterministic, in which case a scaled isotropic Gaussian noise is added to $\pi(s)$ in order to sample multiple actions.

The uncertainty for each pair is calculated using (5.1), and the action resulting in the most uncertain state-action pair is chosen as the action to execute. This exploration strategy is continued until the uncertainty of the current state falls below a certain threshold or the episode terminates.

The degree of exploration is controlled by using an ϵ-greedy variant of the exploration policy that explores using the aforementioned environment-aware uncertainty-based procedure with probability ϵ, and simply follows the policy π otherwise. The overall algorithm is listed in Algorithm 9.

5.2.2.1 Restricted initial states

In certain environments, it might be infeasible to modify the initial state distribution. Even if the agent has access to some candidate initial states at informative regions of the state space, the environment might prevent us from starting directly at those states, instead only allowing the agent to start from the default initial state distribution. This prevents us from executing the first phase of our active learning framework described above.

To solve this problem, we propose a modified version of the Algorithm 9, in which the agent follows a two-stage policy during active collection. In the first stage, it starts from the default initial state of the environment and goes to the optimal candidate initial state, from which the actual exploration can be done in the second stage. We train two separate policies for these two stages.

For the first stage, we train a goal-based policy on the offline dataset. We then create a weighted undirected graph $\mathcal{G}$ from the offline dataset, with each node corresponding to a state and the weight of edge $\{s_1, s_2\}$ being $e^{-\|s_1 - s_2\|}$, and divide the nodes into clusters. The second stage policy is the uncertainty-based exploration policy described in Algorithm 9.

During active collection, the goal-based policy is first used to reach the cluster of states nearest to the identified optimal candidate state, i.e, the candidate state corresponding to the most uncertainty. From here, the agent begins uncertainty-based exploration as described in the previous subsection.

5.3 Experiments

Algorithm 9 Active Offline Reinforcement Learning

Require: Offline Dataset $\mathcal{D}$, Interaction budget `Budget`

```
Train π and {ℰ_k}_{k=1}^K                    // Train offline policy and representation models
while Budget > 0 do
    Buffer = φ
    while Buffer is not full and Budget > 0 do
        Done ← False
        Sample candidate states C = {s_i} ~ ρ̂
        U = (Uncertainty(s_i))_{i=1}^{|C|}    // Calculate uncertainties for candidate states
        idx = argmax_i U_i
        s = s_idx
        while Uncertainty(s) ≥ threshold and done is not True and Budget ≥ 0 do
            a ~ π(s)
            // Explore with probability ϵ
            if ϵ ≤ Uniform(0, 1) then
                Sample M actions as a_m ← a + N(0, I)
                idx = argmax_m Uncertainty(s, a_m)
                a ← a_idx
            end if
            s', r, done ~ env.step(s, a)
            Buffer ← Buffer ∪ {(s, a, s', r, done)}
            Budget ← Budget − 1
            s ← s'
        end while
    end while
    D ← D ∪ Buffer
    Update π and {ℰ_k}_{k=1}^K using augmented dataset D
end while
π_Alg = Alg(D ∪ T)
```

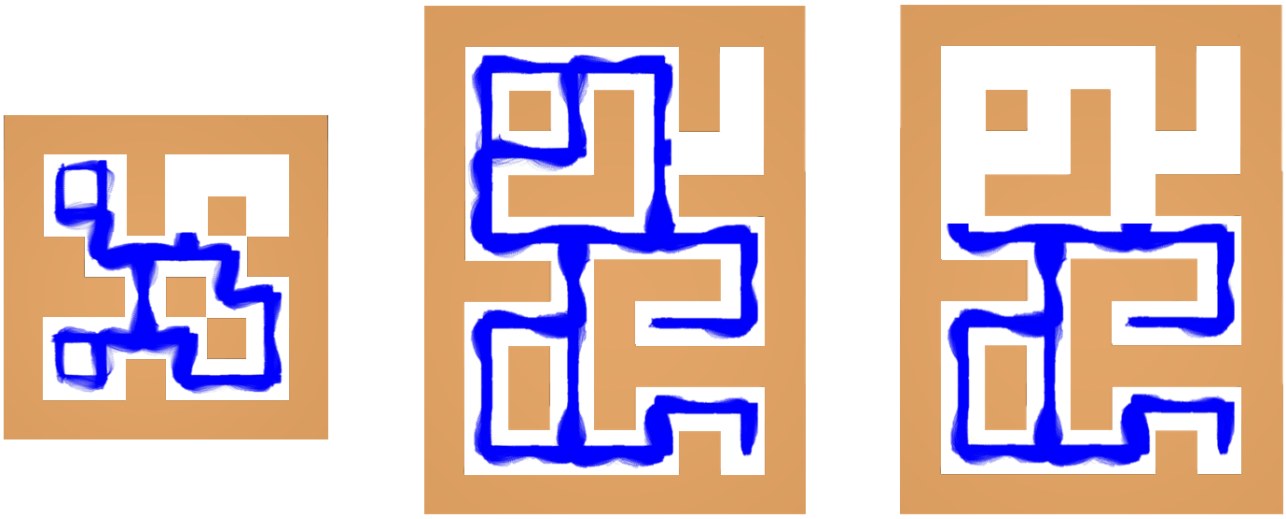


Figure 5.1: The pruned **Maze2d** D4RL datasets. The first image (on the left) corresponds to the `maze2d-medium-v1` environment. We prune the dataset near the goal state to create `maze2d-medium-easy-v1`. The other two images correspond to versions of the `maze2d-large-v1` environment, an easy version `maze2d-large-easy-v1` and a hard version `maze2d-large-hard-v1`, respectively. The blue lines in the images correspond to the offline transitions remaining in the final dataset after pruning.

5.3.1 Environments

For evaluating our algorithm, we use various continuous control MuJoCo (Todorov et al., 2012) environments. For the offline datasets, we consider variants of D4RL (Fu et al., 2020), a collection of datasets for benchmarking offline RL algorithms.

Maze2d These environments involve the agent moving a force-actuated ball in a maze-like environment along the X and Y axes to a fixed target location by adjusting its velocity and direction. The state space $\mathcal{S} \in \mathbb{R}^4$ and the action space $\mathcal{A} \in \mathbb{R}^2$. These environments have varying levels of difficulties such as `maze2d-open-v1`, `maze2d-umaze-v1`, `maze2d-medium-v1` and `maze2d-large-v1`. The agent gets a positive reward when close to the target location and zero everywhere else. (There are also dense-reward versions of these environments, but they make the problem even simpler.)

For the offline datasets, we prune the `medium` and `large` variants by removing trajectories near the goal state to different extents, resulting in the datasets `maze2d-medium-easy-v1`, `maze2d-large-easy-v1`, and `maze2d-large-hard-v1`, which have varying levels of difficulty. This leaves some room for exploration during the active trajectory collection phase. Figure 5.1 visualizes these pruned datasets.

AntMaze This is an extension of the **Maze2d** environment in which the agent is replaced by an "Ant" instead, which can be manipulated by controlling its joints. The state space $\mathcal{S}$ is 29-dimensional, and the action space is 7-dimensional. For the offline dataset, we subsample 30% of the trajectories from the D4RL dataset to make the task challenging.

Locomotion We consider the **HalfCheetah**, **Hopper**, and **Walker2d** locomotion environments from OpenAI Gym MuJoCo, in which the objective is to control a 2D stick figure with multiple joints to stably move forward. For all these environments, we randomly subsampled 20% of the random and medium D4RL datasets to create our offline datasets.

CARLA This is a self-driving vehicle simulator wherein the agent has to control the acceleration and steering of a vehicle so as to stay in its lane, avoid collisions, and reach a specified goal. We use a predefined expert policy to collect the offline dataset at a roundabout with 4 exits. 8 starting "waypoints" are located equidistant from each other throughout the roundabout. The offline dataset is collected with 1 entry and 2 exits. However, the goal exit is not present in the offline dataset. The state space is augmented with the coordinates of the vehicle. For creating the offline dataset, we run a deterministic policy and collect 500K transitions.

IsaacSimGo1 This is a GPU-based simulator to control a legged 4×3 DOF quadrupedal robot using proprioceptive measurements along with ego-centric height information of the terrain. We use the `legged_gym` API (Rudin et al., 2022) to simulate Unitree **Go1** robots. Initially, we used the default walking policy for the physical robot, described as Mode 2 in *Go1 SDK HighLevel Interfaces* (2023), to collect an offline dataset comprising trajectories on a flat surface. For the final evaluation, the robot is expected to walk on a complex terrain consisting of flights of stairs and discrete obstacles, shown in Figure 5.2, requiring it to choose suitable starting locations during the active collection phase.

The state space was 45-dimensional, and we consider a total of 500K transitions as the offline dataset. We adapt it to the **IsaacSim**, where the size of the state space is 48 and the ego-centric height observation space is 187, resulting in a 235-dimensional observation space. Since the offline dataset was collected on flat regions, the 187-dimensional height observations were almost constant and were easy to adapt. We estimated velocities along the three spatial dimensions from the offline dataset itself.

We consider three terrains, `go1-easy`, `go1-medium` and `go1-hard`. The behavior policy was trained on the `go1-easy` terrain and achieves reasonably high rewards for the locomotion task on the flat surface. We then assume that the environment has been modified and the agent needs to update its policy as quickly as possible in the modified environment. We call this the `go1-medium` terrain where on a grid of size 49 (7×7), five cells are upward pyramids with steps,

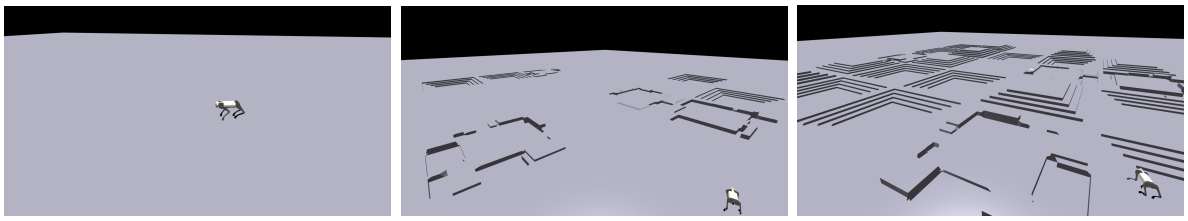


Figure 5.2: The figures display the terrains for the Unitree **Go1** robot experiments in the Nvidia **Isaac** Simulator. We named the three terrains (from left to right in order) `go1-easy`, `go1-medium`, and `go1-hard`. The behavior policy was trained on the `go1-easy` terrain and achieves reasonably high rewards for the locomotion task on the flat surface, as shown. However, we assume that the environment has been modified, and the agent needs to update its policy as quickly as possible in the modified environment. If the agent efficiently uses its exploration budget, then it will be able to generalize the experiences gathered during Active Collection and be able to get high rewards in the `go1-hard` terrain in spite of being given access to `go1-medium` terrain during Active trajectory collection.

five cells are downward pyramids with steps, five cells are uneven terrain (discrete obstacles), and the remaining 34 cells remain flat. The policy already trained on the flat regions can gain more information while adapting to the new terrain by exploring the unexplored obstacles (30% of the terrain). Finally, the `go1-hard` has only 5 cells of flat terrain, whereas the remaining comprise 19 upward pyramids, 19 downward pyramids, and finally 6 cells of uneven terrain. The agent needs to generalize the experiences gathered during Active Collection and be able to get high rewards in the `go1-hard` terrain in spite of being given access to only `go1-medium` terrain during Active trajectory collection.

5.3.2 Base Offline Algorithms

As described in Section 5.2, our overall algorithm consists of two phases. The first phase is the offline learning phase, in which two components are trained: (i) a base offline algorithm that learns the best possible policy based on just the given offline dataset, and (ii) an ensemble of representation models to be used to get uncertainty estimates for the next phase.

For the base offline algorithm, we choose among the following existing standard offline RL algorithms in the literature based on the task at hand.

TD3 + BC is a simple approach to offline reinforcement learning that combines the Twin-Delayed Deep Deterministic Policy Gradient algorithm (TD3) (Fujimoto et al., 2018) with Behavior Cloning (BC). TD3 is a popular and efficient online off-policy algorithm for reinforcement learning that extends and improves DDPG (Lillicrap et al., 2016) by incorporating clipped double-Q learning, delayed policy updates, and target policy smoothing. Behavior cloning is a

simple imitation learning algorithm that learns to imitate a policy from a given dataset of state-action pairs. TD3+BC works by first learning a Q-function from the dataset of state-action pairs. The Q-function is then used to train a deterministic actor with behavior cloning as a regularizer. Essentially, the policy is obtained by solving the following optimization problem:

$$\pi^* = \arg\max_{\pi} \mathsf{E}_{(s,a)\sim\mathcal{D}} \left[Q\left(s, \pi(s)\right) - \alpha \left(\pi(s) - a\right)^2 \right].$$

Conservative Q-Learning is an offline extension of the Soft Actor-Critic (SAC) (Haarnoja et al., 2018) algorithm that minimizes the following additional term for learning the critic along with the standard Bellman error.

$$\alpha \, \mathsf{E}_{s\in\mathcal{D}} \left[\log \sum_{a} \exp\left(Q(s,a)\right) - \mathsf{E}_{a\sim\mathcal{D}} Q(s,a) \right].$$

This prevents overestimation of the Q-values at state-action values that are not in the distribution of the offline dataset, leading to a conservative estimate of the Q-function.

For continuous action spaces, the summation over actions in the first term is replaced by an empirical average based on actions sampled from the uniform distributions and the current policy. The α term is updated via Lagrangian dual descent. The policy update is as in SAC. In our work, we used the d3rlpy (Seno and Imai, 2022) implementation of this algorithm.

Implicit Q-Learning (IQL) (Kostrikov et al., 2022) works by using expectile regression to learn the value function using only samples from the offline dataset. The learned value function is then used to extract a policy via advantage-weighted behavior cloning. The value function estimation updates are made according to the following objectives.

$$\begin{aligned} L_V(\psi) &= \mathsf{E}_{(s,a)\sim\mathcal{D}} \left[L_2^\tau(Q_{\hat{\theta}}(s,a) - V_\psi(s)) \right], \text{ and} \\ L_Q(\theta) &= \mathsf{E}_{(s,a,s')\sim\mathcal{D}} \left[\left(r(s,a) + \gamma V_\psi(s') - Q_\theta(s,a)\right)^2 \right], \end{aligned}$$

where $L_2^\tau(u) = |\tau - \mathbb{I}(u < 0)|u^2$ is the expectile loss. Once the value function is estimated reasonably well, the policy is extracted by optimizing the following loss.

$$L_\pi(\phi) = \mathsf{E}_{(s,a)\sim\mathcal{D}} \left[\exp\left(\beta \left(Q_{\hat{\theta}}(s,a) - V_\psi(s)\right)\right) \log_\phi(a|s) \right].$$

Behavior Proximal Policy Optimization (BPPO) (Zhuang et al., 2023) is an offline version of the PPO algorithm. Initially, a policy is learned to mimic the behavior policy using the given offline dataset. This policy is then updated just as in PPO, except that the transitions

are sampled from the offline dataset, and the advantage function is estimated using a learned Q-function corresponding to the behavior policy instead of the observed rewards.

In our work, we use TD3+BC as the base offline algorithm for the **Maze2d** and **Locomotion** environments, Behavior Cloning for legged quadrupedal locomotion, and CQL and IQL in **CARLA** and **AntMaze** environments, respectively. This is because TD3+BC does not work in environments where some notion of stitching is required for finding an optimal policy.

5.3.3 Uncertainty Model

5.3.3.1 Representation Models

For our purpose, we use $K = 5$ representation models without any weight sharing to compute the epistemic uncertainty for each of the datasets for the corresponding environment. We use 2 affine layers for encoding the state and action representations. For **Maze2d**, **Locomotion**, and **CARLA** environments, we use a hidden dimension of 256, while for **AntMaze** and **IsaacSim-Go1** we increase it to 512. We set $\lambda = 1.0$ for all our experiments.

5.3.3.2 ϵ-greedy Exploration Policy

In the case of algorithms such as TD3+BC, which has a deterministic actor that produces the same action for the same state, we cannot "sample" actions for a particular state. To remedy this, we add $\beta\ \mathcal{N}(\mathbf{0}, \mathbf{I})$ to get a candidate set of actions from which we pick the action that maximizes our uncertainty metric. For stochastic actor algorithms like IQL, we do not require the noise to sample actions, as we find that the underlying stochasticity of the policy is enough.

The uncertainty-based exploration policy is used in an ϵ-greedy setup, where with probability ϵ we select our uncertain action, otherwise we select the policy action. We tried combinations of $\epsilon \in \{0.1, 0.15, 0.2, 0.25, 0.5\}$ and $\beta \in \{0.1, 0.2, 0.3, 0.5\}$. For **Locomotion** environments, we find $\beta = 0.1$ and $\epsilon = 0.25$ to work best, while for simpler **Maze2d** environments, we find larger value $\beta = 0.3$ and $\epsilon = 0.5$ work better. In **AntMaze** environments where we use IQL, we do not add the noise and keep $\beta = 0$ with $\epsilon = 0.3$.

5.3.4 Baselines

To validate the effectiveness of our proposed active collection, we compare our method with a baseline that collects new trajectories after offline learning without active initial state selection and active exploration. The offline phase remains the same, wherein an offline RL algorithm is trained on the given dataset. In the second (fine-tuning) phase (denoted by FT in Table 5.1), new trajectories are collected starting from the original initial state distribution ρ of the MDP $\mathcal{M}$, using the learned offline policy π as the exploration policy.

Table 5.1: Results for our active method compared to respective baselines. Mean normalized scores (according to D4RL) are reported across various runs. As can be observed, we consistently improve the performance of the offline trained policy across multiple environments when compared to existing SOTA methods. We also observe a significant reduction in the number of samples required to reach the same performance as the baselines. (We denote inconclusive reduction by '-').

	BC	Offline	Offline + FT	Offline + RND	Offline + AC (Ours)	%age of less interactions
maze2d-medium-easy-v1	-4.5	-4.0	77.5	59.1	**134.3**	62.5
maze2d-large-easy-v1	1.7	-2.0	21.7	10.2	**197.3**	75
maze2d-large-hard-v1	-2.3	-2.0	6.0	1.0	**201.7**	62.5
Maze-pruned total	-5.1	-8.0	105.2	70.3	**533.3**	
antmaze-umaze-v0	62.0	86.7	86.1	81.5	**88.1**	37.5
antmaze-umaze-diverse-v0	54.0	56.0	43.9	39.2	**71.6**	50
antmaze-medium-play-v0	0.0	59.0	68.9	56.8	**73.1**	37.5
antmaze-medium-diverse-v0	1.3	62.3	68.5	62.1	**73.8**	25
antmaze-large-play-v0	0.0	10.3	19.9	14.0	**22.8**	25
antmaze-large-diverse-v0	0.0	9.0	19.8	9.9	**22.9**	37.5
AntMaze-subsampled total	117.3	283.3	307.1	268.2	**352.3**	
halfcheetah-random-v2	2.3	13.5	36.9	41.8	**42.5**	60
hopper-random-v2	4.2	8.2	26.3	23.6	**28.1**	55.5
walker2d-random-v2	2.0	7.9	9.1	10.8	**11.4**	33.3
halfcheetah-medium-v2	42.8	48.3	59.1	58.1	**62.7**	50
hopper-medium-v2	54.0	68.1	93.4	88.4	**96.7**	37.5
walker2d-medium-v2	73.1	83.6	84.9	85.2	**88.5**	-
Locomotion-subsampled total	178.4	229.6	309.7	307.9	**329.9**	
CARLA	0.0	0.0	72.1	88.8	**98.4**	67
unitree-go1-hard	23.1	23.1	34.6	46.7	**59.0**	50
Combined total	313.7	528.0	828.7	781.9	**1372.9**	

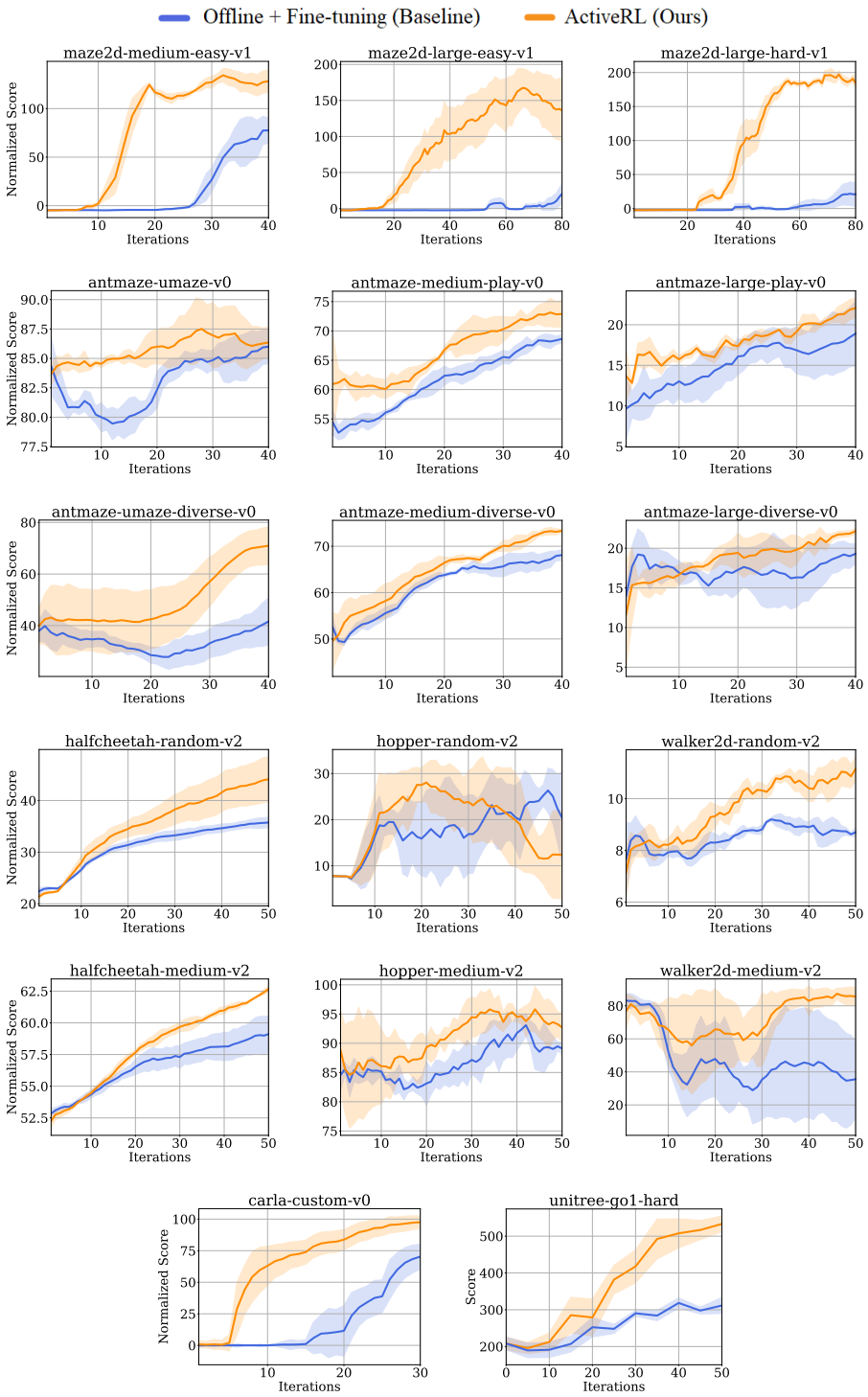


Figure 5.3: Results of our algorithm compared with the corresponding fine-tuning baseline. The plots show the results averaged over multiple random seeds, and the shaded region denotes the standard deviation. It can be observed that our method performs better overall when compared to the corresponding baselines.

