Towards Data-efficient AI: Theoretical analysis and experimental validation of new exploration algorithms for Reinforcement Learning

Aya Kayal

A dissertation submitted in fulfillment
of the requirements for the degree of
Doctor of Philosophy
of
University College London.

Department of Electronic and Electrical Engineering
University College London

November 14, 2025

Abstract

Sequential decision-making is at the core of everyday human activities and complex real-world systems. Equipping machines with this capability has enormous implications for advancing Artificial Intelligence (AI) and developing autonomous systems that can reliably address real-world tasks, from robotics and healthcare to traffic management and large-scale information systems. Unlike machines, the human brain makes intelligent decisions in a remarkably data- and energy-efficient manner. A long-term objective of AI research, often associated with the vision of Artificial General Intelligence (AGI), is to approximate the human brain's capabilities, paving the way for more practical and sustainable AI systems.

Reinforcement Learning (RL) is the mathematical framework that enables machines to learn sequential decision-making through trial and error, mirroring aspects of human learning. While RL has driven many recent advances, training RL agents remains highly data- and compute-intensive—far from the efficiency of the human brain. A central bottleneck to data efficiency is exploration: how machines gather informative experiences to learn effective strategies. This thesis addresses the exploration challenge in RL, approaching it from both empirical and theoretical perspectives, bridging the gap between the two.

First, a proof-of-concept study is developed that provides a deeper understanding of how exploration bonuses shape the behavior of deep RL agents, yielding new empirical insights into the mechanisms driving exploration in practice. Second, a theoretical framework is introduced for the analytical study of RL in the kernel setting, leading to the development of provably efficient exploration algorithms with regret bounds that are tighter than existing approaches. Third, the study of exploration in Bayesian Optimization with preference-based feedback introduces a novel algorithm that, for the first time, achieves order-optimal sample complexity in this setting. Together, these contributions advance the development of sample-efficient decision-making algorithms, bringing AI systems closer to the remarkable efficiency of human learning.

Impact Statement

AI simulates human intelligence in machines, enabling them to learn from experience, make decisions, and perform tasks traditionally requiring human cognition. As a transformative force, AI has immense potential to reshape industries and society. Its rapid advancement is already driving meaningful progress across critical domains.

For instance, AI is revolutionizing healthcare by improving early diagnosis of diseases like breast cancer [1]. In education, AI-powered personalized learning platforms adapt content and pacing to individual student needs, offering crucial support for learners with ADHD, dyslexia, and other challenges [2]. In disaster prediction, AI aids seismologists in detecting early earthquake warning signs, safeguarding lives and infrastructure [3].

Yet, alongside these advances, AI faces fundamental challenges that limit its broader impact. Many AI systems demand vast amounts of data and computational resources. Among these, Reinforcement Learning (RL) stands out as a powerful yet especially data-hungry subfield of AI. While RL has shown great promise in solving sequential decision-making problems in domains such as robotics, personalized education, and healthcare, its practical impact remains constrained by inefficiency and high computational costs.

RL is notoriously data-inefficient, often requiring orders of magnitude more data than humans need to achieve comparable performance. This inefficiency slows progress toward Artificial General Intelligence (AGI) and raises sustainability concerns due to the substantial carbon footprint of prolonged training. Moreover, the high computational and financial costs limit RL's

accessibility for many researchers and practitioners.

This thesis tackles RL's data inefficiency by addressing the exploration challenge—a core mechanism determining how effectively an agent learns from interacting with its environment. We investigate exploration from both theoretical and empirical perspectives, answering two central questions:

How can the agent explore efficiently in any environment, and how can it determine whether sufficient exploration has been achieved?

Our research develops novel exploration strategies that significantly improve RL's data efficiency, bridging the gap between theory and real-world deployment. By enabling RL systems to learn effectively with limited data, this work expands their applicability in data-constrained domains while reducing computational burdens. Ultimately, these contributions mark a step toward more practical, inclusive, and environmentally responsible AI, unlocking RL's potential to drive progress in critical areas like healthcare, education, and beyond.

Acknowledgements

First and foremost, I thank God for His guidance and for granting me the strength, knowledge, and understanding to complete this thesis.

I would like to express my deepest gratitude to my PhD supervisor, Dr. Laura Toni. She has been an incredibly kind, supportive, and inspiring mentor. From the very beginning, she welcomed me into her group, gave me the freedom to pursue my ideas, and consistently encouraged me while providing valuable advice. Her faith in my potential helped me build confidence in myself and in my abilities as a researcher. She created a relaxed yet stimulating environment that minimized the stress of the PhD journey, while pushing me to grow into an independent thinker who can ask the right questions and reason scientifically. I am inspired by her passion for research and by her role as a successful woman in the field of AI. I am also sincerely grateful to Dr. Sattar Vakili, my internship mentor at MediaTek Research. It was a privilege to work with him—I learned enormously from his expertise in theoretical Reinforcement Learning and bandits. He not only shared his insights and perspectives generously but also mentored me with the care and rigor of an advisor, introducing me to the statistical foundations of machine learning and showing me how to approach problems rigorously. His guidance pushed me to be more efficient, productive, and ambitious, enabling me to publish at top AI conferences. Much of the technical content of this thesis would not have been possible without his input and support. I also wish to sincerely thank Dr. Alberto Bernacchia, director of AI research at MediaTek, whose thoughtful questions, constructive critiques, and expert feedback pushed me forward during my internship. I extend my thanks to Dr. Alexandru Cioba for collaborating with me on my last project at MediaTek, and to Dr. Da-Shan Shiu, managing director at MediaTek, for offering valuable perspective on how to present research ideas to industry.