Specifically, for the **Maze2d** and **Locomotion** environments, TD3+BC is used as the offline algorithm in the first phase. In the fine-tuning phase, the same training is continued on the newly collected data, with the α value being exponentially decayed to deal with the distribution shift (Beeson and Montana, 2022). For the **AntMaze** and **CARLA** environments, IQL and CQL, respectively, are used directly for both the offline and online fine-tuning phases, as in Kostrikov et al. (2022) and Kumar et al. (2020). For legged locomotion, we use BPPO (Zhuang et al., 2023) as the offline policy learning algorithm in the active phase, which is perturbed for exploration based on the uncertainty models as described in Section 5.3.3.2.

For these settings, each epoch in the finetuning or active collection phase consists of collecting X transitions from the environment and augmenting the dataset with these transitions, followed by Y updates using this augmented dataset. We use $X = 5000$ and $Y = 25000$ when our base algorithm is TD3+BC, while we use $X = 5000$ and $Y = 50000$ when our base algorithm is IQL. For all of the **AntMaze** tasks we run 8 epochs, and for **Locomotion** tasks we run 10 epochs. For the pruned versions of `maze2d-large-v1`, we run 16 epochs, and for the smaller `maze2d-medium-v1` versions, we run 8 epochs. In cases where our base algorithm is TD3+BC, we exponentially decay α by a factor of 5.0 across the epochs.

As an additional trajectory collection baseline for the active phase, we consider Random Network Distillation (Burda et al., 2019), in which the offline learned policy is distilled into an ensemble of smaller networks and used for exploration. Finally, we also report the performance of a simple Behavior Cloning (BC) baseline without any additional data collection.

The results are given in Table 5.1 and Figure 5.3. In the final column, we report the percentage of fewer additional interactions with the environment required by our algorithm to reach the best performance of the corresponding Offline + Fine-tuning baseline. One can observe that across the various environments and corresponding datasets, our method demonstrates a significant advantage over the corresponding baselines, both in terms of the rewards obtained as well as the number of samples required to achieve a certain performance.

In particular, we observe that our method performs well in scenarios where the behavior policy is sub-optimal and has not learned to explore certain areas of the environment where better rewards are present. Hence, our method augments the offline dataset that does not have good coverage of the state-space in a given MDP. For instance, in Table 5.1 we observe that our method achieves the most performance gain in the pruned `Maze2d` datasets where certain regions are missing from the offline dataset. Figures 5.4 and 5.5 illustrate the differences in exploration behavior between our method and the fine-tuning baseline.

Additionally, our active method was applied on top of TD3+BC, IQL and CQL, depending on the environment, showing that it is compatible with multiple offline algorithms.

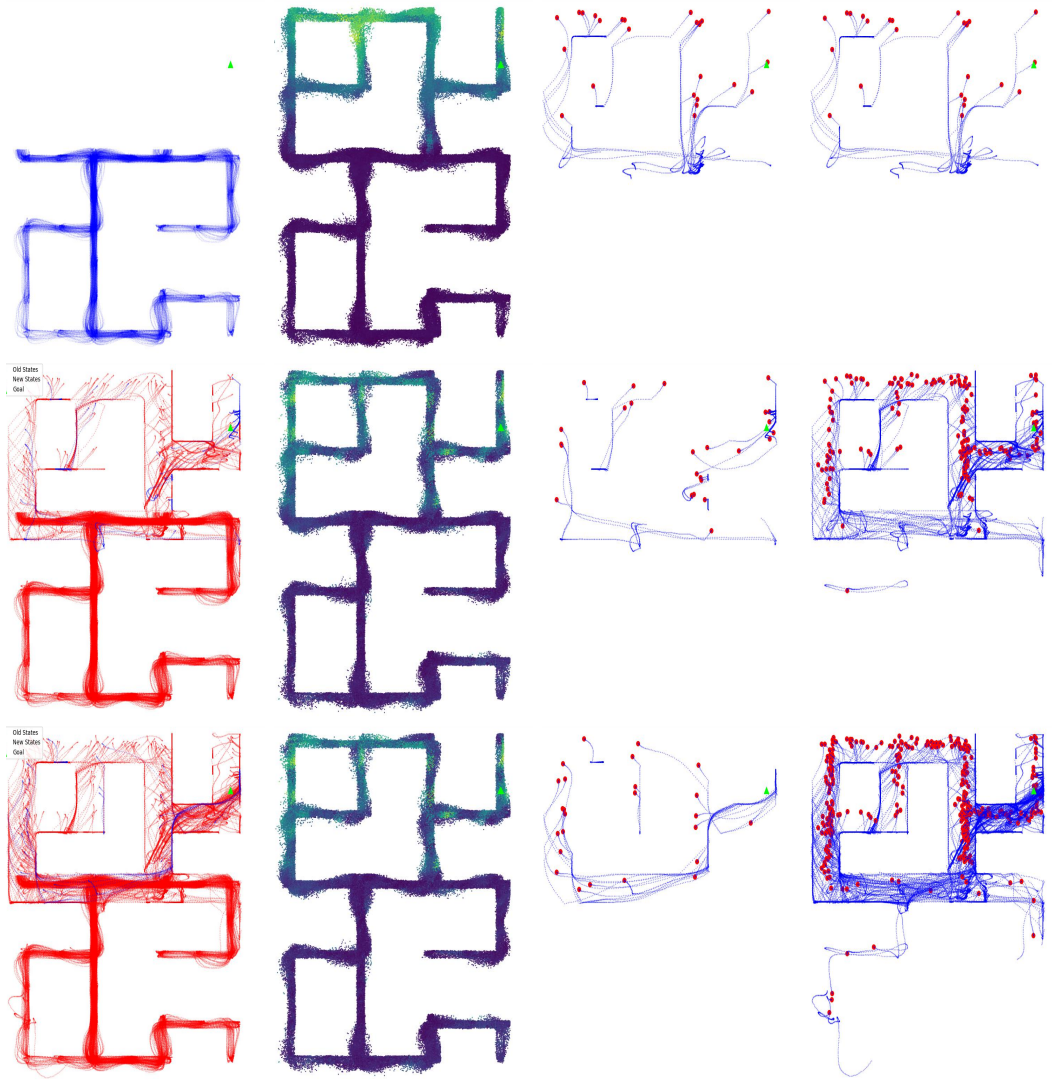


Figure 5.4: Evolution of the dataset during online trajectory collection using our method. First column: Dataset used for offline training. Second column: environment-aware uncertainty. Third column: Newly collected trajectories. Fourth column: Actively collected data till now. Uncertainty is high in regions where the data was not available. The 2nd and 3rd rows correspond to the 6th and 13th epochs of data collection. It can be noticed that the trajectory collection focuses primarily on the unobserved region of the offline dataset.

5.3.5 Ablations and Further Experiments

To effectively analyze the properties of our framework, we conduct additional experiments to understand the impact of different components of our algorithm.

Active Initial State Selection We perform an ablation by starting exploration only from states sampled from ρ, the initial state distribution of the original MDP $\mathcal{M}$. The results of this

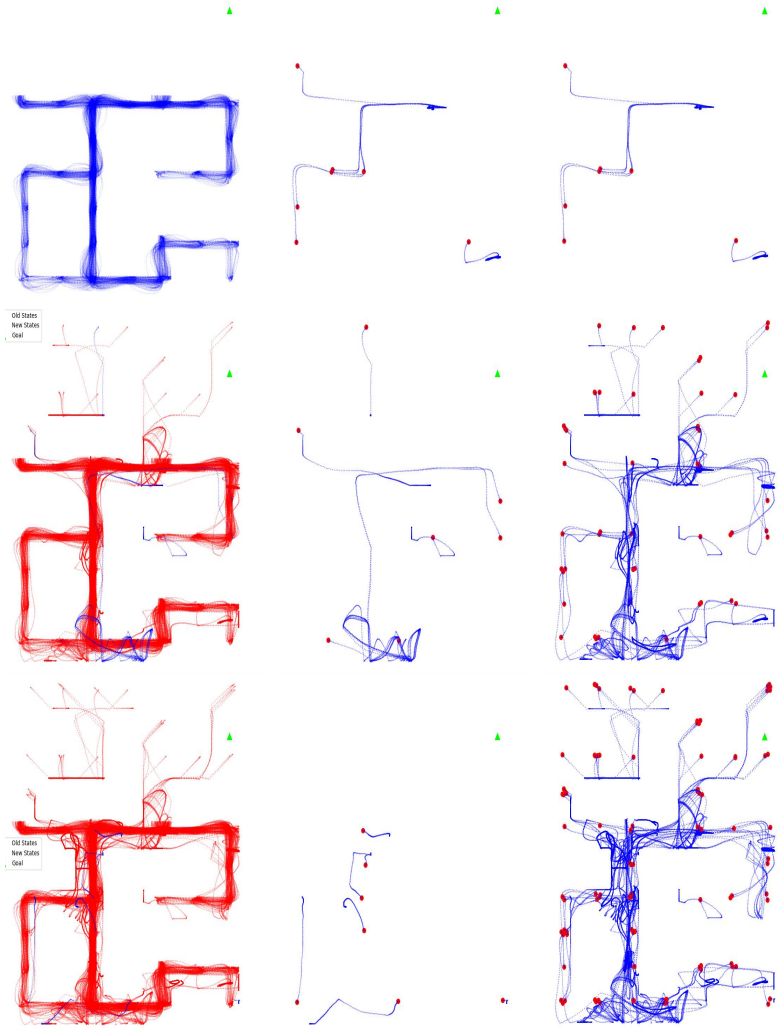


Figure 5.5: Demonstration of the evolution of the dataset during online trajectory collection at the same intervals as in Figure 5.4 using baseline fine-tuning methods. As can be noticed, in contrast to our method, other methods keep collecting redundant trajectories during the online data collection phase and hence do not perform very well, especially if the majority of the reward is in the unexplored regions of the offline dataset.

ablation are given in Table 5.2 as "I+U", denoting the choice of starting at the default initial states and using our uncertainty-based exploration policy.

Exploration Policy To study the importance of a suitable exploration policy, we conduct an ablation by replacing our uncertainty-based exploration strategy (U) with random exploration (R) and exploration using the policy learned offline (P) using the base offline algorithm. The results are shown in Table 5.2.

Table 5.2: Ablation results to understand various components of our approach. For the initial state selection, 'A' denotes active initial state selection, and 'I' denotes usage of the unaltered initial states from the MDP. 'R', 'P', and 'U' denote random policy, offline policy, and uncertainty-based exploration policy, respectively. Active initial state selection followed by an uncertainty-based exploration policy performs best. Further, A and U individually improve performance too.

Algorithm	`maze2d-large-easy`	`maze2d-large-hard`
BC	1.7	-2.3
Offline	-2.0	-2.0
I+R	45.5	25.0
I+P	0.7	0.2
I+U	51.1	-1.5
A+R	88.1	74.6
A+P	92.9	139.9
A+U	**133.8**	**176.3**

Table 5.3: Uncertainty metric ablation on **Maze2d**.

	var	mean	min	max
maze2d-medium-easy	97.1	108.8	105.6	**110.1**
maze2d-large-easy	111.3	122.5	123.9	**133.8**
maze2d-large-hard	144.6	170.8	159.2	**176.3**

It can clearly be seen that both these components of our algorithm are vital to obtain optimal results, while also providing an advantage when individually paired with other baseline strategies.

Uncertainty Metric In our algorithm, the maximum squared difference among the embeddings learned by the representation model ensemble is chosen as the uncertainty metric, which has been used both for initial state selection as well as the exploration policy. To verify whether this is the best choice, we have compared this metric with variants that use the variance of the model estimates, and the mean and minimum of the squared differences between the embeddings. The results in Table 5.3 show that the maximum squared difference is the optimal choice.

Pure Online Setting While the main goal of the algorithm is to efficiently augment a given offline dataset to obtain the optimal policy with the fewest additional interactions, it can also be applied to the setting where there is no offline dataset available. Therefore, we studied the effectiveness of our algorithm for exploring the environment from scratch. Here, we start

Table 5.4: Results for ablation with no initial dataset, analogous to the online setting. When TD3+BC is used as the base offline algorithm, it is evident that our approach performs better than the corresponding online algorithm TD3.

	TD3	ActiveRL (ours)
maze2d-umaze-v1	142.79	**164.8**
maze2d-medium-v1	148.1	**178.8**
maze2d-large-v1	98.7	**169**
unitree-go1-hard	43.9	**52.8**
Total	433.5	**565.4**

from a random policy, collect trajectories, and train on these experiences to simulate online learning. The results in Table 5.4 show that even in the absence of an initial offline dataset, our exploration strategy gains a significant advantage over the corresponding online algorithm.

5.4 Related Work

The use of uncertainty-based methods for actively labeling data points has been studied in the context of supervised learning (Balcan et al., 2006; Gal et al., 2017).

Similarly, in online reinforcement learning, successful methods for exploring MDPs typically rely on estimates of uncertainty about the Q-values (of state-action pairs) in order to encourage the agent to explore the environment (Osband et al., 2016). Some exploration strategies also rely on uncertainty-based intrinsic rewards or bonuses. Popular approaches include indirect methods for uncertainty estimation such as approximate count (Bellemare et al., 2016), random network distillation Burda et al. (2019), and curiosity-driven exploration (Pathak et al., 2017). Mai et al. (2022) learn variance ensembles for capturing the uncertainty.

In offline reinforcement learning, both model-free and model-based methods incorporate uncertainty in different ways. MOPO (Yu et al., 2020) and MOREL (Kidambi et al., 2020) are model-based methods in which the epistemic uncertainty of models learned on the offline dataset is explicitly used to induce pessimism in the trained policy. On the other hand, COMBO (Yu et al., 2021) incorporates conservatism without explicitly estimating the uncertainty of the model.

In a model-free setting, UWAC (Wu et al., 2021) approximates the uncertainty through dropout variational inference. EDAC (An et al., 2021) uses the variance of the gradients of an ensemble of Q networks.

The work of Yin et al. (2023) comes closest to our work in terms of application. However, their approach is applicable to the online setting and is primarily constrained to discrete control settings such as Atari 2600. Our approach differs in the following ways: (i) Unlike our approach,

they use an ensemble of Q-Networks, and the variance across Q-values defines the uncertainty metric, (ii) they allow resetting to a previously observed state, and (iii) they sample actions from a uniform distribution and use local planning for exploration.

Active Offline Policy Selection (Konyushova et al., 2021) studies a related problem where the goal is to collect additional trajectories for evaluating a given set of policies and determining the best among them. In contrast, our method deals with collecting trajectories for the final goal of learning an optimal policy from the augmented dataset and not just evaluating given policies.

In Go-Explore algorithms (Ecoffet et al., 2021), the agent explores and comes back to already observed states to explore again, which does not work when the environment is largely unexplored. Our method, by contrast, assumes a given set of states and chooses the optimal states from which to start based on the available offline dataset.

5.5 Outlook

In the realm of reinforcement learning, where agents need to explore the environment to continuously gather data to improve themselves, practical considerations force one to optimize how the data is collected and used. When an existing dataset that has been collected by some possibly unknown behavior policy is available, offline reinforcement learning techniques optimally make use of this data to learn agents in a completely offline fashion. However, the composition of this dataset frequently imposes a ceiling on the performance of any agent that is trained solely on the given data.

By taking motivation from active learning, which is well-studied in supervised learning settings, we studied how to optimally augment a given offline dataset so as to use existing offline algorithms and learn better agents on this augmented dataset. We proposed a two-phase active learning strategy with which agents, using an ensemble of representation models to estimate epistemic uncertainty, choose optimal regions of the state space to acquire trajectories and explore optimally from thereon to obtain the most useful agent-environment interactions that enhance their performance under a limited budget.

We demonstrated the efficacy of our method through extensive experimentation across a range of continuous control environments. Our results showed that our active trajectory collection method reduces the need for additional online interactions by up to 75% compared to standard fine-tuning approaches, while still significantly improving policy performance. Additionally, ablation studies confirmed the importance of each component of our proposed method, reinforcing the effectiveness of our uncertainty-driven exploration strategy.

Our findings contribute to the broader study of data-efficient reinforcement learning by

providing a principled method for selectively augmenting offline datasets. This work aligns with the concepts of active learning in supervised settings and extends them to the domain of sequential decision-making, where the challenges of exploration and trajectory collection are inherently more complex. Future work could explore further refinements to our approach, including more explicit and adaptive budget allocation strategies, and broader applicability to high-dimensional state spaces and real-world applications. Our results highlight the potential of intelligent exploration strategies in improving reinforcement learning efficiency while minimizing costly online interactions, paving the way for more practical and scalable deployment of RL agents in real-world settings.

Chapter 6

Reinforcement Learning under External Influence

In the mathematical framework of Markov Decision Processes that forms the core of reinforcement learning, an agent interacts with an 'environment', and at each time step, the agent is required to make a decision and take an action. The state of the environment changes stochastically in accordance with the Markov property, relying solely on the present state and the action taken by the agent. While solving various sequential decision-making problems, most RL algorithms assume that while the state of an MDP may be constantly changing, the rules governing state transitions remain unchanged. However, in practical applications, external events may influence the agent's environment and change its dynamics.

For example, in the context of portfolio management in finance, an agent has access to a state consisting of various fundamental and technical indicators of a select number of financial instruments and has to make decisions sequentially on whether to hold on to an instrument, sell it, short it, etc. However, the price movements of these instruments may be affected by external forces, such as government decisions like rate hikes or sudden market movements caused by events occurring in a completely different sector of the economy that are not being considered by the agent. Such an external event might exert lasting influence over different time scales depending on the kind of event that occurred. A rate cut might affect stock prices for a few months, while a flash crash might cease its effect in a few hours. An example illustrating the persistent non-stationarity effect of perturbations due to exogenous events is depicted in Figure 6.1.

As another example, consider a navigation problem in which an agent has control of a robot and needs to travel from one point to another. While the agent can learn to do this task in

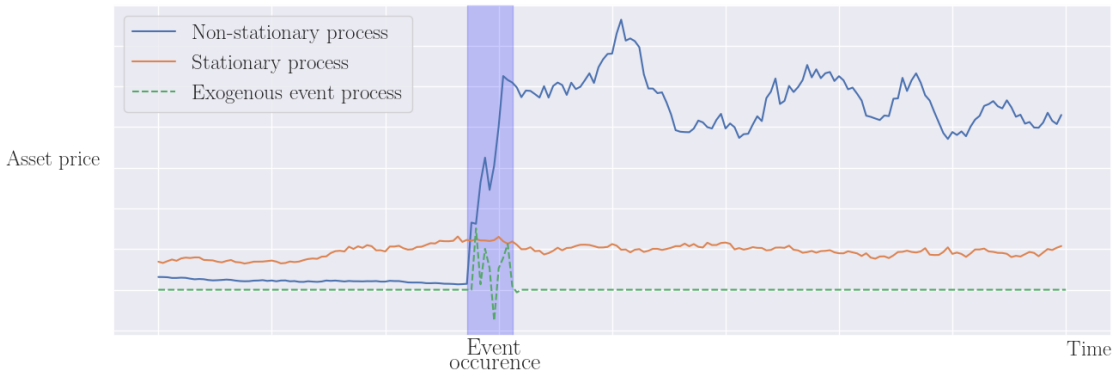


Figure 6.1: Sustained effect of transient non-stationarity. The plot shows the prices of two assets following a Geometric Brownian motion. A few exogenous Gaussian shocks to one asset force it to diverge significantly from the other over the long term, with a markedly different pattern of evolution long after the effect of the exogenous event decays.

ideal conditions, the optimal actions taken by the agent might depend on external conditions such as rain, sudden traffic changes, or human movement patterns. Hence, while an agent can learn to achieve goals in an ideal environment, it cannot be oblivious to external changes that influence the environment dynamics differently.

In the literature, non-stationarity in RL settings has been studied considering many special cases, mostly providing practical solutions. Some works model non-stationarity as piece-wise stationarity (Alegre et al., 2021; Hadoux et al., 2014; Li et al., 2025), while a few consider drifting environments (Lecarpentier and Rachelson, 2019; Cheung et al., 2020). Hallak et al. (2015) study Contextual MDPs, in which the environment dynamics change based on externally specified "contexts" that are episodic and possibly unknown chosen from among a finite set of known values. Tennenholtz et al. (2023) extend this setting to handle dynamic contexts that are known to the agent, change at each time step, and influence the state transition distribution. However, their contexts are not exogenous and come from a finite set of possible vectors.

In contrast to the existing literature, we study this problem in a more general setting, where an MDP is perturbed by an external discrete-time temporal process. This external process affects the dynamics of the MDP in a non-Markovian fashion, resulting in a non-stationary problem. This process, being exogenous to the decision process, does not depend on the state of the environment under consideration or the actions of the agent.

We study conditions that the exogenous process has to satisfy, along with its effect on the MDP, such that almost-optimal solutions exist that are tractable in some sense. We formulate variants of standard reinforcement learning algorithms that consider the above-mentioned non-

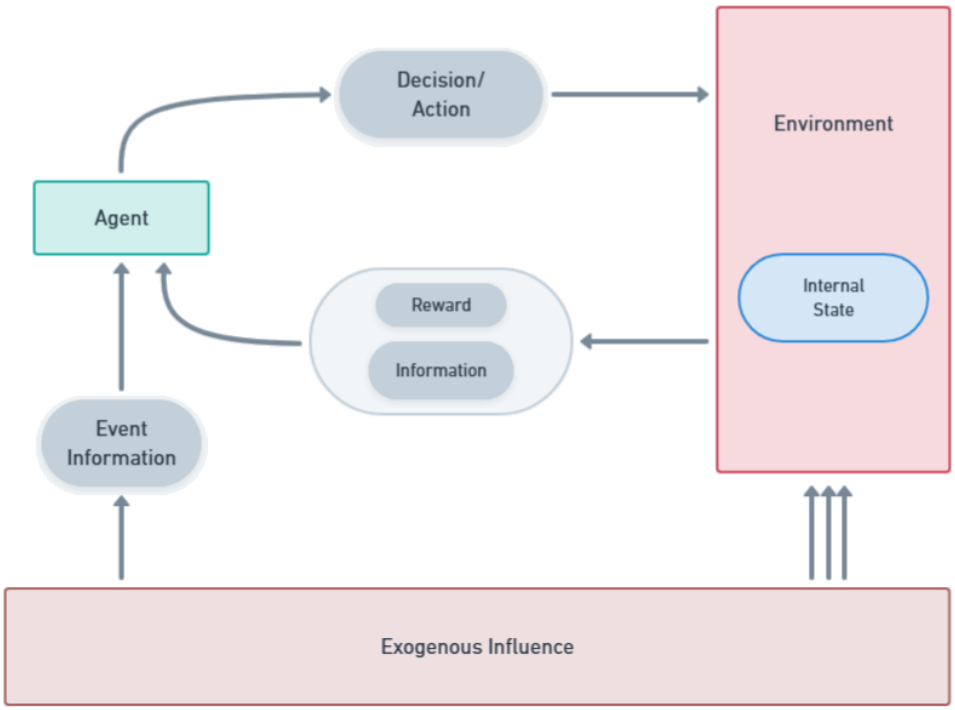


Figure 6.2: Reinforcement Learning under exogenous influence. An external temporal event process influences the dynamics of the environment. The agent has access to information about the events as they occur.

stationarity problem and theoretically analyze the properties of these algorithms. The results presented in this thesis can also be found in Ayyagari and Dukkipati (2024).

6.1 Preliminaries and Notation

Consider a Markov Decision Process (MDP) defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, Q, r, \gamma)$, where $\mathcal{S}$ is the state space, the set of all possible states of the environment, $\mathcal{A}$ is the action space, the set of all possible actions that can be taken by the agent, Q is the transition probability kernel, r is the reward function, and γ is the discount factor. At each time step t, the agent is in state $s \in \mathcal{S}$ and has to take action $a \in \mathcal{A}$ obtaining reward $r(s, a)$, and the environment transitions to state $s' \sim Q(\,.\,|s, a)$.

In general, $\mathcal{S}$ and $\mathcal{A}$ are Borel spaces, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a measurable function, and $\gamma \in (0, 1)$. Q is a stochastic kernel on $\mathcal{S}$ given $\mathcal{S} \times \mathcal{A}$. That is, $Q(.|y)$ is a probability measure on $\mathcal{S}$ for each $y \in \mathcal{S} \times \mathcal{A}$, and $Q(B|.)$ is a measurable function on $\mathcal{S} \times \mathcal{A}$ for each $B \in \mathcal{B}(\mathcal{S})$, the Borel σ-algebra on $\mathcal{S}$.

Given a class of possible policies Π, the goal of the agent is to determine a policy $\pi \in \Pi$ that achieves the maximum possible value function V^π, the expected discounted reward obtained by the agent following policy π:

$$V^\pi(s) = \mathsf{E}^\pi_{s_t,a_t}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right], \qquad \pi \in \Pi, s \in \mathcal{S},$$

the expected sum of discounted rewards obtained by an agent when it starts from a state s and follows a policy π resulting in a trajectory of states $s_0 = s, s_1, s_2, \ldots$ by taking actions $a_0, a_1, a_2, \ldots$. The expectation is over the states and actions in the agent's trajectory due to the stochasticity of the environment and possibly the policy itself. Let π^* be an optimal policy that achieves a corresponding value function V^{π^*}, denoted as V^*.

A measurable function $v : \mathcal{S} \to \mathbb{R}$ is said to be a solution to the Bellman optimality equation if it satisfies $v = \mathcal{T}v$, where $\mathcal{T}$ is the optimal Bellman operator defined as

$$[\mathcal{T}v](s) = \max_{a \in \mathcal{A}}\left[r(s,a) + \gamma \int_{\mathcal{S}} v(s')Q(ds'|s,a)\right], \qquad \text{for all } s \in \mathcal{S}. \tag{6.1}$$

A function that satisfies the optimal Bellman equation is a fixed point of the operator $\mathcal{T}$. Under suitable regularity conditions on the rewards and transition kernel, the optimal value function V^* satisfies this equation.

In this thesis, we have considered the setting where the agent received "rewards" from the environment, defined by the reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and its goal is to maximize the expected returns obtained. Equivalently, many works consider an agent incurring "costs", defined by an analogous cost function $c : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. In this case, the objective of the agent would be to minimize the expected discounted sum of the costs incurred, and so the Bellman operator is defined as a minimization of the right-hand side in (6.1).

For this cost setting, consider the following assumptions.

(i) The one-stage cost c is lower semi-continuous, non-negative and inf-compact on $\mathcal{S} \times \mathcal{A}$.

(ii) Q is strongly continuous.

(iii) There exists a policy π such that $V^\pi(s) < \infty$ for all $s \in \mathcal{S}$.

Under the above assumptions, we can make some statements about the existence and uniqueness of the optimal policy and value function, given by the following Theorem.

Theorem 4 (Theorem 4.2.3 of Hernández-Lerma and Lasserre (1996))**.** *Suppose the above two assumptions hold, then:*

1. *The optimal value function V^* is the pointwise minimal solution to the above Bellman equation. If $u(.)$ is any other solution to the above equation, then $u(.) \geq V^*(.)$.*

2. *There exists a function $\pi^* : \mathcal{S} \to \mathcal{A}$ such that $\pi^*(s) \in \mathcal{A}$ attains the minimum in the Bellman equation, i.e.,*

$$V^*(s) = c(x, \pi^*(s)) + \gamma \int V^*(s')Q(ds'|s, \pi^*(s)), \quad \text{for all } s \in \mathcal{S},$$

and the deterministic stationary policy π^* is optimal. Conversely, any deterministic stationary policy that is optimal satisfies the above equation.

3. *If an optimal policy exists, then there exists one that is deterministic and stationary.*

6.2 MDP under a Temporal Process

Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, Q, r, \gamma)$ be an MDP with state and action spaces $\mathcal{S}$ and $\mathcal{A}$ respectively, transition kernel Q, discount factor γ, and reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, which defines the reward $r(s, a, s')$ as a function of the state-action pair (s, a) and the next state s'. Consider an external discrete-time temporal process $X = (t_i, X_i)_{i \in \mathbb{N}}$ that influences the transition probabilities of $\mathcal{M}$. Here, X is exogenous; hence it is not affected by $\mathcal{M}$. The probability of an event occurring in the external process at time t is a function of all the past (external) events till that time. Associated with each event is a probabilistic mark $x \in \mathcal{X}$, where $\mathcal{X}$ is a closed subset of $\mathbb{R}^d$.

Let $H_t = \{(t_1, x_1), (t_2, x_2), \ldots, (t_j, x_j)\}$ be the history of external events till time t, where the ordered pair (t_i, x_i) denotes an event with mark x_i that occurred at time t_i. History H_t perturbs the stochastic kernel Q of the MDP to give rise to a new transition kernel Q_{H_t} on $\mathcal{S}$ given $\mathcal{S} \times \mathcal{A}$. At the same time, this history also determines the next event that occurs in the temporal process. We denote the distribution of the event mark at time t by $Q^X_{H_t}$, which is a probability distribution on $\mathcal{X}$. Since this describes the event distribution that is external to the MDP, it does not depend on the state of the MDP or the action taken by the agent.

We consider a discrete-time event process and assume the following.

(A1) There exists an event with a mark, say $X = 0$ (zero), that is equivalent to a non-event.

(A2) Events that occurred in the distant past have a reduced influence on the current probability transition kernel of the MDP. Specifically, there exists a convergent series $\sum_T M_T$ of real numbers, such that at any time t, for any event histories H_t and H'_t, if the MDP is

in state s and action a is taken, and if H_t and H'_t differ by only one event at time t', then

$$\mathsf{TV}\left(Q_{H_t}(.|s,a), Q_{H'_t}(.|s,a)\right) < M_T, \quad \text{for all } s \in \mathcal{S}, a \in \mathcal{A},$$

provided $t - t' \geq T$. That is, if an event is older than T, its influence on the current probability distribution has a Total Variation distance upper bounded by M_T. Furthermore, Q_{H_t} is conditionally independent of previous states given the current state. That is, Q_{H_t} is Markovian with respect to the state.

(A3) Similarly, events that occurred in the distant past also have a reduced influence on the probability distribution of the current event mark. Specifically, there exists a convergent series $\sum_T N_T$ of real numbers such that for event histories H_t and H'_t at time t that differ at only one event at time t', if $t - t' \geq T$, then

$$\mathsf{TV}\left(Q^X_{H_t}, Q^X_{H'_t}\right) < N_T.$$

Further, s_{t+1} and x_{t+1} are conditionally independent given the current state s_t, history H_t of events, and action a_t.

Along with the above assumptions **(A1-A3)**, the MDP $\mathcal{M}$ under the influence of the exogenous event process needs to satisfy a set of regularity conditions for guaranteeing the existence and uniqueness of optimal solutions.

Define a new expected reward function $r_P : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ induced from r by P as

$$r_P(s,a) = \mathsf{E}_{s' \sim P(.|s,a)}\left[r(s,a,s')\right],$$

where P is any stochastic kernel on $\mathcal{S}$ given $\mathcal{S} \times \mathcal{A}$. We assume the following regularity conditions on r and Q.

(B1) r_P is a bounded measurable function. Without any loss in generality, assume its co-domain is $[0,1] \subset \mathbb{R}$.

(B2) The function $r_P(s,.)$ is upper semi-continuous for each state $s \in \mathcal{S}$ and any probability distribution $P = Q_H$ induced as a transition kernel on the MDP by any possible external event history H of the temporal process.

(B3) r_P is sup-compact on $\mathcal{S} \times \mathcal{A}$, i.e., for every $s \in \mathcal{S}, r \in \mathbb{R}$ and any probability distribution $P = Q_H$ induced as a transition kernel on the MDP by any possible external event history H of the temporal process, the set $\{a \in A : r_P(s,a) \geq r\} (\subseteq A)$ is compact.

(B4) Q is strongly continuous.

(B5) Q_H and Q_H^X are strongly continuous for any feasible event history H.

Note that a transition kernel P is said to be strongly continuous if $v(s,a) = \mathsf{E}_{s' \sim P(.|s,a)}[v(s')]$ is continuous and bounded on $\mathcal{S} \times \mathcal{A}$ for every measurable bounded function v on $\mathcal{S}$.

6.2.1 Examples

Non-Markov self-exciting event processes that can be described as above are studied extensively in finance, economics, epidemiology, etc., in the form of variants of Hawkes and jump processes. While they are studied predominantly in continuous time and sometimes in discrete time, we consider discrete-time versions to fit into the traditional discrete-time reinforcement learning framework.

Consider a discrete-time marked Hawkes process $(E_t, X_t)_{t \in \mathbb{N}}$ defined as follows: $(E_t)_{t \in \mathbb{N}}$ is a sequence of random variables taking values in $\{0, 1\}$, with E_t sampled from $\mathsf{Bernoulli}\,(p_t)$, where the intensity p_t at time t depends on the realizations of E_t at the previous time steps as

$$\begin{aligned} p_1 &= \alpha_0, \text{ and} \\ p_t &= \alpha_0 + \sum_{t'=1}^{t-1} \alpha_{t-t'} E_{t'} \quad \text{for } t > 1, \end{aligned}$$

and the marks X_t are real-valued Gaussian random variables clipped to $[-b, b]$, satisfying

$$\begin{aligned} &X_1 \sim \mathcal{N}_{[-b,b]}(0,1), \text{ and} \\ &X_t \begin{cases} \sim \mathcal{N}_{[-b,b]}\left(\sum_{t'=1}^{t-1} \beta_{t-t'} E_t X_t,\ 1\right) & \text{if } E_t = 1, \\ = 0 & \text{otherwise,} \end{cases} \quad \text{for } t > 1, \end{aligned}$$

where $\mathcal{N}_{[-b,b]}$ is the normal distribution clipped to be within interval $[-b, b]$, and (α_t) and (β_t) are sequences that converge sufficiently quickly to zero. The sum $S_t = \sum_{t'=1}^{t} E_t$ is a self-exciting counting process that is the discrete-time counterpart of the continuous-time Hawkes process.

The above-defined sequence of random variables satisfies Assumptions **(A1)** and **(A2)** required for the external process. X_t is zero when no event occurs, and old events have a reduced and decaying influence on current events. More precisely, given two historical sequences $(X_{t'})_{t'<t}$ and $(X'_{t'})_{t'<t}$ that only differ by one event at some time $t' \leq t - T$ with $E'_{t'} = X'_{t'} = 0$ and

$E_{t'} = 1$, $X_{t'} = x$, the difference in intensities is $p_t - p'_t = \alpha_{t-t'}$, and hence,

$$\begin{aligned}
\mathsf{TV}\left(X_t, X'_t\right) &\leq \alpha_{t-t'} + p_t \mathsf{TV}\left(\mathcal{N}_{[-b,b]}\left(\sum_{t''=1}^{t-1} \beta_{t-t''} E_t X_t,\ 1\right), \mathcal{N}_{[-b,b]}\left(\sum_{t''=1}^{t-1} \beta_{t-t''} E_t X'_t,\ 1\right)\right) \\
&\leq \alpha_{t-t'} + p_t \mathsf{TV}\left(\mathcal{N}\left(\sum_{t''=1}^{t-1} \beta_{t-t''} E_t X_t,\ 1\right), \mathcal{N}\left(\sum_{t''=1}^{t-1} \beta_{t-t''} E_t X'_t,\ 1\right)\right) \\
&= \alpha_{t-t'} + p_t \ \mathsf{erf}\left(\frac{\beta_{t-t'} x}{2\sqrt{2}}\right) < \alpha_T + \mathsf{erf}\left(\frac{\beta_T b}{2\sqrt{2}}\right) = N_T. \qquad (6.2)
\end{aligned}$$

For sufficiently fast converging α_T and β_T, the total variation perturbation N_T due to events older than T time steps goes to zero, satisfying Assumption **(A2)**.

While (α_n) and (β_n) are any general sequences that need to satisfy some regularity conditions (Seol, 2015) and the above bound, they could also decay in a more structured manner. For instance, for some $c, \lambda > 0$, letting $\alpha_n = ce^{-\lambda n}$ gives a process that has exponentially decaying intensity due to past events. In the terminology employed to describe the standard continuous-time Hawkes process, $\alpha_0 = c$ is like the base or background intensity, and $e^{-\lambda x}$ is the excitation function. Similarly, β_n, which determines the distribution of the event, could also be a parametric sequence that decays in an exponential or polynomial fashion.

6.3 Guarantees for the Existence of Almost-Optimal Solutions

Consider the MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, Q, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are closed subsets of $\mathbb{R}^m$ and $\mathbb{R}^n$ respectively, and $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward as a function of the state, action, and resultant next state. Given an external discrete-time temporal process $X = \{(t_i, X_i)\}_i$ that acts on MDP $\mathcal{M}$, we construct a new augmented MDP $\mathcal{M}_X$, where the marks of all the events that occurred till time t are appended to the state $s \in \mathcal{S}$ to form a new augmented state $\bar{s} \in \bar{\mathcal{S}}$, where

$$\bar{s} = (s, x_t, x_{t-1}, x_{t-2}, x_{t-3}, \dots), \text{ and } \bar{\mathcal{S}} = \mathcal{S} \times \prod_{t=0}^{\infty} \mathcal{X}\left(\subseteq \mathbb{R}^m \times \prod_{t=0}^{\infty} \mathbb{R}^d\right).$$

Here, x_t is the mark of the external event that occurred at time t. If no event occurred at time t, then $x_t = 0$.

Because of the interference from external events, the decision process $(\mathcal{S}, \mathcal{A}, Q, r, \gamma)$ does not remain an MDP since the transition function is being perturbed by external events. However, the corresponding decision process $\mathcal{M}_X = (\bar{\mathcal{S}}, \mathcal{A}, Q_H, r, \gamma)$ is a Markov Decision Process because

all the factors that affect the transitions are included in the augmented state $\bar{s} \in \bar{\mathcal{S}}$.

Further, the state space of $\mathcal{M}_X$ is infinite-dimensional. However, Assumptions **(A2)** and **(A3)** in Section 6.2 allow one to study the possibility of finding a suitable policy that only depends on a finite horizon of past events. The following result formalizes this.

Theorem 5. *Suppose the MDP $\mathcal{M}$ and the external process satisfy the assumptions **(A1-A3)** and the regularity conditions **(B1-B5)**. Then, we have the following.*

(1) There exists a deterministic optimal policy for the augmented MDP $\mathcal{M}_X$, with corresponding optimal value function V^*, that satisfies the optimal Bellman equation.

(2) For any $\epsilon > 0$, there exists a time horizon T and a policy $\pi^{(T)}$ that depends only on the current state and past T events such that

$$V^{\pi^{(T)}} \geq V^* - \epsilon.$$

That is, for every required maximum suboptimality ϵ, there exists a corresponding "finite-horizon" policy that achieves that ϵ-suboptimal value. Further, the past event horizon T required for ϵ-suboptimality is T that satisfies

$$\sum_{t=T+1}^{\infty} M_t < \frac{\epsilon(1-\gamma)^2}{4} \text{ and } \sum_{t=T+1}^{\infty} N_t < \frac{\epsilon(1-\gamma)^2}{4}. \tag{6.3}$$

This result guarantees that the formulated problem is well-posed, with an optimal solution that can be approximated using a tractable policy that is a function of the current state and only a finite history of events. This approximation can be as accurate as necessary based on the properties of the exogenous event process and the size of the event history considered. To prove Theorem 5, we establish the following result, whose proof is given in Section 6.A.1.

Lemma 3. *Consider a new auxiliary MDP $\mathcal{M}_X^{(T)}$ that has the same underlying stochastic process as the augmented MDP $\mathcal{M}_X$, with one difference: only events in the past T time steps affect the current state transition and event distribution, with the events before that having no effect and being effectively zero. Let π be an arbitrary policy as a function of the augmented state $\bar{s} \in \bar{S}$. That is, π can be either stochastic or deterministic and can depend on the current state $s \in \mathcal{S}$ and any number of past events in $\mathcal{X}$. Then,*

$$\left\| V^{\pi}_{\mathcal{M}_X^{(T)}} - V^{\pi}_{\mathcal{M}_X} \right\|_{\infty} \leq \frac{1}{1-\gamma} \left(\|r\|_{\infty} + \gamma \left\| V^{\pi}_{\mathcal{M}_X^{(T)}} \right\|_{\infty} \right) \sum_{t=T+1}^{\infty} (M_t + N_t),$$

where r is the reward function and $V^{\pi}_{\mathcal{M}}$ denotes the value function of policy π in MDP $\mathcal{M}$.

Proof of Theorem 5. The state space, being a countable product of Borel sets, is Borel. So, part (1) follows from the regularity conditions **(B1-B5)** satisfying the assumptions of Theorem 4.2.3 in Hernández-Lerma and Lasserre (1996), coupled with the boundedness of the reward function. For part (2), let π^* be the deterministic optimal stationary policy for $\mathcal{M}_X$ and let $\pi^{*(T)}$ be the deterministic optimal stationary policy for $\mathcal{M}_X^{(T)}$. From Lemma 3, since $\|r\|_\infty \leq 1$ and $\|V\|_\infty \leq \frac{1}{1-\gamma}$,

$$\begin{aligned} V^{\pi^{*(T)}}_{\mathcal{M}_X} \geq V^{\pi^{*(T)}}_{\mathcal{M}_X^{(T)}} - \frac{1}{(1-\gamma)^2}\left(\sum_{t=T+1}^{\infty}(M_t+N_t)\right) &\geq V^{\pi^*}_{\mathcal{M}_X^{(T)}} - \frac{1}{(1-\gamma)^2}\left(\sum_{t=T+1}^{\infty}(M_t+N_t)\right) \\ &\geq V^{\pi^*}_{\mathcal{M}_X} - \frac{2}{(1-\gamma)^2}\left(\sum_{t=T+1}^{\infty}(M_t+N_t)\right). \end{aligned}$$

Since both series $\sum_t M_t$ and $\sum_t N_t$ converge, there exists $T \in \mathbb{N}$ satisfying (6.3). Choosing such a T proves the result. □

6.4 A Policy Iteration Algorithm

Now that we have established the existence of a ϵ-optimal policy that is a function of only a finite event horizon, we wish to find such a policy. We propose a policy iteration algorithm that alternates between approximate policy evaluation and approximate policy improvement. This procedure is listed in Algorithm 10.

The policy evaluation step considers candidate value functions that are functions of the finite event horizon. We define the value function in $\mathcal{M}_X$ at an infinitely augmented state as a function of the state augmented by just a finite event horizon, by sampling the previous older events from some arbitrary distribution μ_1. In practice, μ_1 is just the actual event process. The value function can thus be approximated using Monte Carlo methods. For the purpose of analysis, we consider the exact evaluation of this approximate value function.

In the policy improvement step, the policy is improved based on the reward and value function resulting from one transition. This transition can be due to past events coming from any arbitrary distribution μ_2. The deterministic policy at an augmented state depends only on the finite event horizon and is just the action that maximizes the right-hand side, which depends on the approximate value function.

Solving the optimization problem in the policy improvement step requires knowledge of the models $r(s,a,s')$, Q, and Q^X. In this work, we ignore any approximation errors that arise due

to estimating the models and analyze the algorithm by assuming the exact step is taken as defined.

Algorithm 10 Policy Iteration

Start with arbitrary deterministic policy $\pi_0 : \mathcal{S} \times \mathcal{X}^{T+1} \to A$

for each $k \in \{0, 1, \dots\}$ till termination **do**

// `Approximate Policy Evaluation`

$$\hat{V}_k\left((s, x_{0:\infty})\right) = \mathsf{E}_{x'_{T+1:\infty} \sim \mu_1} V^{\pi_k}\left(\left(s, x_{0:T}, x'_{T+1:\infty}\right)\right).$$

// `Approximate Policy Improvement`

$$\pi_{k+1}\left((s, x_{0:\infty})\right) = \underset{a \in A}{\operatorname{argmax}} \underset{\substack{x' \sim Q^X_{x_{0:T}, x'_{T+1:\infty}}(.|s) \\ x'_{T+1:\infty} \sim \mu_2}}{\mathsf{E}_{s' \sim Q_{x_{0:T}, x'_{T+1:\infty}}(.|s,a)}} \left[r(s, a, s') + \gamma \hat{V}_k\left((s', x', x_{0:T-1})\right) \right].$$

end for

6.4.1 Analysis

For the analysis of this algorithm, we need the following results, which bound the difference in the value function at two different augmented states when they differ in events that are more than T time steps old.

Lemma 4. *Let π be a policy that depends only on the current state and the past T events. Then,*

$$\sup_{\bar{s}=(s,x_{0:\infty})} \left| V^\pi(\bar{s}) - V^\pi\left((s, x_{0:T}, (0)_{T+1:\infty})\right) \right| < \frac{1}{(1-\gamma)^2} \sum_{t=T+1}^{\infty} (M_t + N_t).$$

Here, $(s, x_{0:T}, (0)_{T+1:\infty}) = (s, x_0, x_1, \dots, x_T, 0, 0, \dots) \in \bar{S}$ is the extended state that has been essentially "truncated" and made finite by replacing events older than T time steps by zero. The proof of Lemma 4 is provided in Section 6.A.2.