I am also grateful to my colleague and co-author, Dr. Eduardo Pignatelli at UCL, with whom I shared many fruitful discussions on Reinforcement Learning. He contributed to the work presented in Chapter 4 and was always ready to provide technical advice throughout my PhD. I extend my gratitude to Dr. Alain Andrés Fernández for our early discussions on exploration in deep Reinforcement Learning and for sharing valuable resources, and to MSc student Corina Caraconcea for her assistance in experimental results presented in Chapter 4. I would further like to thank my second advisor, Dr. Ilija Bogunovic, for many enlightening conversations and for examining my transfer thesis.

The research presented in this thesis was funded by the UCL Dean's Prize and by the Institute of Communications and Connected Systems (ICCS) in the EEE Department. I remain indebted to the head of the ICCS group, Professor Izzat Darwazeh, for his generous funding, unwavering support, and for being the person I could always turn to when facing difficulties during my PhD. I am also grateful to Dr. Simon Banks, the faculty tutor, who also played a key role in facilitating my research funding in the EEE Department.

I am grateful as well for the brilliant researchers and supportive friends I was surrounded by at UCL. To my special colleagues and friends in the LASP group—Eduardo Pignatelli, Nagham Osman, Pedro Gomes, Keyue Jiang, and Alan Guedes—thank you for your feedback, opinions, and countless hours of discussion. You helped create a supportive research environment, and I am grateful for your friendship both inside and outside UCL. To Shyam Sundhar Ramesh, thank you for being an exceptional friend, for your care, understanding, and constant support. To Nagham Osman, thank you for your friendship, for sharing both the good and hard moments of life inside and outside the PhD, and for all our London hangouts over the past four years. To Fan Yang, thank you for all the fun, travels, tea breaks, and for being my handbook manual whenever I needed one. To Wasseem Ozan, thank you for your precious advices which shaped my decisions, including encouraging me to pursue an internship at MediaTek. To Mai Hawwa, whom I met toward the end of my PhD, thank you for your fun spirit and for being such a special person who made my final months much lighter.

To many other wonderful people with whom I had the pleasure of sharing moments during my PhD journey—Andrew, Amany, Xinyue, Vittorio, Josh, Reem, Haochen, Iman, Farshad, Maryam, Noora, David, Maria, Gianluca, Fabio, Seerat, Navin, Edo, Ashraf, Seongho, Zun, Yuanyuan, Ayan, and many others—thank you for the lunch breaks, dinners, celebrations, for listening through both struggles and successes, and for showing me that research is brighter with laughter, and shared experiences. You have been like family, offering kindness, encouragement, and support throughout this journey.

I would also like to thank my dear friend from outside UCL, Safa Osta, and the whole Osta family for being my family in London since my arrival. They welcomed me as one of their own, and I am infinitely grateful for that. I also thank my friend Hana Koscec, my ex-roommate, for her kindness, calm presence, and care. To my friend across the ocean, Joudi Hajar, thank you for sending me love, motivation, and support despite the distance.

Finally, and most importantly, I want to express my deepest love and gratitude to mom, dad, and my sisters—Mira, Dima, and Amal. You are my greatest blessing, and I cannot imagine this journey without you by my side. Your love, encouragement, and faith in me have been the foundation that kept me going, especially in the most difficult moments.

To my mother, I dedicate this thesis to you. Thank you for being my constant source of strength, my biggest cheerleader, and my role model. Your support and unwavering belief in me have inspired me more than words can capture. I am forever grateful for the example you set, both as a remarkable professor and as a loving woman.

Contents

List of Figures

List of Tables

Abbreviations

AC Actor-Critic

A2C Advantage Actor-Critic

A3C Asynchronous Advantage Actor-Critic

AGI Artificial General Intelligence

AI Artificial Intelligence

ALE Arcade Learning Environment

ANNs Artificial Neural Networks

APS Active Pretraining with Successor Feature

APT Active Pretraining

BO Bayesian Optimization

BOHF Bayesian Optimization from Human Feedback

BPE Batched Pure Exploration

BTL Bradley-Terry-Luce

BTM Beat the Mean

CNN Convolutional Neural Networks

DDQN Double Deep Q-Networks

DIAYN	Diversity is All You Need
DP	Dynamic Programming
DPO	Direct Preference Optimization
DQN	Deep Q-Networks
DSEE	Deterministic Sequencing of Exploration and Exploitation
E3B	Exploration via Elliptical Episodic Bonuses
EI	Expected Improvement
GAE	Generalized Advantage Estimation
GP	Gaussian Process
GP-ThreDS	Gaussian Process-Thresholded Domain Shrinking
GP-TS	Gaussian Process-Thompson Sampling
GP-UCB	Gaussian Process-Upper Confidence Bound
ICM	Intrinsic Curiosity Module
IR	Intrinsic Reward
LCB	Lower Confidence Bound
LLMs	Large Language Models
LP-GP-UCB	Local Polynomial Gaussian Process-Upper Confidence Bound
MAB	Multi-Armed Bandits
MC	Monte Carlo
MDP	Markov Decision Processes
MI	Mutual Information

MR-LPF	Multi-Round Learning from Preference-based Feedback
MVR	Maximum Variance Reduction
NGU	Never Give Up
NTK	Neural Tangent Kernel
OFU	Optimism in the Face of Uncertainty
OFUL	Optimism in the face of Uncertainty Linear bandits
PI	Probability of Improvement
POMDP	Partially Observable Markov Decision Processes
POP-BO	Principled Optimistic Preferential Bayesian Optimization
PPO	Proximal Policy Optimization
π-GP-UCB	Partitioned Improved Gaussian Process-Upper Confidence Bound
RE3	Random Encoders for Efficient Exploration
RELU	Rectified Linear Unit
RGB	Red, Green, Blue
RIDE	Reward Impact Driven Exploration
RIPS	Robust Inverse Propensity Score
RKHS	Reproducing Kernel Hilbert Spaces
RL	Reinforcement Learning
RLHF	Reinforcement Learning from Human Feedback
RM	Reward Model
RND	Random Network Distillation

RUCB	Relative Upper Confidence Bound
SE	Squared Exponential
SGD	Stochastic Gradient Descent
SMM	State Marginal Matching
SST	Strong Stochastic Transitivity
STI	Stochastic Triangle Inequality
SVGP	Sparse Variational Gaussian Processes
TD	Temporal Difference
TS	Thompson Sampling
UCB	Upper Confidence Bound
VF	Value Function

Chapter 1

Introduction

Artificial Intelligence (AI) is transforming nearly every facet of modern society, from healthcare and education to transportation and scientific discovery. Central to this transformation are AI agents—autonomous systems that perceive their environment, reason about tasks, and take actions to achieve specific goals. These agents leverage capabilities such as planning, memory, and adaptive learning, enabling them to improve over time through interaction with their surroundings.

This process of improvement, known as agent learning, allows AI systems to refine their decision-making, adapt to new conditions, and tackle increasingly complex challenges. While recent advances in AI have shown impressive capabilities, most AI systems today remain specialized and lack the flexibility required for broader problem-solving. They typically struggle to generalize knowledge, transfer skills between domains, or solve unfamiliar problems in the way humans can. Achieving Artificial General Intelligence (AGI)—where machines can learn, reason, and adapt as flexibly as humans—remains one of the field's most ambitious goals.

A critical component of AGI is sequential decision-making: the ability to reason over time, handle uncertainty, and operate in dynamic environments. The potential applications are vast. For instance, intelligent traffic management systems could proactively mitigate congestion. Personalized healthcare could continuously adapt treatment strategies to individual patients. Precision agriculture systems could optimize water usage, crop yields, and soil health by dynamically adjusting to weather patterns and plant needs. Yet replicating the depth and efficiency of human decision-making continues to pose a fundamental challenge for the advancement of AI systems. In fact, humans excel in tasks that demand causal reasoning, conceptual understanding, and generalization from limited experience—skills that enable adaptive behavior across diverse and uncertain environments [6, 7]. Developing computational approaches that approximate these abilities has therefore become central to AI research.

Reinforcement Learning (RL) plays a foundational role in this endeavor. RL is the computational framework through which agents learn to make sequential decisions by interacting with an environment and receiving evaluative feedback in the form of rewards [4]. Unlike supervised learning, where models learn from labeled datasets of correct input-output pairs, or unsupervised learning, where models discover patterns within unlabeled data, RL focuses on learning behavior through trial and error. An RL agent is not given explicit instructions on the correct action to take; instead, it must explore, observe the outcomes of its actions, and iteratively improve its decision-making policy to maximize long-term reward. This paradigm closely mirrors how humans and animals learn from experience.

RL has achieved remarkable success across a range of challenging domains. A notable milestone was reached by AlphaGo, a system that combines deep neural networks with advanced search algorithms, which defeated the world's top human players in the ancient game of Go—a game long considered resistant to traditional AI due to its immense combinatorial complexity [8]. Building on this progress, AlphaTensor used deep RL to autonomously discover faster algorithms for matrix multiplication, a fundamental operation in scientific computing [9]. RL has also been employed to control magnetic coils in tokamaks for plasma confinement, advancing nuclear fusion research [10]. More recently, RL has also found applications in training Large Language Models (LLMs); for instance, DeepSeek-R1 demonstrated how RL techniques can be used to enhance the reasoning capabilities of LLMs [11].

Despite its impressive successes, RL continues to face significant challenges—most notably in scalability, interpretability, safety, the sim-to-real gap, and data inefficiency [12], the latter of which is the primary focus of this thesis. RL agents typically require extensive interaction with the environment to learn effectively. In contrast to humans, who can learn from limited samples, RL systems often depend on millions of samples, making them highly data-inefficient [8, 12, 6]. This inefficiency hinders progress toward developing generalist agents, as most RL models are trained from scratch for each task and struggle to transfer knowledge across tasks or domains [13]. It also renders RL impractical in domains where data is expensive, risky, or ethically sensitive to collect—such as personalized medicine, autonomous driving, or education. RL's data inefficiency goes hand in hand with its heavy computational requirements, amplifying both cost and environmental impact. Training high-performing agents in complex environments often requires large-scale compute clusters over extended periods, leading to substantial energy consumption and carbon emissions. For example, training AlphaGo Zero is estimated to have produced approximately 96 tonnes of CO_2 over 40 days—equivalent to nearly $1{,}000$ hours of air travel [14]. These sustainability concerns are further intensified by the trend toward increasingly large models and datasets, as seen with LLMs whose compute budgets now reach tens of millions of dollars [15] and require millions of human annotations [16]. Moreover, the high costs of data collection, environment simulation, and compute infrastructure restrict broader access to RL research and applications.

Taken together, these challenges highlight an urgent need to develop data-efficient RL algorithms that deliver good performance while minimizing data requirements. In this thesis, we address this need by proposing several approaches to improve data efficiency in RL, enabling agents to learn more effectively with fewer interactions.

1.1 Main Challenges

As outlined in the introduction, data inefficiency poses a central obstacle to the broader applicability of RL—particularly in settings where data is limited. To address this problem, it is crucial to understand the underlying sources of inefficiency. In this section, we highlight three core challenges that contribute to poor sample efficiency in RL: exploration, credit assignment, and generalization.

Exploration. A major contributor to data inefficiency in RL is exploration, which plays a critical role in determining how efficiently an agent learns from interactions with the environment. Exploration has both theoretical and empirical dimensions. Theoretically, efficient exploration aims to guide the agent toward informative samples in order to minimize sample complexity—the number of interactions needed to learn an optimal policy. While this goal is well-understood and provably achievable in simple tabular [17, 18, 19] or linear [20, 21, 22, 23, 24] settings, it remains poorly understood in more complex environments with large or continuous state and action spaces.