Corollary 5. *For a given policy π that is a function of events only in the past T time steps, and for any two augmented states $\bar{s}_1$ and $\bar{s}_2$ that differ in any number of events that occurred before the previous T time steps,*

$$\left| V^\pi(\bar{s}_1) - V^\pi(\bar{s}_2) \right| \leq \frac{2}{(1-\gamma)^2} \sum_{t=T+1}^{\infty} (M_t + N_t).$$

Essentially, this means that if a policy considers only events in the past T time steps, its value at states differing at older events differs by an amount proportional to the extent of non-stationarity induced by all events older than T steps.

This helps characterize the behavior of the policy iteration procedure described in Algorithm 10. While exact policy iteration results in a sequence of policies that converge to the optimal policy, our approximate policy iteration produces policies whose value functions satisfy the following result, whose proof is given later in this section.

Lemma 6.

$$V^{\pi_{k+1}} \geq V^{\pi_k} - \frac{3(1+\gamma)}{(1-\gamma)^3} \sum_{t=T+1}^{\infty} (M_t + N_t).$$

That is, while there is no guarantee that the sequence of value functions $\{V^{\pi_k}\}$ is non-decreasing, Lemma 6 provides a guarantee that they do not decrease by more than a specific amount. This is due to the approximation error $\epsilon = \frac{2}{(1-\gamma)^2} \sum_{t=T+1}^{\infty} (M_t + N_t)$ at each time step induced by the non-stationarity due to the exogenous events.

In general, approximate policy iteration algorithms converge under the assumption that the approximation errors slowly vanish over the course of the algorithm. However, our approximation error of ϵ remains constant throughout, leading to the above situation. Intuitively, it would seem that such a small approximation error ϵ should only disturb the convergence of the algorithm when we are ϵ-close to the optimal solution. During the initial iterations, when far from the solution, such small approximation errors should not matter.

We next establish that the degradation of the value function can occur only in parts of the state space where the Bellman error is close to zero. A Bellman error of zero generally, although not always, corresponds to the optimal policy. This interplay between Bellman error and approximation error in our algorithm is formalized by the following result.

Theorem 6. *At iteration k of the policy iteration procedure given in Algorithm 10, for any augmented state $\bar{s} \in \bar{\mathcal{S}} = \mathcal{S} \times \prod_{t=0}^{\infty} \mathcal{X}$, at least one of the following holds:*

$$V^{\pi_{k+1}}(\bar{s}) \geq V^{\pi_k}(\bar{s}), \text{ or} \tag{6.4}$$

$$|\mathcal{T}V^{\pi_k}(\bar{s}) - V^{\pi_k}(\bar{s})| < \frac{11}{(1-\gamma)^3} \sum_{t=T+1}^{\infty} (M_t + N_t), \tag{6.5}$$

where V^{π} denotes the value function of policy π in the MDP $\mathcal{M}_X$.

That is, the performance of the policy improves everywhere except the region of the state space where the Bellman error is very small. The faster the decay of exogenous event influence, the lesser the approximation error and the smaller the region of state space where the policy may not improve.

Corollary 7. *Consider an MDP with the exogenous process being a discrete-time Hawkes process as described in Section 6.2.1, with an exponentially decaying excitation function $c_\alpha e^{-\lambda_\alpha t}$, along with an exponentially decaying $\beta_t = c_\beta e^{-\lambda_\beta t}$ and state transition perturbations that decay as $M_t = \bar{c}e^{-\bar{\lambda}t}$. Then, each step k of the policy iteration algorithm is guaranteed to keep the value function of the policy non-decreasing everywhere in the state space except regions $\bar{s} \in \bar{\mathbb{S}}$ satisfying*

$$|\mathcal{T}V^{\pi_k}(\bar{s}) - V^{\pi_k}(\bar{s})| < \frac{11}{(1-\gamma)^3}\left(\frac{\bar{c}}{\bar{\lambda}}e^{-\bar{\lambda}T} + \frac{c_\alpha}{\lambda_\alpha}e^{-\lambda_\alpha T} + \frac{c_\beta b}{\sqrt{2\pi}\lambda_\beta}e^{-\lambda_\beta T}\right).$$

As the time window T considered increases, the error reduces as $e^{-\{\bar{\lambda},\lambda_\alpha,\lambda_\beta\}T}$. Hence, depending on the rate of influence decay, increasing the time window beyond a certain point gives diminishing returns. The error also has a proportional dependence on c_α, the base intensity of the Hawkes process, and an inverse dependency on λ_α, the decay of the excitation function of the Hawkes process.

Proof of Theorem 6. Which of the given two statements holds depends on whether or not the state $\bar{s}$ falls in the "sub-optimality" set B, which is the set of all states $\bar{s} = (s, x_{0:\infty}) \in \bar{\mathbb{S}}$ that satisfy

$$\mathsf{E}_{\substack{s'\sim Q_{x_{0:T},x'_{T+1:\infty}}(.|s,a)\\ x'\sim Q^X_{x_{0:T},x'_{T+1:\infty}}(.)\\ x'_{T+1:\infty}\sim\mu_2\\ a=\pi_k(\bar{s})}}\left[r(s,a,s') + \gamma\hat{V}_k((s',x',x_{0:T-1}))\right]$$
$$< \max_{a\in\mathcal{A}} \mathsf{E}_{\substack{s'\sim Q_{x_{0:T},x'_{T+1:\infty}}(.|s,a)\\ x'\sim Q^X_{x_{0:T},x'_{T+1:\infty}}(.)\\ x'_{T+1:\infty}\sim\mu_2}}\left[r(s,a,s') + \gamma\hat{V}_k((s',x',x_{0:T-1}))\right] - \delta$$

for some suitable $\delta > 0$ yet to be chosen. It is to be noted that whether or not this condition holds, the above inequality or the corresponding equality holds for $\delta = 0$ by the definition of π_{k+1}.

Suppose the above condition holds. Then, for $\epsilon = \frac{2}{(1-\gamma)^2}\sum_{t=T+1}^{\infty}(M_t + N_t)$,

$$
\begin{aligned}
V^{\pi_k}((s, x_{0:\infty})) &\leq \mathsf{E}_{x'_{T+1:\infty}\sim\mu_2} V^{\pi_k}((s, x_{0:T}, x'_{T+1:\infty})) + \epsilon \\
&= \mathsf{E}_{\substack{s'\sim Q_{x_{0:T},x'_{T+1:\infty}}(.|s,a) \\ x'\sim Q^X_{x_{0:T},x'_{T+1:\infty}}(.) \\ x'_{T+1:\infty}\sim\mu_2 \\ a=\pi_k(\bar{s})}} \left[r(s, a, s') + \gamma V^{\pi_k}((s', x', x_{0:T}, x'_{T+1:\infty}))\right] + \epsilon \\
&\leq \mathsf{E}_{\substack{s'\sim Q_{x_{0:T},x'_{T+1:\infty}}(.|s,a) \\ x'\sim Q^X_{x_{0:T},x'_{T+1:\infty}}(.) \\ x'_{T+1:\infty}\sim\mu_2 \\ a=\pi_k(\bar{s})}} \left[r(s, a, s') + \gamma \hat{V}_k((s', x', x_{0:T-1}))\right] + \gamma\epsilon + \epsilon \\
&< \mathsf{E}_{\substack{s'\sim Q_{x_{0:T},x'_{T+1:\infty}}(.|s,a) \\ x'\sim Q^X_{x_{0:T},x'_{T+1:\infty}}(.) \\ x'_{T+1:\infty}\sim\mu_2 \\ a=\pi_{k+1}(\bar{s})}} \left[r(s, a, s') + \gamma \hat{V}_k((s', x', x_{0:T-1}))\right] + \gamma\epsilon + \epsilon - \delta \\
&\leq \mathsf{E}_{\substack{s'\sim Q_{x_{0:\infty}}(.|s,a) \\ x'\sim Q^X_{x_{0:\infty}}(.) \\ a=\pi_{k+1}(\bar{s})}} \left[r(s, a, s') + \gamma \hat{V}_k((s', x', x_{0:T-1}))\right] \\
&\qquad + \left(1 + \frac{\gamma}{1-\gamma}\right) \sum_{t=T+1}^{\infty} (M_t + N_t) + \gamma\epsilon + \epsilon - \delta \\
&= \mathsf{E}_{\substack{s'\sim Q_{x_{0:\infty}}(.|s,a) \\ x'\sim Q^X_{x_{0:\infty}}(.) \\ a=\pi_{k+1}(\bar{s})}} r(s, a, s') + \gamma \mathsf{E}_{\substack{s'\sim Q_{x_{0:\infty}}(.|s,a) \\ x'\sim Q^X_{x_{0:\infty}}(.) \\ a=\pi_{k+1}(\bar{s})}} V^{\pi_k}((s', x', x_{0:\infty})) \\
&\qquad + \frac{1}{1-\gamma} \sum_{t=T+1}^{\infty} (M_t + N_t) + \gamma\epsilon + \gamma\epsilon + \epsilon - \delta.
\end{aligned}
$$

That is, $V^{\pi_k}(\bar{s})$ can be bounded recursively in terms of the reward obtained using π_{k+1} and $V^{\pi_k}(\bar{s}')$, where $\bar{s}'$ comes from following policy π_{k+1}. This inequality can further be unrolled infinitely to obtain

$$
\begin{aligned}
V^{\pi_k}(\bar{s}) &\leq \mathsf{E}_{\pi_{k+1}} \sum_{t=0}^{\infty} \gamma^t r(s, a, s') - \delta + \frac{1}{1-\gamma} \left[\frac{1}{1-\gamma} \sum_{t=T+1}^{\infty} (M_t + N_t) + (1 + 2\gamma)\epsilon\right] \\
&= V^{\pi_{k+1}}(\bar{s}),
\end{aligned}
$$

for $\delta = \dfrac{1}{(1-\gamma)^2} \displaystyle\sum_{t=T+1}^{\infty} (M_t + N_t) + \frac{(1+2\gamma)}{1-\gamma}\epsilon.$

Now, suppose $\bar{s} \notin B$. We need to show that this implies (6.5), that is, the Bellman error

should be very small. It is anyway always true that $\mathcal{T}V \geq V$. In the other direction, for any $\bar{s} = (s, x_{0:\infty}) \in \bar{\mathcal{S}}$,

$$
\begin{aligned}
\mathcal{T}V^{\pi_k}(\bar{s}) &= \max_{a \in A} \mathsf{E}_{\substack{s' \sim Q_{x_{0:\infty}}(.|s,a) \\ x' \sim Q_{x_{0:\infty}}(.)}} \left[r(s,a,s') + \gamma V^{\pi_k}\left(s', x', x_{0:\infty}\right) \right] \\
&\leq \max_{a \in A} \mathsf{E}_{\substack{s' \sim Q_{x_{0:\infty}}(.|s,a) \\ x' \sim Q_{x_{0:\infty}}(.)}} \left[r(s,a,s') + \gamma \hat{V}_k\left(s', x', x_{0:T-1}\right) \right] + \gamma\epsilon \qquad (6.6) \\
&\leq \max_{a \in A} \mathsf{E}_{\substack{s' \sim Q_{x_{0:T}, x'_{T+1:\infty}}(.|s,a) \\ x' \sim Q_{x_{0:T}, x'_{T+1:\infty}}(.) \\ x'_{T+1:\infty} \sim \mu_2}} \left[r(s,a,s') + \gamma \hat{V}_k\left(s', x', x_{0:T-1}\right) \right] + \gamma\epsilon + \frac{1}{1-\gamma} \sum_{t=T+1}^{\infty} (M_t + N_t) \\
&= \mathsf{E}_{\substack{s' \sim Q_{x_{0:T}, x'_{T+1:\infty}}(.|s,a) \\ x' \sim Q_{x_{0:T}, x'_{T+1:\infty}}(.) \\ x'_{T+1:\infty} \sim \mu_2 \\ a = \pi_{k+1}(\bar{s})}} \left[r(s,a,s') + \gamma \hat{V}_k\left(s', x', x_{0:T-1}\right) \right] + \gamma\epsilon + \frac{1}{1-\gamma} \sum_{t=T+1}^{\infty} (M_t + N_t) \\
& \qquad\qquad (6.7) \\
&\leq \mathsf{E}_{\substack{s' \sim Q_{x_{0:T}, x'_{T+1:\infty}}(.|s,a) \\ x' \sim Q_{x_{0:T}, x'_{T+1:\infty}}(.) \\ x'_{T+1:\infty} \sim \mu_2 \\ a = \pi_k(\bar{s})}} \left[r(s,a,s') + \gamma \hat{V}_k\left(s', x', x_{0:T-1}\right) \right] + \gamma\epsilon + \frac{1}{1-\gamma} \sum_{t=T+1}^{\infty} (M_t + N_t) + \delta \\
& \qquad\qquad (\text{since } \bar{s} \notin B) \\
&= \mathsf{E}_{\substack{s' \sim Q_{x_{0:T}, x'_{T+1:\infty}}(.|s,a) \\ x' \sim Q_{x_{0:T}, x'_{T+1:\infty}}(.) \\ x'_{T+1:\infty} \sim \mu_2 \\ x''_{T:\infty} \sim \mu_1 \\ a = \pi_k(\bar{s})}} \left[r(s,a,s') + \gamma V^{\pi_k}\left(s', x', x_{0:T-1}, x''_{T:\infty}\right) \right] + \gamma\epsilon \\
& \qquad\qquad + \frac{1}{1-\gamma} \sum_{t=T+1}^{\infty} (M_t + N_t) + \delta \\
&\leq \mathsf{E}_{\substack{s' \sim Q_{x_{0:\infty}}(.|s,a) \\ x' \sim Q_{x_{0:\infty}}(.) \\ a = \pi_k(\bar{s})}} \left[r(s,a,s') + \gamma V^{\pi_k}\left(s', x', x_{0:\infty}\right) \right] + 2\gamma\epsilon + \frac{2}{1-\gamma} \sum_{t=T+1}^{\infty} (M_t + N_t) + \delta \\
&= V^{\pi_k}(\bar{s}) + \delta + 2\gamma\epsilon + \frac{2}{1-\gamma} \sum_{t=T+1}^{\infty} (M_t + N_t).
\end{aligned}
$$

Inequality (6.6) above is due to the definition of $\hat{V}_k$, and (6.7) is due to the definition of

π_{k+1}. Therefore,

$$
\begin{aligned}
|\mathcal{T}V^{\pi_k}(\bar{s}) - V^{\pi_k}(\bar{s})| &\leq \delta + 2\gamma\epsilon + \frac{2}{1-\gamma}\sum_{t=T+1}^{\infty}(M_t + N_t) \\
&= \frac{1}{(1-\gamma)^2}\sum_{t=T+1}^{\infty}(M_t + N_t) + \frac{(1+2\gamma)}{1-\gamma}\epsilon + 2\gamma\epsilon + \frac{2}{1-\gamma}\sum_{t=T+1}^{\infty}(M_t + N_t) \\
&= \sum_{t=T+1}^{\infty}(M_t + N_t)\left[\frac{3-2\gamma+4\gamma}{(1-\gamma)^2} + \frac{2(1+2\gamma)}{(1-\gamma)^3}\right] \\
&= \sum_{t=T+1}^{\infty}(M_t + N_t)\left[\frac{3+2\gamma}{(1-\gamma)^2} + \frac{2(1+2\gamma)}{(1-\gamma)^3}\right] \\
&\leq \frac{11}{(1-\gamma)^3}\sum_{t=T+1}^{\infty}(M_t + N_t).
\end{aligned}
$$

□

Proof of Lemma 6. Letting $\delta = 0$ in the first part of the above proof gives us the Lemma for which there is no assumption on the extent of policy improvement.

$$
\begin{aligned}
V^{\pi_k}(\bar{s}) &\leq V^{\pi_{k+1}}(\bar{s}) + \frac{1}{1-\gamma}\left[\frac{1}{1-\gamma}\sum_{t=T+1}^{\infty}(M_t + N_t) + (1+2\gamma)\epsilon\right] \\
&= V^{\pi_{k+1}}(\bar{s}) + \left[\frac{1}{(1-\gamma)^2} + \frac{2(1+2\gamma)}{(1-\gamma)^3}\right]\sum_{t=T+1}^{\infty}(M_t + N_t) \\
&= V^{\pi_{k+1}}(\bar{s}) + \frac{3+3\gamma}{(1-\gamma)^3}\sum_{t=T+1}^{\infty}(M_t + N_t).
\end{aligned}
$$

□

In Theorem 6, in regions of the augmented state space without guaranteed policy improvement, the Bellman error depends on the approximation error due to ignoring old events. It is a cumulative effect of all events older than T on the current events, as well as the state transition kernel. Specifically, it is proportional to $\sum_{t=T+1}^{\infty}(M_t + N_t)$, where T is the time window beyond which events are discarded, M_t is an upper bound on the total variation disturbance on the state transition kernel caused by an event older than t time steps, and similarly N_t bounds the total variation effect due to old events on the event process itself. The faster the decay of exogenous event influence, the smaller the truncation error due to non-stationarity.

Proof of Corollary 7. From (6.2), the extra error introduced is proportional to

$$\begin{aligned}
\sum_{t=T+1}^{\infty} (M_t + N_t) &\leq \sum_{t=T+1}^{\infty} \left(\bar{c} e^{-\bar{\lambda} t} + c_\alpha e^{-\lambda_\alpha t} + \mathsf{erf}\left(\frac{c_\beta b e^{-\lambda_\beta t}}{2\sqrt{2}} \right) \right) \\
&\leq \sum_{t=T+1}^{\infty} \left(\bar{c} e^{-\bar{\lambda} t} + c_\alpha e^{-\lambda_\alpha t} + \sqrt{1 - \exp\left(\frac{-c_\beta^2 b^2}{2\pi} e^{-2\lambda_\beta t} \right)} \right) \\
&\leq \sum_{t=T+1}^{\infty} \left(\bar{c} e^{-\bar{\lambda} t} + c_\alpha e^{-\lambda_\alpha t} + \frac{c_\beta b}{\sqrt{2\pi}} e^{-\lambda_\beta t} \right) \\
&\leq \int_{t=T}^{\infty} \left(\bar{c} e^{-\bar{\lambda} t} + c_\alpha e^{-\lambda_\alpha t} + \frac{c_\beta b}{\sqrt{2\pi}} e^{-\lambda_\beta t} \right) \mathrm{d}t \\
&= \frac{\bar{c}}{\bar{\lambda}} e^{-\bar{\lambda} T} + \frac{c_\alpha}{\lambda_\alpha} e^{-\lambda_\alpha T} + \frac{c_\beta b}{\sqrt{2\pi}\lambda_\beta} e^{-\lambda_\beta T}.
\end{aligned}$$

□

6.5 Sample Complexity

While the algorithm given in Section 6.4 defines the value function and its update, in practice, it has to be learned using samples obtained from the environment and is generally approximated from a chosen class of functions. In this section, we analyze the sample complexity of policy evaluation by pathwise Least-Squares Temporal Difference (LSTD) (Lazaric et al., 2012), which uses samples from a single sample path induced by the policy, and give error bounds on the evaluation error for the case of a linear function space and some regularity assumptions on the MDP.

In contrast to Lazaric et al. (2012), our work considers an MDP with an infinite-dimensional augmented state space, a stochastic reward function, and an additional error due to the use of tractable policies and value functions on a finite-dimensional domain. This results in additional sample complexity even for evaluating a given policy, depending on the extent to which the exogenous events affect the evolution of the states. In this section, we quantify the precise nature of this error.

Consider a function class $\mathcal{F}$ of functions that are to be used to approximate the value function on the augmented MDP, $\mathcal{F} \subset \left\{ f : \bar{\mathcal{S}} \to \mathbb{R} \right\}$. Since the domain of the functions in this class is infinite-dimensional, this can be further broken down into two approximations for dealing with these functions practically: a truncation operation to make the domain finite-dimensional, and then another class of functions that are then used to cover possible functions on this domain,

i.e,

$$\mathcal{F} = \left\{ f \; o \; f^{(T)}_{\text{trunc}} : f \in \mathcal{F}^{(T)} \right\}, \text{ where } f^{(T)}_{\text{trunc}} : \bar{\mathcal{S}} \to \mathcal{S} \times \prod_{t=0}^{T} \mathcal{X}.$$

$f^{(T)}_{\text{trunc}}$ is a fixed truncation function that removes extra events that are older than T time steps while preserving the current state and new events, and $\mathcal{F}^{(T)}$ is some function class of functions defined on this new truncated domain. For a fixed T, finding the best function in $\mathcal{F}$ is the same as finding the best function in $\mathcal{F}^{(T)}$.

Finally, we wish to break down the error between the true value function and the learned truncated value function in terms of the errors introduced due to these two operations, as well as the stochasticity of the samples used to learn the approximate function. The aim is to provide a bound to $\left\| V - \widetilde{V} \right\|_\rho$, where V is the true value function on $\bar{\mathcal{S}}$, $\widetilde{V}$ is the learned value function in $\mathcal{F}$, ρ is some distribution over $\bar{\mathcal{S}}$, and the norm $\|\cdot\|_\rho$ is the $l^2(\rho)$-norm, which is the expected l^2 norm of the value of the function w.r.t the measure ρ, i.e,

$$\|f\|^2_\rho = \int_{\bar{\mathcal{S}}} f(x)^2 \; \rho(\mathrm{d}x).$$

While the $f^{(T)}_{\text{trunc}}$ operator just ignores old events, we also define a projection operator Π^{trunc} onto the function space defined on the truncated domain, as

$$\Pi^{\text{trunc}} V = \underset{f : \mathcal{S} \times \prod_{t=0}^{T} \mathcal{X} \to \mathbb{R}}{\operatorname{argmin}} \|f - V\|_\rho .$$

which finds the best approximation that does not depend on the old events, where 'best' is defined in expectation with respect to ρ.

A standard approximating function space considered is an arbitrary finite-dimensional function space. Specifically, assume that there are d basis functions $\varphi_i : \mathcal{S} \times \mathcal{X}^{T+1} \to \mathbb{R}$ that linearly span $\mathcal{F}^{(T)}$ and are bounded by L, that is, for each $f \in \mathcal{F}^{(T)}$, there exist $\alpha \in \mathbb{R}^d$ such that $f = \sum_{i \in [d]} \alpha_i \varphi_i$. These basis functions together form a feature representation function $\phi : \mathcal{S} \times \mathcal{X}^{T+1} \to \mathbb{R}^d$, defined by $\phi(x) = (\varphi_1(x), \ldots, \varphi_d(x))$.

6.5.1 Approximate Policy Evaluation

The best approximate value function $\widetilde{V}$ is that function in $\mathcal{F}$ (or equivalently $\mathcal{F}^{(T)}$) that minimizes the above-expected norm difference.