Moreover, exploration strategies that are well understood in theory often fail to scale in practice. This gap between theoretical insight and empirical performance is particularly apparent in deep RL, where exploration remains one of the primary obstacles to sample-efficient learning. For example, in sparse-reward settings—where the agent receives little to no feedback until a task is completed—simple heuristics like ϵ-greedy often fail to reach rewarding states, leading to slow or unsuccessful learning [25]. This is known as the hard exploration problem, and it often results in either poor policy performance or the need for excessive interaction with the environment.

Bridging this theory-practice gap requires studying exploration from both theoretical and empirical perspectives, and remains a central challenge on the path toward data-efficient RL.

Credit Assignment. Another key factor contributing to data inefficiency in RL is the credit assignment problem. In complex environments, the consequences of an agent's actions are often delayed, making it difficult to identify which actions led to the observed outcomes [26]. This temporal gap between actions and their eventual rewards hinders learning, as the agent struggles to assign credit to the appropriate decisions. Without accurate credit assignment, the agent may require many episodes to discover beneficial actions. Ideally, an agent should be able to trace outcomes back to the key decisions that caused them—even across long time horizons—allowing it to update its policy more effectively with fewer interactions.

Generalization. Humans excel at retaining and reusing knowledge, even in tasks that differ substantially from past experiences [6, 7]. In contrast, most RL agents are trained to solve a single, well-defined task in a stationary environment. This often leads to overfitting, where the agent performs well in the training environment but fails to generalize beyond it. As a result, even minor changes—such as variations in the environment, distributional shifts, or task perturbations—can severely degrade performance. RL agents typically lack the ability to transfer previously learned knowledge or skills to novel tasks or dynamic environments. This inability to generalize limits the practical applicability of RL in real-world settings, where environments are often noisy, non-stationary, and subject to unexpected changes (e.g., modeling errors, reward misspecification, or adversarial attacks). To be truly data-efficient, RL agents must be able to reuse prior experience, remain robust under uncertainty, and adapt to new scenarios instead of tackling them from scratch [27].

Motivated by these challenges, this thesis presents several contributions aimed at improving data efficiency in RL, with a particular focus on the first challenge: exploration, examined from both theoretical and empirical perspectives. Specifically, we investigate the following high-level research question:

How can the agent explore efficiently in any environment, and how can it determine whether sufficient exploration has been achieved?

1.2 Contributions

The key contributions of this thesis can be summarized as follows:

1. **Conduct a proof-of-concept study of intrinsic rewards for exploration in deep RL:** The first part of our research question–"How can the agent explore efficiently in any environment?" is addressed from an empirical perspective in Chapter 4. We focus on intrinsic rewards, a prominent class of exploration strategies for tackling hard exploration problem. Despite their widespread use in deep RL, there is little consensus on which intrinsic rewards are most effective in different scenarios. To clarify this, we reinterpret intrinsic rewards through the lens of diversity, classifying them based on the level of diversity they promote in the agent's behavior. We conduct an empirical study to compare different intrinsic rewards across various RL environments and exploration metrics.

2. **Provide empirical insights into the role of diversity in exploration:** From our proof-of-concept study, we analyze how different levels of diversity in exploration behavior influence the efficiency of exploration. Our results offer practical guidance on how to tailor exploration strategies to specific tasks and environments, as presented in Chapter 4 of the thesis.

3. **Design exploration algorithms for reward-free kernel-based RL with theoretical guarantees and empirical validation:** The second part of our research question—"How can the agent determine whether sufficient exploration has been achieved"— is addressed theoretically in Chapter 5. While RL theory is well-established in tabular and linear settings, it is less developed for deep RL. To bridge this gap, we focus on kernel methods, which provide a middle ground between simple models and deep learning. Chapter 5 develops a rigorous theoretical framework for the analysis of kernel-based RL. Within this framework, we propose

novel exploration algorithms that collect unbiased samples and establish tighter sample complexity bounds over a broad class of kernels, supported by synthetic experiments.

4. **Introduce a novel confidence interval for unbiased samples in kernel-based RL:** In Chapter 5, we propose a new confidence interval for kernel ridge regression in the RL setting. This confidence interval underpins the theoretical guarantees of our exploration algorithms and may be broadly applicable to other RL scenarios, including offline RL, model-based RL, and infinite-horizon problems.

5. **Propose a novel algorithm for efficient exploration under preference-based feedback:** We extend the thesis's central theme of efficient exploration to settings where feedback is relative and limited. Motivated by applications such as prompt optimization, we study exploration in the context of preference-based feedback, where agents only observe comparisons between outcomes (e.g., "A is preferred to B") rather than numerical rewards. We formulate this as a preference-based Bayesian Optimization (BO) problem and propose Multi-Round Learning from Preference-based Feedback (MR-LPF) algorithm, which iteratively selects action pairs based on the highest uncertainty in their preference, achieving order-optimal sample complexity. This work appears in Chapter 6.

6. **Establish theoretical guarantees and practical utility for MR-LPF algorithm:** In Chapter 6, we prove regret bounds and derive sample complexity results showing that our MR-LPF algorithm matches conventional BO bounds with scalar feedback despite relying only on preference queries. We further validate the algorithm on synthetic and real-world datasets, demonstrating its practical effectiveness.

1.3 Thesis outline

This thesis consists of seven chapters. Following this introductory chapter, the remainder of the document is organized as follows:

1. Chapter 2 provides the necessary background on RL, covering foundational concepts such as Markov Decision Processes (MDPs), value functions, model-based and model-free approaches. This chapter also introduces key algorithms and function approximation techniques, including deep RL and policy gradient methods. It lays the groundwork essential for understanding the methods developed in subsequent chapters.