Suppose we are given just N samples $\bar{s}_1, \ldots, \bar{s}_N$ from a Markov chain induced by policy π in the augmented MDP $\mathcal{M}_X$, and corresponding rewards $r_1, \ldots, r_N$. We minimize the empirical

norm difference at these points, defined as

$$\|f\|_N = \left(\frac{1}{N}\sum_{t=1}^{N} f(x_t)^2\right)^{\frac{1}{2}}.$$

The above function defines a norm, along with a corresponding inner product, in a new inner product space $\mathcal{F}_N$, the space of all (N) values of the functions at the given sample points. This can be considered as a subset of $\mathbb{R}^N$ with the obvious inner product.

$$\mathcal{F}_N = \{(f(\bar{s}_1), \ldots, f(\bar{s}_N)) : f \in \mathcal{F}\} = \left\{\left(f\left(s_1^{(T)}\right), \ldots, f\left(s_N^{(T)}\right)\right) : f \in \mathcal{F}^{(T)}\right\} = \left\{\Phi\alpha : \alpha \in \mathbb{R}^d\right\},$$

where $\Phi = \left(\phi\left(\bar{s}_1^{(T)}\right), \ldots, \phi\left(\bar{s}_N^{(T)}\right)\right)_{N\times d}$ and $\bar{s}_i^{(T)} = f_{\text{trunc}}^{(T)}(\bar{s}_i)$.

In pathwise LSTD, given a sequence of states $\bar{s_1}, \ldots, \bar{s_n}$ and corresponding rewards $r_1, \ldots, r_n$ obtained by following policy π, the value function is approximated as

$$\hat{V} = \sum_{i\in[d]} \hat{\alpha}_i\varphi_i, \text{ for } \hat{\alpha} = \left[\Phi^\intercal\left(I - \gamma\hat{P}\right)\right]^+ \Phi^\intercal r,$$

where $\Phi = \left(\phi\left(\bar{s}_1^{(T)}\right), \ldots, \phi\left(\bar{s}_N^{(T)}\right)\right)_{N\times d}$ is the feature matrix, with $\bar{s}_i^{(T)} = f_{\text{trunc}}^{(T)}(\bar{s}_i)$, and $\hat{P}_{n\times n} = (\mathbb{I}\{j = i+1\})_{i,j}$ is the pathwise empirical transition matrix.

For determining the expected error of this estimate, the intermediate task is to bound the difference between this and the true solution on these given data points, i.e., $\|V - \hat{V}\|_N$, as a function of the number of data points and the complexity of the function class under consideration.

Lemma 8. *Let $v = (V(\bar{s}_t))_{t\in[N]}$ and $\hat{v} = \left(\hat{V}(\bar{s}_t)\right)_{t\in[N]}$. Then, with probability at least $1-\delta$,*

$$\|v - \hat{v}\|_N \le \frac{1}{\sqrt{1-\gamma^2}}\left\|v - \hat{\Pi}v\right\|_N + \frac{L}{(1-\gamma)^2}\sqrt{\frac{d}{\nu_N}}\left(\sqrt{\frac{2\log(2d/\delta)}{N}} + \frac{1}{N}\right),$$

where N is the number of samples used, $\hat{\Pi}$ is the projection onto $\mathcal{F}_N$, d is the dimensionality of the linear function space considered, L is the upper bound of the basis functions, and ν_N is the smallest positive eigenvalue of the Gram matrix $\Phi^\intercal\Phi$.

Proof. See Section 6.A.3. □

The above inequality is almost identical to the one in Lazaric et al. (2012), the reason being that we are still operating in the range space of the basis functions restricted to a finite set

of states, and so the structure of the MDP in our setting does not affect the analysis. Since the proof follows a similar path, we have included in the Appendix those steps that differ from the original proof, which are mainly due to the noise terms taken in the martingale difference sequence. In our setting, these noise terms also include stochasticity due to the reward in addition to the transition function, leading to a difference in the final expression.

While this final inequality is quite similar, we shall see that when considering the expected error in the augmented state space, we can bring out the properties of our setting by splitting the expected error in terms of error due to function truncation and the inherent error due to linear function approximation.

6.5.2 Expected Error

The above inequality bounds the average error of the estimated value function evaluated only at those points obtained as a sample from the environment. It is desirable to study the error in the approximate value function learned as an expectation over the states with respect to some distribution ρ. Generally, ρ is taken to be the stationary distribution of the Markov chain induced in the MDP by the policy π. To obtain such a result, additional assumptions are imposed on this Markov chain, such as the speed of convergence to its stationary distribution, etc.

For linear function spaces, assuming that the policy induces a β-mixing Markov chain gives the following generalization bound for policy evaluation.

Theorem 7. *Assume that the policy π induces on the MDP $\mathcal{M}_X$ a β-mixing Markov chain with parameters $\bar{\beta}$, b, κ and stationary distribution ρ. Let $\bar{s}_1, \ldots, \bar{s}_{N+\widetilde{N}}$ be a sample path obtained using policy π. Suppose the first $\widetilde{N}$ samples are discarded for Markov chain burn-in, and the remaining N samples are used to compute the truncated least-squares path-wise estimate $\widetilde{V}$ of the true value function V. Let ν be a lower bound on the eigenvalues of the sample-based Gram matrix that holds with probability $1 - \delta/4$. Then, provided the number of discarded samples is $\tilde{N} = \left(\frac{1}{b}\log\frac{2e\bar{\beta}n}{\delta}\right)^{\frac{1}{\kappa}}$, with probability at least $1 - \delta$,*

$$\left\|\widetilde{V} - V\right\|_\rho \leq \frac{4\sqrt{2}}{\sqrt{1-\gamma^2}}\left(\frac{3}{(1-\gamma^2)}\sum_{t=T+1}^{\infty}(M_t + N_t) + \left\|\Pi^{\text{trunc}}V - \Pi V\right\|_\rho\right)$$
$$+ \frac{2L}{(1-\gamma)^2}\sqrt{\frac{d}{\nu}}\left(\sqrt{\frac{2\log(8d/\delta)}{N}} + \frac{1}{N}\right) + \epsilon_1(N, \bar{\beta}, b, \kappa, d, \delta) + 2\sqrt{2}\epsilon_2(N, \bar{\beta}, b, \kappa, \delta),$$

$$\textit{where } \epsilon_1(N,d,\delta) = \frac{24}{1-\gamma}\sqrt{\frac{2\Lambda_1(N,\bar{\beta},d,\delta/4)}{N}\max\left\{\frac{\Lambda_1(N,\bar{\beta},d,\delta/4)}{b},1\right\}^{\frac{1}{\kappa}}},$$

$$\epsilon_2(N,\delta) = 12\left(\frac{1}{1-\gamma}+L\left\|\alpha^*\right\|\right)\sqrt{\frac{2\Lambda_2(N,\bar{\beta},\delta/4)}{N}\max\left\{\frac{\Lambda_2(N,\bar{\beta},\delta/4)}{b},1\right\}^{\frac{1}{\kappa}}},$$

V is the true value function, $\widetilde{V}$ is the learned value function clipped at $\frac{1}{1-\gamma}$, $\Pi^{\text{trunc}}V$ is the best possible value function on the truncated state space, ΠV is the best approximating value function in the linear function space considered, M_t and N_t are the upper bounds on the total variation due to exogenous events older than t time steps induced in the transition dynamics and event mark distribution respectively, $\Lambda_1(N,d,\delta)$ and $\Lambda_2(N,\delta)$ are as defined in Lemma 9 in Section 6.A.3, and α^* is such that $\Pi V = \sum_i \alpha_i^*\varphi_i$.

This result breaks down the overall error into individual components, viz. the approximation error due to the non-stationarity, the inherent error due to linear function approximation, the error due to stochasticity of the samples that reduces with the number of samples, and other errors due to mixing of the Markov chain, the dimensionality of the function space, etc.

The effect of the non-stationarity is reflected in the first term proportional to $\sum_t(M_t+N_t)$ for a general exogenous process whose events have an effect that decays as M_t and N_t on the state transition and next event distribution, respectively. For a specific temporal process, such as the discrete Hawkes process described in Corollary 7, this term is replaced by its corresponding upper bound $\left(\frac{\bar{c}}{\bar{\lambda}}e^{-\bar{\lambda}T}+\frac{c_\alpha}{\lambda_\alpha}e^{-\lambda_\alpha T}+\frac{c_\beta}{\sqrt{2\pi}\lambda_\beta}e^{-\lambda_\beta T}\right)$ that depends on the parameters of the process.

The detailed background for the conditions on the MDP under which the results hold, the exact expressions for Λ_1 and Λ_2, as well as supporting generalization Lemmas needed for the proof, are given in Section 6.A.3.

Proof of Theorem 7. Following proof of Theorem 5 in Lazaric et al. (2012),

$$2\left\|\hat{V}-V\right\|_N \geq 2\left\|\widetilde{V}-V\right\|_N \geq \left\|\widetilde{V}-V\right\|_\rho - \epsilon_1, \tag{6.8}$$

where the first inequality results from truncation, and the second inequality is a generalization Lemma for Markov chains stated in Section 6.A.3, and so

$$\left\|\widetilde{V}-V\right\|_\rho \leq 2\left\|\hat{V}-V\right\|_N + \epsilon_1 \tag{6.9}$$

$$\leq \frac{2}{\sqrt{1-\gamma^2}}\left\|V-\hat{\Pi}V\right\|_N + \frac{2L}{(1-\gamma)^2}\sqrt{\frac{d}{\nu}}\left(\sqrt{\frac{2\log(2d/\delta')}{N}}+\frac{1}{N}\right)+\epsilon_1 \tag{6.10}$$

$$\leq \frac{2}{\sqrt{1-\gamma^2}} \|V - \Pi V\|_N + \frac{2L}{(1-\gamma)^2}\sqrt{\frac{d}{\nu}}\left(\sqrt{\frac{2\log(2d/\delta')}{N}} + \frac{1}{N}\right) + \epsilon_1 \tag{6.11}$$

$$\leq \frac{4\sqrt{2}}{\sqrt{1-\gamma^2}} \|V - \Pi V\|_\rho + \frac{2L}{(1-\gamma)^2}\sqrt{\frac{d}{\nu}}\left(\sqrt{\frac{2\log(2d/\delta')}{N}} + \frac{1}{N}\right) + \epsilon_1 + 2\sqrt{2}\epsilon_2 \tag{6.12}$$

$$\leq \frac{4\sqrt{2}}{\sqrt{1-\gamma^2}} \left(\left\|V - \Pi^{\text{trunc}} V\right\|_\rho + \left\|\Pi^{\text{trunc}} V - \Pi V\right\|_\rho\right) + \frac{2L}{(1-\gamma)^2}\sqrt{\frac{d}{\nu}}\left(\sqrt{\frac{2\log(2d/\delta')}{N}} + \frac{1}{N}\right) + \epsilon_1 + 2\sqrt{2}\epsilon_2 \tag{6.13}$$

$$\leq \frac{4\sqrt{2}}{\sqrt{1-\gamma^2}} \left(\|V - V^\epsilon(s, x_{0:T}, \bar{0})\|_\rho + \left\|\Pi^{\text{trunc}} V - \Pi V\right\|_\rho\right) + \frac{2L}{(1-\gamma)^2}\sqrt{\frac{d}{\nu}}\left(\sqrt{\frac{2\log(2d/\delta')}{N}} + \frac{1}{N}\right) + \epsilon_1 + 2\sqrt{2}\epsilon_2 \tag{6.14}$$

$$\leq \frac{4\sqrt{2}}{\sqrt{1-\gamma^2}} \left(\|V - V^\epsilon\|_\rho + \|V^\epsilon - V^\epsilon(s, x_{0:T}, \bar{0})\|_\rho + \left\|\Pi^{\text{trunc}} V - \Pi V\right\|_\rho\right) + \frac{2L}{(1-\gamma)^2}\sqrt{\frac{d}{\nu}}\left(\sqrt{\frac{2\log(2d/\delta')}{N}} + \frac{1}{N}\right) + \epsilon_1 + 2\sqrt{2}\epsilon_2 \tag{6.15}$$

$$\leq \frac{4\sqrt{2}}{\sqrt{1-\gamma^2}} \left(\frac{3}{(1-\gamma^2)} \sum_{t=T+1}^{\infty} (M_t + N_t) + \left\|\Pi^{\text{trunc}} V - \Pi V\right\|_\rho\right) + \frac{2L}{(1-\gamma)^2}\sqrt{\frac{d}{\nu}}\left(\sqrt{\frac{2\log(2d/\delta')}{N}} + \frac{1}{N}\right) + \epsilon_1 + 2\sqrt{2}\epsilon_2. \tag{6.16}$$

Inequality (6.10) is due to Lemma 8, (6.11) is by the definition of $\hat{\Pi}$, and (6.12) is the generalization bound similar to (6.8) in the opposite direction for upper bounding the empirical error in terms of the expected error. (6.13) is just the triangle inequality.

(6.14) is by definition of the Π^{trunc} operator, where V^ϵ is the 'truncated' value function of an ϵ-suboptimal policy that depends only on the past T events. This is guaranteed from the proof of Theorem 5 for $\epsilon = \frac{2}{(1-\gamma)^2}\sum_{t=T+1}^{\infty}(M_t + N_t)$. (6.15) is due to the triangle inequality by adding and subtracting the actual value function V^ϵ of the ϵ-suboptimal policy. The final inequality (6.16) is due to Lemma 4 and the value of ϵ.

Both these generalization bounds happen with probability at least $1 - \delta'$ each. Further, with the lower bound ν on ν_N holding with probability at least $1 - \delta'$, the final bound holds with probability at least $1 - \delta = 1 - 4\delta'$. □

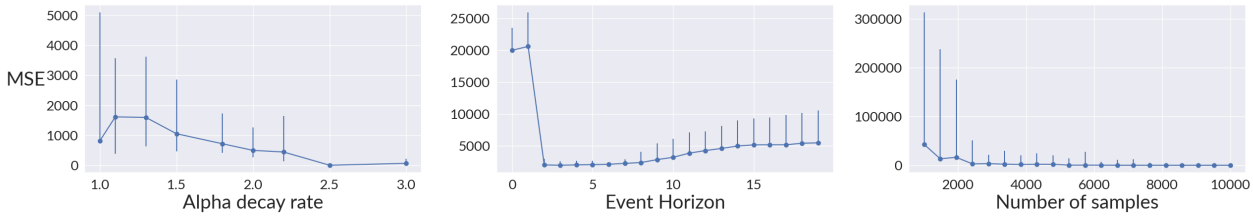


Figure 6.3: Performance of pathwise LSTD. The plots show the dependence of the Mean Squared Error of the value function learned using pathwise LSTD as a function of the rate of decay of the exogenous Hawkes process, the horizon of past events considered, and the number of samples used.

The generalization terms due to linear function approximation using Markov samples remain essentially the same as in Lazaric et al. (2012). The main difference is the first extra term in the error that is induced due to truncating the augmented state space by discarding the features of old events. The amount of error introduced due to this approximation depends on the amount of influence exerted by old events on the current events, as well as the state transition kernel.

The results required for the proof can be taken from Lazaric et al. (2012) without any modifications because, even if some of the results, specifically the results on the bound on the covering numbers for linear function spaces taken from Györfi et al. (2002), require the domain of the basis functions to be finite-dimensional, the analysis takes place in the co-domain and so can be directly applied to our setting. We state the complete theorem along with the Lemmas in Section 6.A.3.

6.6 Experiments

Policy Evaluation We conducted experiments for analyzing policy evaluation in a non-stationary variant of the classic Pendulum-v1 environment in the control suite of OpenAI Gym. The task consists of applying torque to a pendulum to swing it and keep it upright. We considered a fixed neural-network policy that has been pre-trained partially using DDPG (Lillicrap et al., 2016). We then used pathwise LSTD with linear function approximation for evaluating this policy as described in Section 6.5. The features considered are the standard cosine and sine of the angle and the angular velocity, with additional features being the angle itself along with the squares of all these features, making the state 8 dimensional.

For the non-stationarity, we consider the discrete-time Gaussian marked Hawkes process described in Section 6.2.1. The intensity due to an event is decayed as $\alpha = e^{-\lambda_\alpha t}$ with $\lambda_\alpha = 1.0$, and the effect on the event mark decays as $\beta_t = 1/(1+t^2)$. The events are added to the torque applied to the pendulum. Figure 6.3 shows the expected error of the learned value function

as a function of the number of samples, the event horizon T, and the rate λ_α of decay of the Hawkes process.

For a fixed event horizon 5, the error reduces as the number of samples increases, which is quite intuitive. For a fixed number of samples $10,000$, the average error initially reduces as the event horizon increases, corresponding to the first term in the sample complexity of the total variation induced by events older than the event horizon. After a certain stage, increasing the event horizon results in a gradual increase in the error. This is because the events older than the horizon now contribute a negligible influence on the current dynamics, while the dimensionality of the features is increasing due to the inclusion of more past events in the features, resulting in a slightly greater expected error.

The first plot shows the dependence on λ_α, the rate of decay of α_t. Faster decay results in lesser non-stationarity and lower approximation error due to the first term in the sample complexity upper bound.

We conducted 20 trials and plotted the median of the Mean Squared Errors for the experiments with respect to decay rate and number of samples. The error bars indicate the range of 40-60 percentile versus the decay rate and 20-80 percentile for the other two.

These results empirically demonstrate the sample complexity bound for policy evaluation with linear function approximation in Theorem 7. The average error decreases with increasing N and decreasing N_t.

Increasing the event horizon T increases d while reducing $\sum_{t=T+1}^{\infty}\{M_t, N_t\}$, resulting in a trade-off between the corresponding two terms in the bound, with the minimal error being achieved for $T = 2$.

Policy Deployment In this chapter, we have considered the setting where the environment containing an agent is perturbed due to the presence of exogenous factors. Suppose there is a trained policy available to the agent that is optimal in the absence of these external influences. Now, we wish to deploy this agent in a non-stationary environment where these external events do affect the MDP dynamics. In this case, it would be more efficient to modify the existing policy to deal with the non-stationarity directly than to train a new policy from the ground up.

We consider a simple model-based planning strategy, wherein the agent learns the dynamics model and uses it to simulate a few trajectories using the current policy. More precisely, given a pretrained policy π, when at augmented state $\bar{s} \in \mathcal{S}$ during deployment, d actions $a_1, \ldots, a_d$ are sampled by perturbing the action $a = \pi(\bar{s})$. From each of these actions a_i, a trajectory $(s, a_i, s_{i,1}, a_{i,1}, \ldots, s_{i,H}, a_{i,H})$ is obtained and its corresponding sum of rewards $r_i = \sum_{t=1}^{H} r(s_{i,1}, a_{i,1})$ is calculated. The action a_i that gave rise to the highest reward r_i is chosen and taken in the environment. This process is repeated at each time step during deployment.

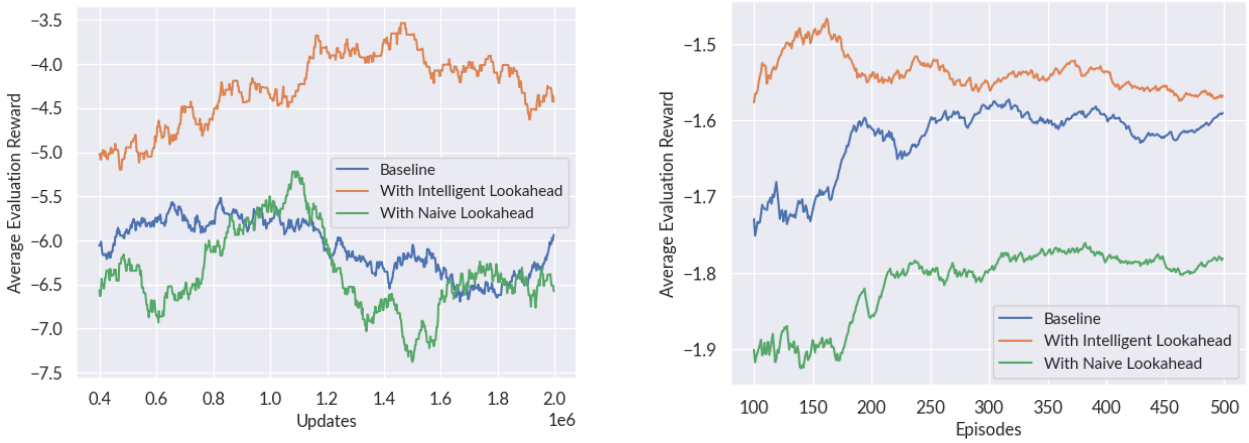


Figure 6.4: Results on non-stationary Pendulum and Point Maze environments. Learning state dynamics and exogenous events separately and planning intelligently outperforms a stationary policy or a model-based policy that tries to learn the dynamics model in the augmented state space directly.

This procedure can be done in two different ways. A naive way to do this is to learn a dynamics model $\hat{T} : \bar{\mathcal{S}} \times \mathcal{A} \to \Delta\bar{\mathcal{S}}$ directly in the augmented state space and use it to simulate trajectories. However, a more intelligent method to this is to learn the event process separately, and use it to simulate events $e_{i,t}$ that are fed into a dynamic model $\bar{T} : \bar{\mathcal{S}} \times \mathcal{A} \to \mathcal{S}$ that only predicts the states. Figure 6.4 depicts the results obtained using these two methods, along with those of a baseline that is just the stationary policy learned in the stationary version of the environment. The first plot shows the results on the non-stationary pendulum task described previously. The second plot is for a non-stationary version of the Gymnasium Point Maze environment (Fu et al., 2020), in which a 2DoF ball is force-actuated in the Cartesian x-y directions to reach a random specified target goal in a closed, continuous 2D maze. The non-stationarity is induced by independent Hawkes processes as described previously that occur at 5 randomly sampled and fixed points in the maze, each of which exerts a pulling force on the agent towards itself that is inversely proportional to the squared distance. From the plots, it is evident that an intelligent model-based look-ahead policy that mimics the structure of the non-stationarity performs better than the naive baselines.

The algorithms theoretically analyzed in the previous sections are extensions to standard reinforcement learning algorithms that operate in the augmented state space instead. This is a valid strategy in many cases, as observed in the policy evaluation experiments described above. However, such a strategy ignores the structure of the non-stationarity, sacrificing performance

for generality. When there is additional knowledge on the nature of the exogenous event process, a valid question to ask is whether using this knowledge helps solve the problem better. This experiment showed that, for the case of deploying a stationary policy in a non-stationary environment by using model-based planning, taking into account the nature of the event process and incorporating it intelligently with current algorithms will lead to better results than ignoring the structure of the non-stationarity and solely operating in the augmented state space.

This discrepancy is due to the nature of neural-network function approximation primarily used in practical scenarios. The approximate policy improvement step in Algorithm 10 attempts to maximize the one-step returns obtained from each augmented state $\bar{s}$, which requires knowledge of the dynamics model. When this is unknown, an approximate model is used, which can be constructed in two ways, as described above. Standard neural network dynamics models used in literature are ill-equipped to model event processes, compelling one to model them separately from the original state $s \in \mathcal{S}$ when possible. Further research needs to be done on how precisely to model and incorporate event processes in reinforcement learning policies in a way that utilizes the structure of the non-stationarity and the nature of the induced perturbations.