2. Chapter 3 introduces the multi-armed bandit framework and explores various extensions, including linear bandits, Gaussian Process bandits (also knows as Bayesian Optimization), and dueling bandits, along with associated algorithms and theoretical guarantees. This chapter provides essential theoretical background for understanding exploration-exploitation trade-offs in simpler settings compared to RL such as bandits problems.

3. Chapter 4 presents an empirical study on the impact of diversity on exploration in deep RL, serving as a preliminary investigation into the complex relationship between diversity and exploration. It introduces a proposed taxonomy of diversity levels induced by intrinsic rewards, conducts a comparative empirical analysis, and evaluates multiple exploration metrics. The chapter concludes with a discussion of the findings, offering practical insights into the effectiveness of different intrinsic rewards.

4. Chapter 5 introduces novel exploration algorithms for reward-free RL, both with and without access to a generative model. It also provides theoretical analyses of their sample complexity and validate the improved bounds through synthetic experiments.

5. Chapter 6 introduces the Bayesian Optimization from Human Feedback (BOHF) framework and proposes the novel MR-LPF algorithm. A theoretical analysis of the algorithm is provided, and its performance is evaluated on various test functions, including both synthetic and real-world cases.

6. Chapter 7 concludes the thesis with a concise summary of the main contributions and outlines important directions for future research.

7. The appendices include proofs of the theoretical results presented in the above chapters, as well as supplementary experimental details and plots.

1.4 Publications

The research conducted in the course of this dissertation has led to the following publications:

- **Aya Kayal**, Eduardo Pignatelli, and Laura Toni. "Does behavioral diversity in intrinsic rewards help exploration?" In *NeurIPS 2023 Second Agent Learning in Open-Endedness Workshop*, 2023.

- **Aya Kayal**, Eduardo Pignatelli, and Laura Toni. "The impact of intrinsic rewards on exploration in reinforcement learning". *Neural Computing and Applications*, 37:16269–16303, 2025.

- **Aya Kayal**, Sattar Vakili, Laura Toni, and Alberto Bernacchia. "Near-optimal sample complexity in reward-free kernel-based reinforcement learning". In *Proceedings of the 28th International Conference on Artificial Intelligence and Statistics,* volume 258. PMLR, 2025.

- **Aya Kayal**, Sattar Vakili, Laura Toni, Da-Shan Shiu and Alberto Bernacchia. "Bayesian optimization from human feedback: near-optimal regret bounds". In *Proceedings of the International Conference on Machine Learning,* volume 267. PMLR, 2025.

Chapter 2

Background on Reinforcement Learning

In RL, an artificial agent interacts with an unknown environment to learn how to solve a task. As illustrated in Figure 2.1, RL consists of two primary components: the agent and the environment. During each interaction cycle, the agent observes the current state of the environment and selects an action. In response to the agent's action, the environment transitions to a new state according to its internal dynamics, potentially yielding a reward. The agent receives this reward as feedback, which it uses to adjust its behavior and improve future decision-making. Over time, the agent aims to maximize the cumulative reward by learning an optimal strategy through repeated interactions.

The key elements of an RL problem include the policy, value function, and reward. The policy determines the agent's action based on the current state. Rewards provide immediate feedback, shaping the agent's learning process by signaling the desirability of specific actions. Value functions estimate the expected long-term return from a given state, accounting for future rewards and transitions. Unlike in supervised learning, where the correct output is explicitly provided, RL agents learn through trial and error—exploring different actions and refining their decision-making strategies based on the observed outcomes. This process enables the agent to learn the optimal policy, defined as the policy that maximizes the value function.

Formally, RL is grounded in the framework of Markov Decision Processes (MDPs), which model the agent-environment interaction in a probabilistic manner. In this chapter, we theoretically introduce MDPs and key concepts such as value functions and Bellman optimality equations, which are essential for evaluating and optimizing policies. We then present the major categories of RL problems: model-free vs. model-based, each offering distinct strategies for learning optimal behavior. We explain the core ideas and principles behind how these algorithms learn from experience and improve their decision-making over time.

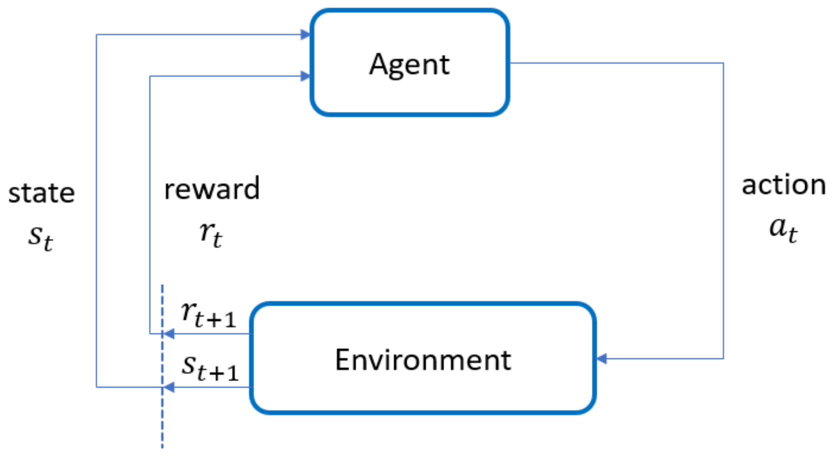


Figure 2.1: RL agent-environment interaction [4].

2.1 Markov Decision Processes

MDPs provide a foundational framework for modeling sequential decision-making under uncertainty [28, 4]. The most common formulation used in pratical RL is the *infinite-horizon discounted MDP*, which captures ongoing interaction between an agent and its environment over time. It is defined by the tuple:

$$M = \big(\mathcal{S}, \mathcal{A}, P, r, \gamma\big),$$

where:

- $\mathcal{S}$ is a set of states,

- $\mathcal{A}$ is a set of actions,
- $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ specifies the transition dynamics,
- $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the expected immediate reward function, and
- $\gamma \in [0,1)$ is the discount factor.