6.7 Related Work

Many previous studies on non-stationarity primarily provide practical solutions that lack theoretical backing. Here, we do not discuss such works but try to position our work among the existing theoretical results.

Lecarpentier and Rachelson (2019) considers a finite horizon MDP in which the transition dynamics and reward function evolve such that they are Lipschitz continuous with respect to time. Cheung et al. (2020) considers such a setting in a finite MDP where the amount of total change in the environment, referred to as the variation budget, is known to the agent, proposing an optimistic Value Iteration-based algorithm with guarantees on the upper bound of its dynamic regret. Wei and Luo (2021) proposes a black-box reduction that turns any suitable algorithm with optimal performance in a near-stationary environment into an algorithm that achieves optimal dynamic regret in a non-stationary environment. Guo et al. (2024) studies average reward non-stationary MDPs in Borel spaces and proposes a rolling-horizon algorithm to obtain arbitrarily almost-optimal solutions based on a suitably changing horizon. At each stage, a finite horizon problem starting from that stage is solved to obtain the decision rule at that stage. In contrast, our algorithm considers a horizon of past external events instead for choosing the current action. Further, along with standard regularity conditions of compactness, continuity, etc, Condition 1 in Guo et al. (2024) contains global conditions on the sequence of transition kernels at each stage of the MDP to guarantee the existence of optimal policies and

solutions to the Average Optimality Equation. In contrast, our assumptions (A2) and (A3) are more local in nature, with bounds on perturbations of the transition kernel.

Hadoux et al. (2014) formalized a class of piecewise stationary MDPs called Hidden-Semi-Markov-Model MDPs, where the transition from one hidden stationary MDP mode to another is semi-Markov in nature. To solve this problem, they adapt the Partially Observable Monte Carlo Planning (POMCP) algorithm used for solving Partially Observable Markov Decision Processes (POMDPs).

Some algorithms assume additional structure on the nature of non-stationarity. Feng et al. (2022) propose Factored Non-stationary MDPs (FN-MDPs), where the transition dynamics are defined in terms of a Dynamic Bayesian Network over the states, actions, rewards, and latent change factors that cause non-stationarity. Here, both the states and the latent change factors have transition dynamics obeying this causal graph, and the latent factors evolve in a Markov fashion. They propose using a Variational Auto-Encoder (VAE) framework to learn transition dynamics and present results in MuJoCo and robotic manipulation tasks.

A factorization setting with a resemblance to ours is the Exogenous MDP (Efroni et al., 2022), wherein the environment state is divided into two parts, an endogenous state which is controllable by the agent's actions, and an exogenous state, which evolves on its own and does not affect the agent's reward. The precise way in which this division exists is unknown to the agent. While this may seem similar to our formulation of states and external events, in our case, the external events do influence the dynamics of the state and, thereby, the rewards obtained by the agent. Further, Efroni et al. (2022) only consider the tabular setting and propose algorithms that can only deal with finite horizon episodes.

A similar decomposition of the state space into endogenous and exogenous sub-spaces was studied in Dietterich et al. (2018), where it is assumed that the exogenous states evolve in a Markov fashion and the reward function can be additively decomposed into an exogenous reward and an endogenous reward. This work proposed algorithms to determine the exogenous component of the state as projections that maximize the partial correlation coefficient as a proxy for the mutual information between the exogenous next state and the endogenous current state and action conditioned on the exogenous current state.

While previous studies like Efroni et al. (2022) and Dietterich et al. (2018) focus on the decomposition of states into endogenous and exogenous components, our work addresses a fundamentally different problem: the impact of integrating exogenous event information into reinforcement learning.

Changing dynamics of environments based on externally specified "contexts" has also been studied under Contextual MDPs (Hallak et al., 2015). However, these studies consider contexts

that are constant for each episode and have a finite number of possible values. The aforementioned work proposes the CECE algorithm that learns a set of policies corresponding to all the contexts and determines the latent context for an episode (if unknown) to choose the policy corresponding to the current context.

More specifically, CECE clusters an initial set of trajectories into groups corresponding to a fixed set of latent contexts. For each new episode, a partial realization is used to classify it into one of these clusters and to use a model learned on that cluster to exploit the rest of the trajectory. This algorithm cannot handle the setting considered by us because each new time step results in a change in the context, and knowledge of a latent context for the current time step does not guarantee the same context for the rest of the episode, except in a trivial special case.

Modi et al. (2018) consider a relatively closer setting to ours, where the context is known and possibly continuous, with their algorithm being more amenable to being adapted to handle dynamic contexts. However, it works by keeping count of the number of occurrences of (s, a) in each of a set of representative contexts to learn a set of explicit models, which is only possible in the setting of finite state and action spaces.

Tennenholtz et al. (2023) extend Contextual MDPs to handle dynamic contexts that are known to the agent, change at each time step, and influence the state transition distribution. However, this work differs from ours in several significant aspects. This work considers the contexts at each time step to depend on the history of states, actions, and contexts, while the next state depends only on the current state, action, and context. This means causality exists in both directions, from states to contexts and vice versa. Whereas in our work, the contexts are exogenous events that cannot be controlled by the agent, and the causality flows only in one direction, from contexts to states. Further, their work considers finite state spaces, with the analysis explicitly geared towards such spaces and bounds that depend on the size of the state and action sets.

In Tennenholtz et al. (2023), the context space is a fixed finite set of M vectors, and a latent map is learned from the (state, action, context) space to $\mathbb{R}^M$, with the range of the map still being a finite albeit exponentially increasing set. This work also considers a finite-horizon MDP. This relatively simple setting allows one to tractably find the latent map and learn an optimal policy. In this sense, our setting is much more general and challenging. In addition to this, the work of Tennenholtz et al. (2023) presents regret analysis, while our study focuses on the convergence results of algorithms.

Our analysis addresses the non-stationarity induced in the MDP due to the influence of non-Markovian events, necessitating the learning of policies that are contingent upon both

the current state and a history of prior events. Although this approach bears resemblance to the strategies employed in Partially Observable MDPs (POMDPs) (Sunberg and Kochenderfer, 2018), the underlying rationale diverges significantly. In POMDPs, histories of observations are utilized despite the Markovian nature of state transition and observation functions, primarily to mitigate the agent's inability to perceive the true state. Conversely, in our framework, both the state and external events are fully observable, and the necessity for an augmented space that incorporates event histories arises from the intrinsic non-Markovian characteristics of the non-stationary environment. Moreover, the causal framework of our scenario is distinctly different. The events exhibit non-Markovian properties, and the state at any given time directly affects the distribution of the subsequent state, regardless of the knowledge of the event. In POMDPs, while beliefs and sufficient statistics are contingent upon a history of observations, the states themselves maintain a Markovian nature, leading to observations that are causally independent.

Our work makes significant inroads into understanding how characteristics of exogenous event processes affect the tractability of the problem and the convergence and sample complexity of RL algorithms. To the best of our knowledge, there are no existing works in the literature that offer these findings.

6.8 Outlook

This work lays the groundwork for understanding sequential decision-making problems under the influence of external temporal events and provides several theoretical insights in a reinforcement learning framework. The results are established in a very general setting and rely on learning functions in an augmented state space that could be exponentially bigger in the presence of long-term dependencies due to external events. To mitigate this, one needs to develop provable approximate methods for reducing the size of the state space. Due to the "factorization" of the environment into Markovian states and non-Markovian events that impact these states, compressing the histories of events into latent features that are independent of the states may provide a strategic advantage. This approach could facilitate the learning of representations of the expanded states by developing a distinct event distribution model that can be scaled up, as it does not rely on the agent, alongside a separate state transition model that necessitates agent interaction with the environment. A compressed representation of events derived from extensive event data can subsequently be utilized to learn a feasible state transition kernel. This can be achieved either implicitly through function approximators such as recurrent neural networks (or LSTMs) or explicitly by learning compressed state representations using mutual information bottlenecks, etc. While these are practical methods, the theoretical analysis of such learned compressed state representations remains an open problem.

The analysis of sample complexity in Section 6.5 hinges on generalizing the sample error of policy evaluation to an expected error over the entire state space. Such generalization generally requires assumptions on the distribution over states. In our result, the assumption pertains to the rate of convergence of the Markov chain induced from the augmented MDP by the policy under evaluation, a notion frequently encountered in reinforcement learning literature. However, in our case, the specific structure of the state space–specifically, the separation of the augmented state space into states and events with different characteristics–may necessitate alternate assumptions. For instance, while the recurrence and aperiodicity of the Markov chain may not hold, it may be reasonable to assume stationarity solely for the event sequence. Due to this, the generalizability of the value function in our setting can be analyzed in a more specific way that takes into account this exogenous structure.

The policy improvement step involves optimizing over the action space based on the expected one-step reward and the value function at the next time step, which requires knowledge of the model. Thus, in the case of an unknown transition and reward model, it is important to investigate the sample complexity of learning the models from samples and how this affects the performance of the resultant policy.

6.A Appendix: Details of Lemmas and Proofs

6.A.1 Proof of Lemma 3

The aim is to study the value function of the policy in two MDPs $\mathcal{M}_X$ and $\mathcal{M}_X^{(T)}$, where $\mathcal{M}_X$ is the non-stationary MDP with augmented states consisting of the actual state and the entire history of events till then, and $\mathcal{M}_X^{(T)}$ is an approximate MDP where only events in the past T time steps affect the state transition and event distribution. O

We adapt the proof of Theorem 1 from Xiao et al. (2019). For notational convenience, one can extend the state space of $\mathcal{M}_X^{(T)}$ to that of $\mathcal{M}_X$ by including the information of every past event in the state without changing the transition function. Define a class of MDPs $\mathcal{M}_h^{(T)}$, $h \in \{0, 1, 2, \dots\}$ such that when starting from initial (augmented) state $\bar{s} \in \bar{S}$, the transitions follow the transition kernel of $\mathcal{M}_X$ for h time steps and then the transition kernel of $\mathcal{M}_X^{(T)}$ for all transitions thereafter. The value functions induced by π in $\mathcal{M}_X$ and $\mathcal{M}_X^{(T)}$ can be related using these new intermediate MDPs, which interpolate between those two environments.

Since only a single policy is under discussion, the π in V^π is omitted for the rest of this proof for simplicity of notation. V denotes the value function of the policy in $\mathcal{M}_X$, $V^{(T)}$ denotes its value in $\mathcal{M}_X^{(T)}$, and $V_h^{(T)}$ in $\mathcal{M}_h^{(T)}$.

From the definition of $\mathcal{M}_h^{(T)}$ and the boundedness of r, we have

$$V^{(T)} = V_0^{(T)}, \text{ and } V = \lim_{h\to\infty} V_h^{(T)},$$

and hence,

$$\begin{aligned} V - V^{(T)} = \lim_{H\to\infty} V_H^{(T)} - V_0 = \lim_{H\to\infty} \left(V_H^{(T)} - V_0\right) &= \lim_{H\to\infty} \sum_{h=0}^{H-1} \left(V_{h+1}^{(T)} - V_h^{(T)}\right) \\ &= \sum_{h=0}^{\infty} \left(V_{h+1}^{(T)} - V_h^{(T)}\right). \end{aligned}$$

That is, the difference between V and $V^{(T)}$ can be characterized in terms of differences between $V_h^{(T)}$ and $V_{h+1}^{(T)}$ for each $h \in \{0, 1, 2, \dots\}$. Now, for any augmented state $\bar{s} \in \bar{S}$,

$$\begin{aligned} V_h^{(T)}(\bar{s}) = \sum_{t=0}^{h-1} \gamma^t \mathsf{E}_{\substack{\bar{s}_t\sim p_t(.|\bar{s}) \\ a_t\sim\pi(.|\bar{s}_t) \\ \bar{s}_{t+1}\sim p(.|\bar{s}_t,a_t)}} [r(\bar{s}_t, a_t, \bar{s}_{t+1})] + \gamma^h \mathsf{E}_{\substack{\bar{s}_h\sim p_h(.|\bar{s}) \\ a_h\sim\pi(.|\bar{s}_h) \\ \bar{s}_{t+1}\sim p^{(T)}(.|\bar{s}_h,a_h)}} [r(\bar{s}_h, a_h, \bar{s}_{h+1})] \\ + \sum_{t=h+1}^{\infty} \gamma^t \mathsf{E}_{\substack{\bar{s}_t\sim p_{t-h}^{(T)}\circ p_h(.|\bar{s}) \\ a_t\sim\pi(.|\bar{s}_t) \\ \bar{s}_{t+1}\sim p^{(T)}(.|\bar{s}_t,a_t)}} [r(\bar{s}_t, a_t, \bar{s}_{t+1})], \end{aligned}$$

where $p_t(.|\bar{s})$ denotes a transition of t steps in the environment $\mathcal{M}_X$ starting from $\bar{s}$, a superscript $^{(T)}$ denotes the corresponding object in $\mathcal{M}_X^{(T)}$, and the policy π is implicit whenever it is not explicitly mentioned in any expression. The reward function does not depend on the external events, so $r(\bar{s}_t, a_t, \bar{s}_{t+1})$ is just $r(s_t, a_t, s_{t+1})$, where $\bar{s} = (s, x)$ and s is the actual state.

This can also be written as

$$V_h^{(T)}(\bar{s}) = \sum_{t=0}^{h-1} \gamma^t \mathsf{E}_{\substack{\bar{s}_t\sim p_t(.|\bar{s}) \\ a_t\sim\pi(.|\bar{s}_t) \\ \bar{s}_{t+1}\sim p(.|\bar{s}_t,a_t)}} [r(\bar{s}_t, a_t, \bar{s}_{t+1})] + \gamma^h \mathsf{E}_{\bar{s}_h\sim p_h(.|\bar{s})} V^{(T)}(\bar{s}_h).$$

Using these two representations for $V_h^{(T)}$ and $V_{h+1}^{(T)}$ respectively gives

$$V_{h+1}^{(T)}(\bar{s}) - V_h^{(T)}(\bar{s}) = \gamma^h \mathsf{E}_{\substack{\bar{s}_h\sim p_h(.|\bar{s}) \\ a_h\sim\pi(.|\bar{s}_h) \\ \bar{s}_{h+1}\sim p(.|\bar{s}_h,a_h)}} [r(\bar{s}_h, a_h, \bar{s}_{h+1})] + \gamma^{h+1} \mathsf{E}_{\bar{s}_{h+1}\sim p_{h+1}(.|\bar{s})} V^{(T)}(\bar{s}_{h+1})$$

$$
\begin{aligned}
&- \gamma^h \mathsf{E}_{\substack{\bar{s}_h \sim p_h(.|\bar{s}) \\ a_h \sim \pi(.|\bar{s}_h) \\ \bar{s}_{t+1} \sim p^{(T)}(.|\bar{s}_h, a_h)}} [r(\bar{s}_h, a_h, \bar{s}_{h+1})] + \gamma^{h+1} \mathsf{E}_{\bar{s}_{h+1} \sim p^{(T)} \circ p_h(.|\bar{s})} V^{(T)}(\bar{s}_{h+1}) \\
&= \gamma^h \mathsf{E}_{\substack{\bar{s}_h \sim p_h(.|\bar{s}) \\ a_h \sim \pi(.|\bar{s}_h)}} \left[\mathsf{E}_{\bar{s}_{h+1} \sim p(.|\bar{s}_h, a_h)} r(\bar{s}_h, a_h, \bar{s}_{h+1}) - \mathsf{E}_{\bar{s}_{h+1} \sim p^{(T)}(.|\bar{s}_h, a_h)} r(\bar{s}_h, a_h, \bar{s}_{h+1}) \right] \\
&\quad + \gamma^{h+1} \mathsf{E}_{\substack{\bar{s}_h \sim p_h(.|\bar{s}) \\ a_h \sim \pi(.|\bar{s}_h, a_h)}} \left[\mathsf{E}_{\bar{s}_{h+1} \sim p(.|\bar{s}_h, a_h)} V^{(T)}(\bar{s}_{h+1}) - \mathsf{E}_{\bar{s}_{h+1} \sim p^{(T)}(.|\bar{s}_h, a_h)} V^{(T)}(\bar{s}_{h+1}) \right].
\end{aligned}
$$

Since the current state and the current event mark are independent and depend only on the previous events, the transition kernel can be factored as two independent distributions over S and $\mathcal{X}$. Hence, replacing the augmented state $\bar{s} \in \bar{S}$ with the explicit representation (s, H), where $s \in S$ and $H \in \mathcal{X}^\infty$ gives, for any bounded measurable function f on $S \times \mathcal{X}^\infty$,

$$
\begin{aligned}
&\mathsf{E}_{(s_{t+1}, H_{t+1}) \sim p(.|s_t, H_t, a_t)} f(s_{t+1}, H_{t+1}) - \mathsf{E}_{(s_{t+1}, H_{t+1}) \sim p^{(T)}(.|s_t, H_t, a_t)} f(s_{t+1}, H_{t+1}) \\
&\quad = \mathsf{E}_{\substack{s_{t+1} \sim Q_{H_t}(.|s_t, a_t) \\ X_{t+1} \sim Q^X_{H_t}(.|s_t, a_t)}} f(s_{t+1}, X_{t+1}, H_t) - \mathsf{E}_{\substack{s_{t+1} \sim Q^{(T)}_{H_t}(.|s_t, a_t) \\ X_{t+1} \sim Q^{X(T)}_{H_t}(.|s_t, a_t)}} f(s_{t+1}, X_{t+1}, H_t) \\
&\quad = \mathsf{E}_{\substack{s_{t+1} \sim Q_{H_t}(.|s_t, a_t) \\ X_{t+1} \sim Q^X_{H_t}(.|s_t, a_t)}} f(s_{t+1}, X_{t+1}, H_t) - \mathsf{E}_{\substack{s_{t+1} \sim Q^{(T)}_{H_t}(.|s_t, a_t) \\ X_{t+1} \sim Q^X_{H_t}(.|s_t, a_t)}} f(s_{t+1}, X_{t+1}, H_t) \\
&\qquad + \mathsf{E}_{\substack{s_{t+1} \sim Q^{(T)}_{H_t}(.|s_t, a_t) \\ X_{t+1} \sim Q^X_{H_t}(.|s_t, a_t)}} f(s_{t+1}, X_{t+1}, H_t) - \mathsf{E}_{\substack{s_{t+1} \sim Q^{(T)}_{H_t}(.|s_t, a_t) \\ X_{t+1} \sim Q^{X(T)}_{H_t}(.|s_t, a_t)}} f(s_{t+1}, X_{t+1}, H_t) \\
&\quad \leq \mathsf{E}_{X_{t+1} \sim Q^X_{H_t}(.|s_t, a_t)} \left[\|f(., X_{t+1}, H_t)\|_\infty \, \mathsf{TV}\left(Q_{H_t}(.|s_t, a_t), Q^{(T)}_{H_t}(.|s_t, a_t) \right) \right] \\
&\qquad + \mathsf{E}_{s_{t+1} \sim Q^{(T)}_{H_t}(.|s_t, a_t)} \left[\|f(s_{t+1}, ., H_t)\|_\infty \, \mathsf{TV}\left(Q^X_{H_t}(.|s_t, a_t), Q^{X(T)}_{H_t}(.|s_t, a_t) \right) \right] \\
&\quad \leq \|f\|_\infty \left(\sum_{t=T+1}^{\infty} M_t + \sum_{t=T+1}^{\infty} N_t \right).
\end{aligned}
$$

The second-to-last inequality is a result of Total Variation Distance being an integral probability metric (Sriperumbudur et al., 2012). Using this expression in the previous equation gives

$$
\begin{aligned}
V^{(T)}_{h+1}(\bar{s}) - V^{(T)}_h(\bar{s}) &\leq \gamma^h \|r\|_\infty \left(\sum_{t=T+1}^{\infty} (M_t + N_t) \right) + \gamma^{h+1} \left\| V^{(T)} \right\|_\infty \left(\sum_{t=T+1}^{\infty} (M_t + N_t) \right) \\
&\leq \gamma^h \left(\|r\|_\infty + \gamma \left\| V^{(T)} \right\|_\infty \right) \left(\sum_{t=T+1}^{\infty} (M_t + N_t) \right).
\end{aligned}
$$

This gives the final result,

$$\begin{aligned}\left\|V - V^{(T)}\right\|_\infty &\leq \sum_{h=1}^{\infty} \gamma^h \left(\|r\|_\infty + \gamma \left\|V^{(T)}\right\|_\infty\right) \left(\sum_{t=T+1}^{\infty} (M_t + N_t)\right) \\ &= \frac{1}{1-\gamma} \left(\|r\|_\infty + \gamma \left\|V^{(T)}\right\|_\infty\right) \left(\sum_{t=T+1}^{\infty} (M_t + N_t)\right).\end{aligned}$$

□

6.A.2 Proof of Lemma 4

Let π be a policy that depends only on the current state and the past T events.

$$\begin{aligned}V^\pi((s, x_{0:\infty})) &= \mathsf{E}_{\substack{s' \sim Q_{x_{0:\infty}}(.|s,a) \\ x' \sim Q_{x_{0:\infty}}(.) \\ a = \pi((s, x_{0:\infty}))}} \left[r(s, a, s') + V^\pi\left((s', x', x_{0:\infty})\right)\right], \\ \text{and } V^\pi((s, x_{0:T}, 0)) &= \mathsf{E}_{\substack{s' \sim Q_{x_{0:T}}(.|s,a) \\ x' \sim Q_{x_{0:T}}(.) \\ a = \pi((s, x_{0:T}))}} \left[r(s, a, s') + V^\pi\left((s', x', x_{0:T})\right)\right].\end{aligned}$$

Taking the difference between these two expressions and adding and subtracting the additional term

$$\mathsf{E}_{\substack{s' \sim Q_{x_{0:T}}(.|s,a) \\ x' \sim Q_{x_{0:T}}(.) \\ a = \pi((s, x_{0:T}))}} V^\pi((s', x', x_{0:\infty}))$$

gives $|V^\pi((s, x_{0:\infty})) - V^\pi((s, x_{0:T}, 0))|$

$$\begin{aligned}&\leq \|r\|_\infty \left(\sum_{t=T+1}^{\infty} M_t\right) + \|V^\pi\|_\infty \left(\sum_{t=T+1}^{\infty} M_t + N_t\right) \\ &\quad + \gamma \mathsf{E}_{\substack{s' \sim Q_{x_{0:T}}(.|s,a) \\ x' \sim Q_{x_{0:T}}(.) \\ a = \pi((s, x_{0:T}))}} \left|V^\pi\left((s', x', x_{0:\infty})\right) - V^\pi\left((s', x', x_{0:T})\right)\right|.\end{aligned}$$

In the above inequality, the left-hand side is the difference between two value functions whose first T events are the same and differ everywhere else, but the right-hand side has a discounted term in which the first $T + 1$ terms coincide. So, repeating the same procedure results in value function differences at states that increasingly coincide.