The transition dynamics function P gives the probability of moving from state s to state s' when action a is taken:

$$P(s' \mid s,a) = \mathbb{P}\{s_{t+1} = s' \mid s_t = s, a_t = a\}.$$

The expected immediate reward is defined as:

$$r(s,a) = \mathbb{E}[r_{t+1} \mid s_t = s, a_t = a],$$

where r_{t+1} is the instantaneous immediate reward received at the next time step. The discount factor γ determines how future rewards are weighted: when $\gamma = 0$, only immediate rewards are considered, whereas as $\gamma \to 1$, future rewards are valued more heavily.

At each discrete time step t, the agent observes the current state $s_t \in \mathcal{S}$, selects an action $a_t \in \mathcal{A}$ according to a policy π, transitions to a new state $s_{t+1} \sim P(\cdot \mid s_t, a_t)$, and receives reward r_{t+1}. The policy π is a mapping that, for each state s, assigns to each action $a \in \mathcal{A}$ the probability of taking that action in state s. It is denoted by $\pi(a \mid s) = \mathbb{P}\{a_t = a \mid s_t = s\}$. A policy can either be stochastic, in which case multiple actions may have nonzero probability, or deterministic, in which case a single action has probability 1, often denoted simply by $\pi(s)$ to indicate the action chosen in state s.

The Markov Property. A key assumption in MDPs is the *Markov property*, which states that the future is conditionally independent of the past given the present. Formally,

$$P(s_{t+1} \mid s_t, a_t) = P(s_{t+1} \mid s_0, a_0, \ldots, s_t, a_t),$$

meaning that the current state s_t contains all relevant information for predicting future transitions and making optimal decisions.

MDPs can be categorized along several dimensions:

- **Horizon**:
  - *Finite-horizon*: each trajectory (of state-action-reward tuples) has a fixed length of exactly H steps,
  - *Indefinite-horizon*: each trajectory has finite but random length,
  - *Infinite-horizon*: trajectories continue indefinitely.
- **Discounting**:
  - *Discounted*: $\gamma < 1$, future rewards are discounted,
  - *Undiscounted*: $\gamma = 1$, all rewards are treated equally.
- **Stationarity**:
  - *Stationary*: transition dynamics and rewards are time-invariant (do not depend on time),
  - *Non-stationary*: they vary across time steps.

The infinite-horizon discounted setting is more commonly used in the applied RL community due to its simplicity and real-world relevance. It guarantees the existence of a stationary optimal policy (one that does not depend on time), which simplifies the algorithm design and implementation. Discounting also naturally aligns with economic and engineering applications, where future rewards are uncertain. In contrast, the theoretical RL community has largely focused on the finite-horizon undiscounted (also known as *episodic*) setting, primarily due to the persistent challenges in analyzing infinite-horizon MDPs. Episodic problems offer greater tractability in proofs, enabling cleaner regret bounds—formal guarantees on the cumulative difference between an algorithm's total reward and that of the best possible performance—without the complications introduced by discounted or unbounded time horizons.

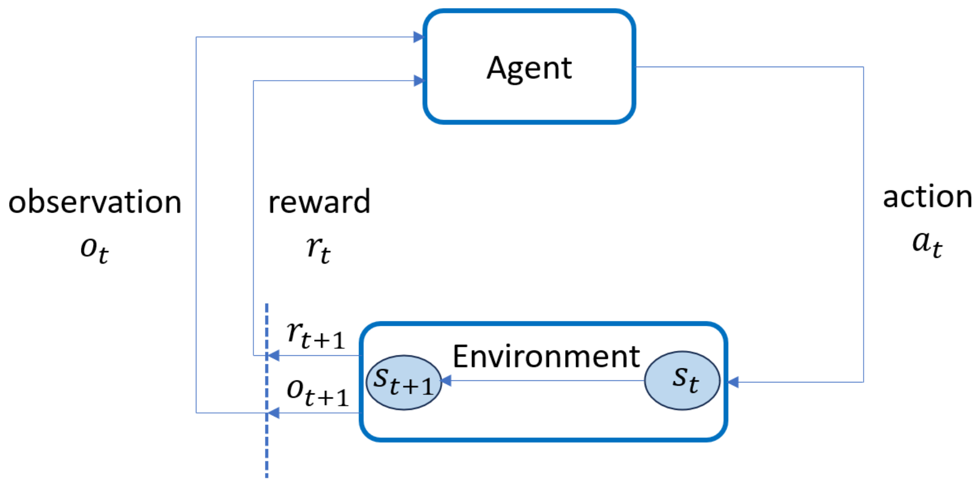


Figure 2.2: POMDP agent-interaction with the environment [5].

In the episodic formulation, the MDP is given by:

$$M = \left(\mathcal{S}, \mathcal{A}, \{P_h\}_{h=1}^{H}, \{r_h\}_{h=1}^{H}, H\right),$$

where H is the fixed episode length. The transition dynamics P_h and reward functions r_h may vary with the time index h, which denotes the position within an episode (analogous to the time step t in the infinite-horizon setting, but ranging from 1 to H). This formulation allows for *non-stationarity*, where the dynamics and rewards can change across the episode. A more detailed description of the episodic MDP framework is provided in Section 5.3.1 of Chapter 5.

2.1.1 Partially Observable Markov Decision Processes

The framework described above assumes that the environment's states are fully observable. However, this assumption is often unrealistic in real-world scenarios, where certain aspects of the environment may be hidden from the agent or affected by sensor noise. For example, in robotics applications, a robot's sensors may not precisely capture its exact location. This motivates the use of Partially Observable Markov Decision Processes (POMDPs), where the agent does not have direct access to the true state but instead receives partial information through observations. Figure 2.2 illustrates this concept.