Since each repetition adds a γ factor to the term, which itself is bounded by $\frac{1}{1-\gamma}$ due to the

rewards being bounded, the resulting series converges, giving us the following bound.

$$
\begin{aligned}
&\sup_{\bar{s}=(s,x_{0:\infty})} \left|V^\pi((s,x_{0:\infty})) - V^\pi((s,x_{0:T},0))\right| \\
&\qquad \le \sum_{t=T+1}^{\infty} \gamma^{t-T-1} \left[\|r\|_\infty \left(\sum_{t'=t}^{\infty} M_{t'} \right) + \|V\|_\infty \left(\sum_{t'=t}^{\infty} M_{t'} + N_{t'} \right) \right] \\
&\qquad \le \left(1 + \frac{\gamma}{1-\gamma}\right) \sum_{t=T+1}^{\infty} \sum_{t'=t}^{\infty} \gamma^t (M_{t'} + N_{t'}) \\
&\qquad \le \frac{1}{1-\gamma} \sum_{t'=T+1}^{\infty} \sum_{t=0}^{T-t'-1} \gamma^t (M_{t'} + N_{t'}) \\
&\qquad = \frac{1}{1-\gamma} \sum_{t'=T+1}^{\infty} \frac{1-\gamma^{t'-T}}{1-\gamma} (M_{t'} + N_{t'}) \\
&\qquad \le \frac{1}{(1-\gamma)^2} \sum_{t=T+1}^{\infty} (M_t + N_t)\,.
\end{aligned}
$$

□

6.A.3 Sample Complexity

The proof of Theorem 7 closely follows the results in Lazaric et al. (2012). In this section, we state the preliminaries and statements of some of their Lemmas and give details where our proof differs from theirs.

The final result is a bound on the expected error of the learned value function, so the proof has two parts, the first being Lemma 8, a bound on the empirical value function, and the second part being generalization to the entire space. The bound on the empirical error is quite similar to the original in the stationary case, due to which we skip some steps. The final bound on the expected error, whose proof is given in Section 6.A.3, offers more insights into the trade-off between the tractability of the algorithm and the approximation error due to non-stationarity.

Empirical Error

We wish to show that with probability at least $1-\delta$,

$$
\|v - \hat{v}\|_N \le \frac{1}{\sqrt{1-\gamma^2}} \left\| v - \hat{\Pi} v \right\|_N + \frac{L}{(1-\gamma)^2} \sqrt{\frac{d}{\nu_N}} \left(\sqrt{\frac{2\log(2d/\delta)}{N}} + \frac{1}{N} \right).
$$

Following Lazaric et al. (2012),

$$\left\|V - \hat{V}\right\|_N = \|v - \hat{v}\|_N \leq \frac{1}{\sqrt{1-\gamma^2}} \left\|v - \hat{\Pi}v\right\|_N + \frac{1}{1-\gamma} \left\|\hat{\Pi}v - \hat{\Pi}\hat{\mathcal{T}}v\right\|_N . \tag{6.17}$$

The first term is the approximation error, which is due to the restricted function space and is unavoidable. The second term also includes the estimation error due to using the pathwise Bellman operator instead of the actual Bellman operator and is a function of the number of samples used. It can be written as $\left\|\hat{\Pi}\xi\right\|_N$, where $\xi = \hat{\mathcal{T}}v - v$ is the estimation error. Now, for $t < N$,

$$\begin{aligned}
\xi_t &= r_t + \gamma V(\bar{s}_{t+1}) - V(\bar{s}_t) \\
&= r(s_t, \pi(\bar{s}_t), s_{t+1}) + \gamma V(\bar{s}_{t+1}) - \mathsf{E}_{\bar{S}_{t+1}}\left[r(s_t, \pi(\bar{s}_t, \bar{S}_{t+1}) + \gamma V(\bar{S}_{t+1})\right] \\
&= r(s_t, \pi(\bar{s}_t), s_{t+1}) - \mathsf{E}_{S_{t+1}\sim Q_{x:t}(.|s_t,\pi(\bar{s}_t))}\left[r(s_t, \pi(\bar{s}_t), S_{t+1})\right] \\
&\qquad + \gamma\left[V(\bar{s}_{t+1}) - \mathsf{E}_{\bar{S}_{t+1}\sim Q, Q^X_{x:t}(.|s_t,\pi(\bar{s}_t))} V(\bar{S}_{t+1})\right],
\end{aligned}$$

and for $t = N$,

$$\xi_t = r(s_t, \pi(\bar{s}_t), s_{t+1}) - \mathsf{E}_{S_{t+1}\sim Q_{x:t}(.|s_t,\pi(\bar{s}_t))}\left[r(s_t, \pi(\bar{s}_t), S_{t+1})\right] - \gamma \mathsf{E}_{\bar{S}_{t+1}\sim Q, Q^X_{x:t}(.|s_t,\pi(\bar{s}_t))}\left[V(\bar{S}_{t+1})\right].$$

ξ_t are functions of $\bar{s}_t$, which are samples from the Markov chain induced by the policy π in the augmented MDP $\bar{M}$. Considering them as a random variables, $(\xi_t \varphi_t(\bar{s}_t))_{1\leq t<N}$ is a Martingale difference sequence w.r.t $\bar{s}_t$, with each element being zero mean and bounded by $\frac{L}{1-\gamma}$. So, applying Azuma's inequality to $(\xi_t \varphi_i(\bar{s})_t)_{1\leq t<N}$ gives, for each $i \in [d]$,

$$\mathsf{P}\left(\left|\sum_{t=1}^{N-1} \xi_t \varphi_i(s_t)\right| \geq \epsilon\right) < 2\exp\left(-\frac{\epsilon^2(1-\gamma)^2}{2(N-1)L^2}\right).$$

Letting the right hand side be $\frac{\delta}{d}$, with probability at least $1-\delta$,

$$\left|\sum_{t=1}^{N-1} \xi_t \varphi_i(s_t)\right| < \frac{L}{1-\gamma}\sqrt{2(N-1)\log\left(\frac{2d}{\delta}\right)} \quad \text{for all } i \in [d].$$

Since $|\xi_n \varphi_i(\bar{s}_t)| < \frac{L}{1-\gamma}$ always, we have,

$$\left|\sum_{t=1}^{N-1} \xi_t \varphi_i(s_t)\right| < \frac{L}{1-\gamma}\left(\sqrt{2(N-1)\log\left(\frac{2d}{\delta}\right)} + 1\right), \text{ for all } i \in [d] \text{ w.p. } \geq 1-\delta. \tag{6.18}$$

Following Lazaric et al. (2012), this can be used to bound $||\hat{\Pi}\xi_n||_N$ as

$$\left\|\hat{\Pi}\hat{\mathcal{T}}v - \hat{\Pi}v\right\|_N \leq \frac{1}{N}\sqrt{\frac{d}{\nu_n}}\max_i \left|\sum_{t=1}^{N}\xi_t\varphi_i(s_t)\right|, \tag{6.19}$$

where ν_n is the smallest positive eigenvalue of the Gram matrix $\Phi^\intercal\Phi$.

Substituting (6.18) and (6.19) in (6.17) gives the desired result.

Generalization

Intuitively, bounding the expected error of the estimated value function based on the error at just a few samples requires the samples to be close to the stationary distribution and the function values at the samples to be close to their expected values.

The first requirement is formalized in terms of the rate of mixing of the Markov chain induced by the MDP by the policy.

Definition 1. *A Markov chain $\mathcal{M} = (X_t)_{t\geq 1}$ is said to be exponentially β-mixing with parameters $\bar{\beta}, b, \kappa$ if its mixing coefficients*

$$\beta_i = \sup_{t\geq 1}\left[\sup_{B\in\sigma(X_{t+i},\dots)}\left|\mathsf{P}\left(B|X_1,\dots,X_t\right) - \mathsf{P}(B)\right|\right]$$

satisfy $\beta_i \leq \bar{\beta}\exp\left(-bi^\kappa\right)$.

When such an exponentially mixing Markov chain is also ergodic and aperiodic with stationary distribution ρ, for any initial distribution λ, there is a bound on the following total variation: (Definition 21 of Lazaric et al. (2012))

$$\left\|\int_{\mathcal{X}}\lambda(dx)\mathsf{P}\left(.|x\right) - \rho(.)\right\|_{\mathsf{TV}} \leq \bar{\beta}\exp\left(-bi^\kappa\right).$$

This helps in bounding the difference between the norm of a function in the stationary distribution and the empirical distribution defined using a few samples.

Lemma 9 (Generalization Lemma for Markov chains, Lemma 24 of Lazaric et al. (2012))**.** *Let $\tilde{\mathcal{F}}$ be the d-dimensional class of linear functions truncated at threshold B. Let $(X_t)_{t\in[n]}$ be a sequence of samples from an ergodic, aperiodic exponentially β-mixing Markov chain with parameters $\bar{\beta}, b, \kappa$, arbitrary initial distribution, and stationary distribution ρ. If the first $\tilde{n}$ samples are discarded and only the last $n - \tilde{n}$ samples are used, then with probability at least*

$1-\delta$, *the empirical and* $l_2(\rho)$ *norm of any function* $\tilde{f} \in \tilde{\mathcal{F}}$ *are related as,*

$$\left\|\tilde{f}\right\| - 2\left\|\tilde{f}\right\|_{X_{1:n}} \leq \epsilon(\delta),$$
$$\left\|\tilde{f}\right\|_{X_{1:n}} - 2\sqrt{2}\left\|\tilde{f}\right\| \leq \epsilon(\delta),$$

$$\textit{for } \epsilon(\delta) = 12B\sqrt{\frac{2\Lambda(n-\tilde{n},d,\delta)}{n-\tilde{n}}\max\left\{\frac{\Lambda(n-\tilde{n},d,\delta)}{1},1\right\}^{\frac{1}{\kappa}}},$$

$$\Lambda(n,d,\delta) = 2(d+1)\log n + \log\frac{\epsilon}{\delta} + \log^{+}\left(\max\left\{16(6e)^{2(d+1)},\bar{\beta}\right\}\right),$$

$$\textit{and } \tilde{n} = \left(\frac{1}{b}\log\left(\frac{2e\bar{\beta}n}{\delta}\right)\right)^{\frac{1}{\kappa}}.$$

The above lemma is essentially a generalization bound for truncated linear functions operating on samples from an exponentially mixing Markov chain and is used in the proof of Theorem 7 to translate the bound on the empirical error to a bound on the expected error in the stationary distribution of the Markov chain induced by the policy under evaluation.

The above bounds hold for every member of the d-dimensional class of functions, hence the dependence of ϵ on d through $\Lambda(n,d,\delta)$. In contrast, when there is just one truncated linear function under consideration, the above bound holds, but with

$$\Lambda(n,\delta) = \log\frac{\epsilon}{\delta} + \log\left(\max\left\{6,n\bar{\beta}\right\}\right).$$

Another issue for generalizing the sample-based bound in Lemma 8 to the entire state space is its dependence on the eigenvalues of the sample Gram matrix. For tackling this problem, Lazaric et al. (2012) derived the following probabilistic lower bound on the smallest eigenvalue of the sample-based Gram matrix as a function of the smallest eigenvalue of the Gram matrix $G = \mathsf{E}_{x\sim\rho}\left[\phi(x)\phi(x)^{\intercal}\right]$.

Lemma 10 (Lemma 4 of Lazaric et al. (2012))**.** *Let* $\omega > 0$ *be the smallest eigenvalue of the Gram matrix defined above. If the data generating process is an exponentially* β*-mixing Markov chain with parameters* $\bar{\beta}, b, \kappa$*, then the smallest eigenvalue* ν_n *of the sample-based Gram matrix constructed using* n *samples satisfies*

$$\sqrt{\nu_n} \geq \sqrt{\nu} = \frac{\sqrt{\omega}}{2} - 6L\sqrt{\frac{2\Lambda(n,d,\delta)}{n}\max\left\{\frac{\Lambda(n,d,\delta)}{b},1\right\}^{\frac{1}{\kappa}}} > 0,$$

provided the number of samples n satisfies

$$n > \frac{288L^2\Lambda(n,d,\delta)}{\omega}\max\left\{\frac{\Lambda(n,d,\delta)}{b},1\right\}^{\frac{1}{\kappa}},$$

$$\textit{where}\quad \Lambda(n,d,\delta) = 2(d+1)\log n + \log\frac{e}{\delta} + \log^{+}\left(\max\left\{18(6e)^{2(d+1)},\bar{\beta}\right\}\right)$$

comes from a simpler version of Lemma 9 without the additional initial $\tilde{n}$ samples for burn-in of the Markov chain.

The generalization Lemma 9, combined with the lower bound on the eigenvalues and our results in Sections 6.3 and 6.4.1, give rise to Theorem 7 on the sample complexity of policy evaluation.

Chapter 7

Conclusion

While reinforcement learning offers a very general framework to train agents that can perform arbitrary sequential decision-making tasks, it is very data-intensive and offers unique challenges that are mostly not present in other learning tasks such as supervised and unsupervised learning. In this thesis, we focus on studying reinforcement learning at its limits, where the agent has to operate under constraints imposed on its behavior and objectives. Specifically, we explored three key challenges: (i) risk-sensitive sequential decision-making, (ii) learning in the absence of online access to the environment, and (iii) dealing with non-stationarity in the environment due to the influence of an exogenous temporal event process.

First, we studied sequential decision-making under a risk metric, wherein the agent is expected to optimize according to worst-case scenarios of obtained rewards. We analyzed this problem in the setting of combinatorial semi-bandits, where an agent selects a subset of actions (super-arms) at each time step and receives feedback only for the selected arms. The objective of the agent is to minimize the Conditional Value-At-Risk (CVaR) of the sum of the rewards obtained at each time step, over a given time horizon. We proposed and theoretically analyzed two algorithms to minimize the CVaR regret, considering cases where the reward distributions of the individual arms of the bandit are Gaussian and bounded, respectively. Since the algorithm for the bounded rewards case is significantly more computationally expensive, we studied the effect of discretizing the algorithm on the regret and the computational complexity. Finally, we validated our theoretical results through experiments on synthetic environments with Gaussian and Bernoulli rewards.

Next, we considered the task of learning complex agents with limited data. We first considered the case where the agent is given a fixed dataset collected in an unknown fashion and needs to learn hierarchical policies that can plan over high-level tasks. We introduced a general method that enables the adaptation of existing online hierarchical RL algorithms to the offline

setting by leveraging a pessimistic model. More specifically, we showed that by planning in the latent action space of a Conditional Variational Auto-Encoder learned on the offline dataset in a pessimistic manner, an online hierarchical algorithm can be applied in the offline setting without losing the advantages associated with learning a hierarchy online. Experimental results on standard continuous control tasks demonstrated the efficacy of this approach in standard, transfer, and goal-conditioned settings.

We then broadened our focus to consider the setting where the agent is given the option to minimally augment the given data so as to obtain an optimal policy. We showed that using a representation-based uncertainty model to choose new regions of the environment to explore, followed by using a novel exploration policy based on the same model, allows the agent to optimize the tradeoff between final performance and the cost of exploration and augment the given dataset in a targeted and efficient manner. Through extensive experimentation on continuous control benchmarks, we showed that our active trajectory collection method significantly reduces the need for additional online interaction compared to conventional fine-tuning approaches.

Finally, we considered the most general reinforcement learning setting wherein the environment of the agent itself evolves dynamically due to the influence of an exogenous temporal event process. In such cases, the transition dynamics of the environment are subject to external perturbations that do not depend on the agent's actions, thereby violating the standard Markovian assumption. We formulated this problem rigorously and derived conditions under which the problem is well-posed and a tractable, approximately optimal solution exists. Our analysis revealed that if the influence of past events on the environment as well as future events decays at a sufficiently fast rate, then the agent can effectively learn policies using a finite history of past events, with the requisite length of the history depending on the rate of decay of the perturbations.

We proposed a policy iteration algorithm for this setting and theoretically analyzed its convergence behavior. We show that while it is not guaranteed to converge, because of approximation errors due to the non-Markovian nature of the external influence, the policy is guaranteed to improve in those regions of the state space where the Bellman error is sufficiently large, depending again on the rate of decay of influence of the exogenous events. Furthermore, we extended the finite-sample analysis of Least Squares Temporal Difference (LSTD) to our setting and quantified the impact of non-stationarity on policy evaluation. We then conducted experiments using a non-stationary pendulum control task influenced by a discrete-time Hawkes process, empirically validating our theoretical results.

While there are numerous challenges that hinder the deployment of reinforcement learning

algorithms in the real world, this thesis is a step towards understanding and analyzing fundamental problems that arise in many such scenarios. For the problems of developing risk-averse sequential decision-making algorithms, learning from offline data, and adapting to dynamically evolving environments, we provide theoretical insights and algorithmic advances that contribute to making RL more robust and applicable to constrained and uncertain settings. Future work may extend these ideas further by exploring richer classes of risk measures, extending our uncertainty-based algorithms to high-dimensional and real-world environments, and developing scalable algorithms that can handle complex non-stationary environments.

References

Acerbi, Carlo and Dirk Tasche (2002). "On the coherence of expected shortfall". In: *Journal of banking & finance* 26.7, pp. 1487–1503 (cit. on p. 24).

Agrawal, Shipra and Navin Goyal (2012). "Analysis of thompson sampling for the multi-armed bandit problem". In: *25th Annual Conference on Learning Theory*. (COLT) (cit. on p. 2).

Ajay, Anurag, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum (2021). "OPAL: Offline Primitive Discovery for Accelerating Offline Reinforcement Learning". In: *9th International Conference on Learning Representations*. (ICLR) (cit. on p. 68).

Alegre, Lucas N., Ana L. C. Bazzan, and Bruno C. da Silva (2021). "Minimum-Delay Adaptation in Non-Stationary Reinforcement Learning via Online High-Confidence Change-Point Detection". In: *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. (AAMAS) (cit. on p. 94).

An, Gaon, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song (2021). "Uncertainty-based offline reinforcement learning with diversified q-ensemble". In: *Advances in Neural Information Processing Systems*. Vol. 34. (NeurIPS) (cit. on p. 90).

Anantharam, V., P. Varaiya, and J. Walrand (1987). "Asymptotically efficient allocation rules for the multiarmed bandit problem with multiple plays-Part I: I.I.D. rewards". In: *IEEE Transactions on Automatic Control* 32.11, pp. 968–976 (cit. on pp. 27, 34).

Andrychowicz, Marcin, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba (2017). "Hindsight experience replay". In: *Advances in Neural Information Processing Systems*. Vol. 30. (NeurIPS) (cit. on p. 59).

Auer, Peter, Nicolo Cesa-Bianchi, and Paul Fischer (2002a). "Finite-time analysis of the multiarmed bandit problem". In: *Machine Learning* 47.2-3, pp. 235–256 (cit. on pp. 2, 3, 9, 18, 33).

Auer, Peter, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire (2002b). "The nonstochastic multiarmed bandit problem". In: *SIAM journal on computing* 32.1, pp. 48–77 (cit. on p. 25).

Ayoub, Alex, Zeyu Jia, Csaba Szepesvari, Mengdi Wang, and Lin Yang (2020). "Model-based reinforcement learning with value-targeted regression". In: *Proceedings of the 37th International Conference on Machine Learning*. (ICML) (cit. on p. 4).

Ayyagari, Ranga Shaarad and Ambedkar Dukkipati (2023). "Risk-averse combinatorial semi-bandits". In: *2023 IEEE International Symposium on Information Theory (ISIT)*. (ISIT). IEEE (cit. on p. 27).

— (2024). "Markov Decision Processes under External Temporal Processes". In: *arXiv* 2305.16056 (cit. on p. 95).

Ayyagari, Ranga Shaarad, Anurita Ghosh, and Ambedkar Dukkipati (2024). "Temporal Abstraction in Reinforcement Learning with Offline Data". In: *arXiv* 2407.15241 (cit. on p. 52).

Bachman, Philip, Alessandro Sordoni, and Adam Trischler (2017). "Learning algorithms for active learning". In: *Proceedings of the 34th International Conference on Machine Learning*. (ICML) (cit. on p. 6).

Bacon, Pierre-Luc, Jean Harb, and Doina Precup (2017). "The option-critic architecture". In: *Proceedings of the 31st AAAI conference on artificial intelligence* (cit. on pp. 5, 53, 67).

Balcan, Maria-Florina, Alina Beygelzimer, and John Langford (2006). "Agnostic active learning". In: *Proceedings of the 23rd International Conference on Machine Learning*. (ICML) (cit. on p. 90).

Beeson, Alex and Giovanni Montana (2022). "Improving td3-bc: Relaxed policy constraint for offline learning and stable online fine-tuning". In: *3rd Offline Reinforcement Learning Workshop at Neural Information Processing Systems*. (NeurIPS Workshop) (cit. on p. 86).

Bellemare, Marc G., Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos (2016). "Unifying Count-Based Exploration and Intrinsic Motivation". In: *Advances in Neural Information Processing Systems*. Vol. 29. (NeurIPS) (cit. on p. 90).

Bertsekas, Dimitri P. (2012). *Dynamic Programming and Optimal Control, Volume 2*. 4th. Belmont, MA: Athena Scientific (cit. on pp. 10, 21).

Bhat, Sanjay P. and Prashanth L.A. (2019). "Concentration of risk measures: A Wasserstein distance approach". In: *Advances in Neural Information Processing Systems*. Vol. 32. (NeurIPS) (cit. on p. 36).

Brockman, Greg, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba (2016). "OpenAI Gym". In: *arXiv* 1606.01540 (cit. on pp. 56, 65).

Bubeck, Sébastien, Nicolo Cesa-Bianchi, and Gábor Lugosi (2013). "Bandits with heavy tail". In: *IEEE Transactions on Information Theory* 59.11, pp. 7711–7717 (cit. on p. 3).

Burda, Yuri, Harrison Edwards, Amos Storkey, and Oleg Klimov (2019). "Exploration by random network distillation". In: *7th International Conference on Learning Representations*. (ICLR) (cit. on pp. 86, 90).

Cassel, Asaf, Shie Mannor, and Assaf Zeevi (2018). "A General Approach to Multi-Armed Bandits Under Risk Criteria". In: *31st Annual Conference on Learning Theory*. (COLT) (cit. on p. 36).

— (2023). "A General Framework for Bandit Problems Beyond Cumulative Objectives". In: *Mathematics of Operations Research* 18.4, pp. 2196–2232 (cit. on pp. 36, 44).

Castro, Rui M (2007). "Active learning and adaptive sampling for non-parametric inference". PhD thesis. Rice University (cit. on p. 23).

Chen, Wei, Wei Hu, Fu Li, Jian Li, Yu Liu, and Pinyan Lu (2016). "Combinatorial Multi-Armed Bandit with General Reward Functions". In: *Advances in Neural Information Processing Systems*. Vol. 29. (NeurIPS) (cit. on pp. 36, 37, 45, 46).

Chen, Wei, Yajun Wang, and Yang Yuan (2013). "Combinatorial Multi-Armed Bandit: General Framework and Applications". In: *Proceedings of the 30th International Conference on Machine Learning*. (ICML) (cit. on pp. 27, 36, 39).