Formally, a POMDP is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, P, r, \gamma, \Omega, O \rangle$, where $\langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$ defines the underlying fully observable MDP, Ω is a finite set of possible observations, and O is the observation function $O : \mathcal{S} \times \mathcal{A} \times \Omega \rightarrow [0,1]$, which gives the probability of observing $o \in \Omega$ after taking action $a \in \mathcal{A}$ and arriving in state $s' \in \mathcal{S}$, denoted as $O(o \mid s', a)$ [29]. Although the agent cannot directly observe the true state, it uses the received observations to infer a belief over possible states and selects actions to maximize the expected cumulative discounted reward over time.

2.2 Value Functions and Bellman Optimality

This section adopts the notation and framework presented in [4], which assumes an infinite-horizon discounted MDP with stationary transition dynamics and reward functions. The agent's objective is to maximize the expected discounted return over time. While this setting is widely used in practical RL, the core ideas—such as value functions and Bellman optimality—are general and can be readily applied to other MDP formulations. For example, they naturally extend to finite-horizon settings (where $T = H$) and undiscounted problems (where $\gamma = 1$). Let's start by defining the expected discounted return G_t defined as:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots + \gamma^{T-t-1} r_T = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1}. \tag{2.1}$$

A value function $V^\pi(s)$, defined with respect to policy π, estimates the expected return from a particular state s following π, and it is defined as:

$$V^\pi(s) = \mathbb{E}_\pi[G_t | s_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1} | s_t = s \right], \text{ for all } s \in \mathcal{S}, \tag{2.2}$$

where G_t is the discounted total return and r_{t+k+1} are immediate rewards. Similarly, the state-action value function is defined as the expected return from state s, taking action a and following policy π:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[G_t | s_t = s, a_t = a] = \mathbb{E}_{\pi}\left[\sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right]. \quad (2.3)$$

It is worth noting that the state value function V and the state-action value function Q are strictly correlated by the following relationship:

$$V^{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) Q^{\pi}(s,a). \quad (2.4)$$

A central recursive relationship in RL which relates the value of a state to the value of its successor states is the Bellman equation:

$$V^{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a|s)\Big[r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V^{\pi}(s')\Big]. \quad (2.5)$$

Value functions define a partial ordering over policies. A policy π is better than another policy π' if it has a higher expected return for all states, which means a higher value function. In other terms:

$$\pi \geq \pi' \Leftrightarrow V^{\pi}(s) \geq V^{\pi'}(s), \forall s \in \mathcal{S}. \quad (2.6)$$

Thus, there is always at least one policy that is better than or equal to all other policies. Such a policy is called an optimal policy, denoted by $\pi^{\star}$. Although multiple optimal policies may exist, they all share the same optimal state value function $V^{\star}$, which gives the maximum expected return achievable from each state:

$$V^{\star}(s) = \max_{\pi} V^{\pi}(s), \forall s \in \mathcal{S}. \quad (2.7)$$

Similarly, the optimal state-action value function $Q^{\star}$ gives the maximum expected return achievable from each state-action pair:

$$Q^{\star}(s,a) = \max_{\pi} Q^{\pi}(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}. \quad (2.8)$$

Intuitively, the optimal value function $V^\star$ is the state-action value function with the best action from that state. It also satisfies the recursive property of the Bellman equation (2.5). The Bellman equation for $V^\star$, known as the Bellman optimality equation, can be written as:

$$\begin{aligned} V^\star(s) &= \max_a Q^{\pi^\star}(s,a) \\ &= \max_a \mathbb{E}_{\pi^\star}\Big[G_t | s_t = s, a_t = a\Big] \\ &= \max_a \mathbb{E}_{\pi^\star}\Big[r_{t+1} + \gamma V^\star(s_{t+1}) | s_t = s, a_t = a\Big] \\ &= \max_a \Big[r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V^\star(s')\Big]. \end{aligned} \tag{2.9}$$

The state-action value version of the Bellman optimality equation is:

$$Q^\star(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) \max_{a' \in \mathcal{A}} Q^\star(s',a'). \tag{2.10}$$

Once the optimal value function or state-action value function is solved, the optimal policy $\pi^\star$ can be determined:

$$\pi^\star(s) = \arg\max_{a \in \mathcal{A}} Q^\star(s,a). \tag{2.11}$$

Solving the Bellman optimality equation requires accurate knowledge of the environment's dynamics, as well as substantial memory and computational resources. These assumptions are rarely met in practice; therefore, RL techniques aim to implement approximate solutions to the Bellman optimality equation.

2.3 Model-Based vs Model-Free RL

In model-based RL, the agent either has access to a model of the environment or learns one through interaction. By "model", we refer to the transition dynamics and reward function, which together allow the agent to predict the consequences of its actions. Model-based methods typically involve two components: model learning, where the agent builds an estimate of the environment, and planning, where it uses this model to compute value functions or improve its policy through simulated experience.

In contrast, model-free RL assumes no prior or learned knowledge of the environment's dynamics or reward function. Instead, the agent interacts directly with the environment—selecting actions, observing state transitions, and receiving rewards—to estimate value functions or learn policies through trial-and-error. This approach forgoes model construction and planning, simplifying the learning process at the cost of typically requiring more experience.

One key distinction between these paradigms lies in their use of experience. Model-based methods often make fuller use of limited interaction data by generating simulated experience from the model, enabling more rapid policy improvement with fewer environmental interactions [4]. However, these benefits come with trade-offs. Model-free methods are generally simpler and not subject to modeling errors. In contrast, model-based methods depend on the quality of the learned model. Inaccuracies in the model can introduce bias, leading to compounding errors during planning and ultimately degrading the agent's performance when deployed [4].