Cheung, Wang Chi, David Simchi-Levi, and Ruihao Zhu (2020). "Reinforcement Learning for Non-Stationary Markov Decision Processes: The Blessing of (More) Optimism". In: *Proceedings of the 37th International Conference on Machine Learning*. (ICML) (cit. on pp. 94, 118).

Cohn, David A, Zoubin Ghahramani, and Michael I Jordan (1996). "Active learning with statistical models". In: *Journal of Artificial Intelligence Research* 4, pp. 129–145 (cit. on p. 6).

Combes, Richard, M. Sadegh Talebi, Alexandre Proutiere, and Marc Lelarge (2015). "Combinatorial Bandits Revisited". In: *Advances in Neural Information Processing Systems*. Vol. 28. (NeurIPS) (cit. on p. 36).

Dann, Christoph and Emma Brunskill (2015). "Sample complexity of episodic fixed-horizon reinforcement learning". In: *Advances in Neural Information Processing Systems*. Vol. 28. (NeurIPS) (cit. on p. 4).

Dietterich, Thomas, George Trimponias, and Zhitang Chen (2018). "Discovering and removing exogenous state variables and rewards for reinforcement learning". In: *Proceedings of the 35th International Conference on Machine Learning*. (ICML) (cit. on p. 119).

Domingues, Omar Darwiche, Pierre Ménard, Emilie Kaufmann, and Michal Valko (2021). "Episodic reinforcement learning in finite mdps: Minimax lower bounds revisited". In: *32nd International Conference on Algorithmic Learning Theory*. (ALT) (cit. on p. 4).

Dong, Kefan, Jiaqi Yang, and Tengyu Ma (2021). "Provable Model-based Nonlinear Bandit and Reinforcement Learning: Shelve Optimism, Embrace Virtual Curvature". In: *Advances in Neural Information Processing Systems*. Vol. 34. (NeurIPS) (cit. on p. 4).

Dukkipati, Ambedkar, Ranga Shaarad Ayyagari, Bodhisattwa Dasgupta, Parag Dutta, and Prabhas Reddy Onteru (2025). "Active Reinforcement Learning Strategies for Offline Policy Improvement". In: *Proceedings of the 39th AAAI Conference on Artificial Intelligence* (cit. on p. 72).

Ecoffet, Adrien, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune (2021). "First return, then explore". In: *Nature* 590.7847, pp. 580–586 (cit. on p. 91).

Efroni, Yonathan, Dylan J Foster, Dipendra Misra, Akshay Krishnamurthy, and John Langford (2022). "Sample-Efficient Reinforcement Learning in the Presence of Exogenous Information". In: *35th Annual Conference on Learning Theory*. (COLT) (cit. on p. 119).

Feng, Fan, Biwei Huang, Kun Zhang, and Sara Magliacane (2022). "Factored adaptation for non-stationary reinforcement learning". In: *Advances in Neural Information Processing Systems*. Vol. 35. (NeurIPS) (cit. on p. 119).

Fu, Justin, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine (2020). "D4RL: Datasets for deep data-driven reinforcement learning". In: *arXiv* 2004.07219 (cit. on pp. 56, 65, 79, 117).

Fujimoto, Scott, Herke Hoof, and David Meger (2018). "Addressing function approximation error in actor-critic methods". In: *Proceedings of the 35th International conference on machine learning*. (ICML) (cit. on pp. 22, 81).

Fujimoto, Scott, David Meger, and Doina Precup (2019). "Off-policy deep reinforcement learning without exploration". In: *Proceedings of the 36th International conference on machine learning*. (ICML) (cit. on pp. 5, 68).

Gai, Yi, Bhaskar Krishnamachari, and Rahul Jain (2010). "Learning Multiuser Channel Allocations in Cognitive Radio Networks: A Combinatorial Multi-Armed Bandit Formulation". In: *2010 IEEE Symposium on New Frontiers in Dynamic Spectrum*. (DySPAN) (cit. on pp. 3, 27, 36).

— (2012). "Combinatorial Network Optimization With Unknown Variables: Multi-Armed Bandits With Linear Rewards and Individual Observations". In: *IEEE/ACM Transactions on Networking* 20.5, pp. 1466–1478 (cit. on pp. 3, 27, 29).

Gal, Yarin, Riashat Islam, and Zoubin Ghahramani (2017). "Deep Bayesian Active Learning with Image Data". In: *Proceedings of the 34th International Conference on Machine Learning*. (ICML) (cit. on p. 90).

Galichet, Nicolas (2015). "Contributions to Multi-Armed Bandits : Risk-Awareness and Sub-Sampling for Linear Contextual Bandits". PhD thesis. Université Paris Sud - Paris XI (cit. on p. 36).

Galichet, Nicolas, Michèle Sebag, and Olivier Teytaud (2013). "Exploration vs Exploitation vs Safety: Risk-Aware Multi-Armed Bandits". In: *Proceedings of the 5th Asian Conference on Machine Learning*. (ACML) (cit. on pp. 34, 36).

Garivier, Aurélien and Olivier Cappé (2011). "The KL-UCB algorithm for bounded stochastic bandits and beyond". In: *24th Annual Conference on Learning Theory*. (COLT) (cit. on p. 3).

Go1 SDK HighLevel Interfaces (2023). https://unitree-docs.readthedocs.io/en/latest/get_started/Go1_Edu.html. Accessed: 2025-02-13 (cit. on p. 80).

Guo, Xin, Yonghui Huang, and Yi Zhang (2024). "On Average Optimality for Non-Stationary Markov Decision Processes in Borel Spaces". In: *Mathematics of Operations Research* (cit. on p. 118).

Györfi, László, Michael Kohler, Adam Krzyzak, Harro Walk, et al. (2002). *A distribution-free theory of nonparametric regression*. Springer Series in Statistics. Springer (cit. on p. 115).

Haarnoja, Tuomas, Aurick Zhou, Pieter Abbeel, and Sergey Levine (2018). "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *Proceedings of the 35th International conference on machine learning*. (ICML) (cit. on pp. 6, 22, 56, 82).

Hadoux, Emmanuel, Aurélie Beynier, and Paul Weng (2014). "Solving hidden-semi-Markov-mode Markov decision problems". In: *8th International Conference on Scalable Uncertainty Management*. (SUM) (cit. on pp. 94, 119).

Hallak, Assaf, Dotan Di Castro, and Shie Mannor (2015). "Contextual markov decision processes". In: *arXiv* 1502.02259 (cit. on pp. 94, 119).

Han, Dong, Beni Mulyana, Vladimir Stankovic, and Samuel Cheng (2023). "A survey on deep reinforcement learning algorithms for robotic manipulation". In: *Sensors* 23.7, p. 3762 (cit. on p. 4).

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd. New York, NY: Springer (cit. on p. 23).

Hernández-Lerma, Onésimo and Jean B Lasserre (1996). *Discrete-time Markov control processes: basic optimality criteria*. Applications of Mathematics. Springer (cit. on pp. 96, 102).

Hill, Ashley et al. (2018). *Stable Baselines*. https://github.com/hill-a/stable-baselines (cit. on pp. 65, 66).

Huang, Baihe, Kaixuan Huang, Sham Kakade, Jason D Lee, Qi Lei, Runzhe Wang, and Jiaqi Yang (2021a). "Going beyond linear rl: Sample efficient neural function approximation". In: *Advances in Neural Information Processing Systems*. Vol. 34. (NeurIPS) (cit. on p. 4).

Huang, Xueqin, Han Deng, Wei Zhang, Ran Song, and Yibin Li (2021b). "Towards multi-modal perception-based navigation: A deep reinforcement learning method". In: *IEEE Robotics and Automation Letters*. (RA-L) 6.3, pp. 4986–4993 (cit. on p. 4).

Jin, Chi, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan (2023). "Provably efficient reinforcement learning with linear function approximation". In: *Mathematics of Operations Research* 48.3, pp. 1496–1521 (cit. on p. 4).

Kagrecha, Anmol, Jayakrishnan Nair, and Krishna Jagannathan (2019). "Distribution oblivious, risk-aware algorithms for multi-armed bandits with unbounded rewards". In: *Advances in Neural Information Processing Systems*. Vol. 32. (NeurIPS) (cit. on p. 36).

Kaufmann, Elia, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza (2023). "Champion-level drone racing using deep reinforcement learning". In: *Nature* 620.7976, pp. 982–987 (cit. on p. 4).

Khajonchotpanya, Najakorn, Yilin Xue, and Napat Rujeerapaiboon (2021). "A revised approach for risk-averse multi-armed bandits under cvar criterion". In: *Operations Research Letters* 49.4, pp. 465–472 (cit. on p. 25).

Khetarpal, Khimya, Martin Klissarov, Maxime Chevalier-Boisvert, Pierre-Luc Bacon, and Doina Precup (2020). "Options of interest: Temporal abstraction with interest functions". In: *Proceedings of the 34th AAAI Conference on Artificial Intelligence* (cit. on pp. 5, 67).

Kidambi, Rahul, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims (2020). "MOREL: Model-based offline reinforcement learning". In: *Advances in Neural Information Processing Systems*. Vol. 33. (NeurIPS) (cit. on pp. 5, 52, 56, 67, 90).

Klissarov, Martin and Doina Precup (2021). "Flexible option learning". In: *Advances in Neural Information Processing Systems*. Vol. 34. (NeurIPS) (cit. on pp. 56, 66, 69).

Konyushova, Ksenia, Yutian Chen, Thomas Paine, Caglar Gulcehre, Cosmin Paduraru, Daniel J Mankowitz, Misha Denil, and Nando de Freitas (2021). "Active offline policy selection". In: *Advances in Neural Information Processing Systems*. Vol. 34. (NeurIPS) (cit. on p. 91).

Kostrikov, Ilya, Ashvin Nair, and Sergey Levine (2022). "Offline reinforcement learning with implicit q-learning". In: *10th International Conference on Learning Representations*. (ICLR) (cit. on pp. 82, 86).

Kumar, Aviral, Aurick Zhou, George Tucker, and Sergey Levine (2020). "Conservative q-learning for offline reinforcement learning". In: *Advances in Neural Information Processing Systems*. Vol. 33. (NeurIPS) (cit. on pp. 5, 56, 86).

Kveton, Branislav, Zheng Wen, Azin Ashkan, and Csaba Szepesvari (2015). "Tight Regret Bounds for Stochastic Combinatorial Semi-Bandits". In: *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*. (AISTATS) (cit. on p. 36).

Lattimore, Finnian, Tor Lattimore, and Mark D Reid (2016). "Causal bandits: Learning good interventions via causal inference". In: *Advances in Neural Information Processing Systems*. Vol. 29. (NeurIPS) (cit. on p. 3).

Lattimore, Tor and Csaba Szepesvári (2020). *Bandit algorithms*. Cambridge University Press (cit. on p. 9).

Laurent, B. and P. Massart (2000). "Adaptive Estimation of a Quadratic Functional by Model Selection". In: *The Annals of Statistics* 28.5, pp. 1302–1338 (cit. on pp. 33, 38).

Lazaric, Alessandro, Mohammad Ghavamzadeh, and Rémi Munos (2012). "Finite-sample analysis of least-squares policy iteration". In: *Journal of Machine Learning Research* (cit. on pp. 13, 109, 111, 113, 115, 126–129).

Lecarpentier, Erwan and Emmanuel Rachelson (2019). "Non-Stationary Markov Decision Processes, a Worst-Case Approach using Model-Based Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Vol. 32. (NeurIPS) (cit. on pp. 94, 118).

Levine, Sergey, Aviral Kumar, George Tucker, and Justin Fu (2020). "Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems". In: *arXiv* 2005.01643 (cit. on p. 5).

Li, Gen, Yuting Wei, Yuejie Chi, and Yuxin Chen (2024). "Breaking the sample size barrier in model-based reinforcement learning with a generative model". In: *Operations Research* 72.1, pp. 203–221 (cit. on p. 4).

Li, Jinning, Chen Tang, Masayoshi Tomizuka, and Wei Zhan (2022). "Hierarchical planning through goal-conditioned offline reinforcement learning". In: *IEEE Robotics and Automation Letters* 7.4, pp. 10216–10223 (cit. on p. 68).

Li, Ming, Chen Shi, Zhize Wu, and Piotr Fryzlewicz (2025). "Testing Stationarity and Change Point Detection in Reinforcement Learning". In: *Annals of Statistics*. To appear (cit. on p. 94).

Liaw, Richard, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica (2018). "Tune: A Research Platform for Distributed Model Selection and Training". In: *arXiv* 1807.05118 (cit. on p. 64).

Lillicrap, Timothy P., Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra (2016). "Continuous control with deep reinforcement learning". In: *4th International Conference on Learning Representations*. (ICLR) (cit. on pp. 22, 59, 81, 115).

Lu, Cong, Philip J. Ball, Jack Parker-Holder, Michael A. Osborne, and Stephen J. Roberts (2022). "Revisiting Design Choices in Offline Model Based Reinforcement Learning". In: *10th International Conference on Learning Representations*. (ICLR) (cit. on p. 68).

Maehara, Takanori (2015). "Risk Averse Submodular Utility Maximization". In: *Operations Research Letters* 43.5, 526–529 (cit. on p. 37).

Mai, Vincent, Kaustubh Mani, and Liam Paull (2022). "Sample Efficient Deep Reinforcement Learning via Uncertainty Estimation". In: *10th International Conference on Learning Representations*. (ICLR) (cit. on p. 90).

Merlis, Nadav and Shie Mannor (2019). "Batch-Size Independent Regret Bounds for the Combinatorial Multi-Armed Bandit Problem". In: *32nd Annual Conference on Learning Theory*. (COLT) (cit. on p. 36).

Mishra, Nikita and Abhradeep Thakurta (2015). "(Nearly) optimal differentially private stochastic multi-arm bandits". In: *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*. (UAI) (cit. on p. 3).

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. (2015). "Human-level control through deep reinforcement learning". In: *Nature* 518.7540, pp. 529–533 (cit. on p. 4).

Modi, Aditya, Nan Jiang, Satinder Singh, and Ambuj Tewari (2018). "Markov decision processes with continuous side information". In: *29th International Conference on Algorithmic Learning Theory*. (ALT) (cit. on p. 120).

Nasiriany, Soroush, Vitchyr Pong, Steven Lin, and Sergey Levine (2019). "Planning with goal-conditioned policies". In: *Advances in Neural Information Processing Systems*. Vol. 32. (NeurIPS) (cit. on p. 68).

Ohsaka, Naoto and Yuichi Yoshida (2017). "Portfolio Optimization for Influence Spread". In: *Proceedings of the 26th International Conference on World Wide Web*. (WWW) (cit. on p. 37).

Osband, Ian, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy (2016). "Deep Exploration via Bootstrapped DQN". In: *Advances in Neural Information Processing Systems*. Vol. 29. (NeurIPS) (cit. on p. 90).

Pathak, Deepak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell (2017). "Curiosity-Driven Exploration by Self-Supervised Prediction". In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops*. (CVPRW) (cit. on p. 90).

Prudencio, Rafael Figueiredo, Marcos ROA Maximo, and Esther Luna Colombini (2023). "A survey on offline reinforcement learning: Taxonomy, review, and open problems". In: *IEEE Transactions on Neural Networks and Learning Systems* (cit. on p. 5).

Rudin, Nikita, David Hoeller, Philipp Reist, and Marco Hutter (2022). "Learning to walk in minutes using massively parallel deep reinforcement learning". In: *Conference on Robot Learning*. (CoRL) (cit. on p. 80).

Salter, Sasha, Markus Wulfmeier, Dhruva Tirumala, Nicolas Heess, Martin Riedmiller, Raia Hadsell, and Dushyant Rao (2022). "Mo2: Model-based offline options". In: *Conference on Lifelong Learning Agents*. (PMLR) (cit. on p. 68).

Sani, Amir, Alessandro Lazaric, and Rémi Munos (2012). "Risk-Aversion in Multi-Armed Bandits". In: *Advances in Neural Information Processing Systems*. Vol. 25. (NeurIPS) (cit. on pp. 3, 23, 36).

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov (2017). "Proximal policy optimization algorithms". In: *arXiv* 1707.06347 (cit. on p. 56).

Seno, Takuma and Michita Imai (2022). "d3rlpy: An offline deep reinforcement learning library". In: *Journal of Machine Learning Research* 23.315, pp. 1–20 (cit. on p. 82).

Seol, Youngsoo (2015). "Limit theorems for discrete Hawkes processes". In: *Statistics & Probability Letters* 99, pp. 223–229 (cit. on pp. 26, 100).

Settles, Burr (2011). "From Theories to Queries: Active Learning in Practice". In: *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*. (PMLR) (cit. on p. 6).

Sharpe, William F. (1966). "Mutual Fund Performance". In: *The Journal of Business* 39.1, pp. 119–138 (cit. on p. 3).

Shi, Chengshuai and Cong Shen (2021). "Federated multi-armed bandits". In: *Proceedings of the 35th AAAI Conference on Artificial Intelligence* (cit. on p. 3).

Smith, Matthew, Herke Hoof, and Joelle Pineau (2018). "An inference-based policy gradient method for learning options". In: *Proceedings of the 35th International Conference on Machine Learning*. (ICML) (cit. on pp. 5, 67).

Soma, Tasuku and Yuichi Yoshida (2021). "Online Risk-Averse Submodular Maximization". In: *Proceedings of the 30th International Joint Conference on Artificial Intelligence*. (IJCAI) (cit. on p. 37).

Song, Yunlong, Mats Steinweg, Elia Kaufmann, and Davide Scaramuzza (2021). "Autonomous drone racing with deep reinforcement learning". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*. (IROS). IEEE (cit. on p. 4).

Sriperumbudur, Bharath K, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, and Gert RG Lanckriet (2012). "On the empirical estimation of integral probability metrics". In: *Electronic Journal of Statistics* 6, pp. 1550–1599 (cit. on p. 124).

Sunberg, Zachary and Mykel Kochenderfer (2018). "Online algorithms for POMDPs with continuous state, action, and observation spaces". In: *Proceedings of the International Conference on Automated Planning and Scheduling*. (ICAPS) (cit. on p. 121).

Sutton, Richard S and Andrew G Barto (1998). *Reinforcement learning: An introduction*. MIT press Cambridge (cit. on p. 10).

Sutton, Richard S, Doina Precup, and Satinder Singh (1999). "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning". In: *Artificial intelligence* 112.1-2, pp. 181–211 (cit. on pp. 4, 51, 67).

Tamkin, Alex, Ramtin Keramati, Christoph Dann, and Emma Brunskill (2019). "Distributionally-aware exploration for cvar bandits". In: *NeurIPS 2019 Workshop on Safety and Robustness in Decision Making*. (NeurIPS Workshop) (cit. on p. 34).

Tennenholtz, Guy, Nadav Merlis, Lior Shani, Martin Mladenov, and Craig Boutilier (2023). "Reinforcement learning with history dependent dynamic contexts". In: *Proceedings of the 40th International Conference on Machine Learning*. (ICML) (cit. on pp. 94, 120).

Todorov, Emanuel, Tom Erez, and Yuval Tassa (2012). "Mujoco: A physics engine for model-based control". In: *2012 IEEE/RSJ international conference on intelligent robots and systems*. (IROS). IEEE (cit. on p. 79).

Verma, Arun, Manjesh Hanawal, Arun Rajkumar, and Raman Sankaran (2019). "Censored semi-bandits: A framework for resource allocation with censored feedback". In: *Advances in Neural Information Processing Systems*. Vol. 32. (NeurIPS) (cit. on p. 37).

Wang, Ruosong, Russ R Salakhutdinov, and Lin Yang (2020). "Reinforcement learning with general value function approximation: Provably efficient approach via bounded eluder dimension". In: *Advances in Neural Information Processing Systems*. Vol. 33. (NeurIPS) (cit. on p. 4).

Wang, Siwei and Wei Chen (2020). "Thompson Sampling for Combinatorial Semi-Bandits". In: *Journal of Machine Learning Research* 20, pp. 1–53 (cit. on p. 36).

Watkins, Christopher JCH and Peter Dayan (1992). "Q-learning". In: *Machine learning* 8.3, pp. 279–292 (cit. on p. 22).

Wei, Chen-Yu and Haipeng Luo (2021). "Non-stationary Reinforcement Learning without Prior Knowledge: an Optimal Black-box Approach". In: *34th Annual Conference on Learning Theory*. (COLT) (cit. on p. 118).

Wu, Yue, Shuangfei Zhai, Nitish Srivastava, Joshua M Susskind, Jian Zhang, Ruslan Salakhutdinov, and Hanlin Goh (2021). "Uncertainty Weighted Actor-Critic for Offline Reinforcement Learning". In: *Proceedings of the 38th International Conference on Machine Learning*. (ICML) (cit. on p. 90).

Xiao, Chenjun, Yifan Wu, Chen Ma, Dale Schuurmans, and Martin Müller (2019). "Learning to Combat Compounding-Error in Model-Based Reinforcement Learning". In: *NeurIPS 2019 Deep Reinforcement Learning Workshop*. (NeurIPS Workshop) (cit. on p. 122).

Yang, Xintong, Ze Ji, Jing Wu, Yu-Kun Lai, Changyun Wei, Guoliang Liu, and Rossitza Setchi (2021). "Hierarchical reinforcement learning with universal policies for multistep robotic manipulation". In: *IEEE Transactions on Neural Networks and Learning Systems*. (TNNLS) 33.9, pp. 4727–4741 (cit. on pp. 63, 65, 70).

Yin, Dong, Sridhar Thiagarajan, Nevena Lazic, Nived Rajaraman, Botao Hao, and Csaba Szepesvari (2023). "Sample Efficient Deep Reinforcement Learning via Local Planning". In: *arXiv* 2301.12579 (cit. on p. 90).

Yu, Tianhe, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn (2021). "COMBO: Conservative offline model-based policy optimization". In: *Advances in Neural Information Processing Systems*. Vol. 34. (NeurIPS) (cit. on p. 90).

Yu, Tianhe, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma (2020). "MOPO: Model-based offline policy optimization". In: *Advances in Neural Information Processing Systems*. Vol. 33. (NeurIPS) (cit. on pp. 5, 56, 67, 90).

Zhou, Wenxuan, Sujay Bajracharya, and David Held (2021). "Plas: Latent action space for offline reinforcement learning". In: *Conference on Robot Learning*. (CoRL) (cit. on p. 68).

Zhu, Anjie, Feiyu Chen, Hui Xu, Deqiang Ouyang, and Jie Shao (2021). "Empowering the diversity and individuality of option: Residual soft option critic framework". In: *IEEE Transactions on Neural Networks and Learning Systems*. (TNNLS) (cit. on pp. 5, 67).

Zhuang, Zifeng, Kun Lei, Jinxin Liu, Donglin Wang, and Yilang Guo (2023). "Behavior Proximal Policy Optimization". In: *11th International Conference on Learning Representations*. (ICLR) (cit. on pp. 82, 86).