2.4 Model-Based Learning: Dynamic Programming

When the environment model is fully known, as in classical planning settings, Dynamic Programming (DP) provides foundational algorithms for solving MDPs. Two canonical DP algorithms are policy iteration and value iteration. These algorithms operate by iteratively refining value functions and policies based on the known environment dynamics.

2.4.1 Policy Iteration

In policy iteration, the agent alternates between two steps: policy evaluation and policy improvement. During policy evaluation, the value function is updated for the current policy until convergence. In the improvement step, a new policy is derived by acting greedily with respect to the updated value function. This process continues until the policy stabilizes. For a pseudocode, please see Algorithm 1.

Algorithm 1 Policy Iteration

```
Initialization:
    Initialize $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}$ arbitrarily for all $s \in \mathcal{S}$
Policy Evaluation:
repeat
  $\Delta \leftarrow 0$
  for each $s \in \mathcal{S}$ do
    $v \leftarrow V(s)$
    $V(s) \leftarrow r(s,\pi(s)) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s,\pi(s)) V(s')$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  end for
until $\Delta < \theta$ {a small positive number}
Policy Improvement:
policy-stable ← true
for each $s \in \mathcal{S}$ do
  $a \leftarrow \pi(s)$
  $\pi(s) \leftarrow \arg\max_{a \in \mathcal{A}} \left[ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s,a) V(s') \right]$
  if $a \neq \pi(s)$ then policy-stable ← false
end for
if policy-stable then return $V$ and $\pi$ else go to Policy Evaluation
```

2.4.2 Value Iteration

An important special case occurs when only a single sweep over all states is performed. This leads to the value iteration algorithm (see Algorithm 2), which merges policy evaluation and policy improvement into a single update.

2.5 Model-Free Prediction

Two foundational classes of model-free prediction methods are Monte Carlo (MC) and Temporal Difference (TD) learning. Both aim to estimate the value function, but they differ in how they use sampled data and when they update their estimates.

Algorithm 2 Value Iteration

```
Initialize $V(s)$ arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}$)
repeat
  $\Delta \leftarrow 0$
  for each $s \in \mathcal{S}$ do
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \left[ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s,a) V(s') \right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  end for
until $\Delta < \theta$ {a small positive number}
Output: A deterministic policy $\pi$ such that:
```

$$\pi(s) = \arg\max_a \left[r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s,a) V(s') \right]$$

2.5.1 Monte Carlo Methods

MC methods learn directly from experience by collecting complete episodes—that is, trajectories of states, actions, and rewards from the beginning to the end of an episode. The underlying idea is straightforward: the value of a state is estimated by averaging the empirical returns (i.e., discounted cumulative rewards) observed after each visit to that state. As the number of sampled episodes increases, this estimate converges to the true expected value function. Let $G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots + \gamma^{T-t-1} r_T$ denote the return following time step t until the terminal step T. In MC methods, the value function is updated according to:

$$V(s_t) \leftarrow V(s_t) + \alpha(G_t - V(s_t)), \tag{2.12}$$

where α is the learning rate (step size).

2.5.2 Temporal Difference Learning

TD learning employs the concept of *bootstrapping*, where the value estimate of a state is updated based on the estimated value of its successor state. In other words, TD learning updates $V(s_t)$ toward an estimated return $G_t = r_{t+1} + \gamma V(s_{t+1})$. After each time step, the agent performs the following update:

$$V(s_t) \leftarrow V(s_t) + \alpha \left(r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \right). \tag{2.13}$$

The term $r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$ is known as the *TD error*, and the objective of TD learning is to minimize this error.

Unlike MC methods, which must wait until the end of an episode to compute returns, TD learning updates value estimates online at each step, without requiring the final outcome. This allows TD to update more efficiently. TD methods generally exhibit lower variance but higher bias compared to MC methods. The higher bias arises because TD uses the estimate $V(s_{t+1})$ instead of the actual return. Its lower variance stems from the fact that the TD target $r_{t+1} + \gamma V(s_{t+1})$ is based on a single reward and a learned estimate, rather than a complete return composed of many random variables. This reduces the variability introduced by long-term stochastic transitions.

Two of the most widely used TD algorithms are SARSA and Q-learning. SARSA is an *on-policy* method, whereas Q-learning is *off-policy*. The distinction between these two learning paradigms is as follows:

- **On-policy learning** evaluates and improves the same policy that is used to generate experience. That is, it learns about policy π using samples collected from π.
- **Off-policy learning** evaluates and improves a target policy π using data collected from a different behavior policy π'. In other words, it learns about π from experiences generated by π'.

SARSA updates the state-action value function using the TD error while following the current policy π (e.g., an ϵ-greedy policy derived from Q). The update rule is:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \Big[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \Big]. \tag{2.14}$$

The main difference between SARSA and Q-learning lies in how they compute the value of the next state-action pair. SARSA uses the action a_{t+1} sampled from the current policy, while Q-learning uses the greedy action that maximizes the Q-function at the next state. The Q-learning update rule is:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right]. \tag{2.15}$$

SARSA is generally more conservative than Q-learning. For example, in scenarios where a large penalty exists near the optimal path, SARSA tends to avoid it and prefer safer actions, whereas Q-learning aggressively learns the optimal (but potentially riskier) policy.

2.6 Function Approximation in RL

The methods discussed in Sections 2.5.1 and 2.5.2—namely MC and TD learning—use explicit lookup tables to represent value functions, maintaining a separate entry for every state or state-action pair. While effective in small environments, this approach becomes impractical as the dimensionality grows, due to the computational and memory demands that scale poorly—a problem known as the curse of dimensionality. To overcome this, function approximation techniques have been introduced [30], which estimate the value function with a parameterized function. For instance, instead of directly storing $V(s)$, these methods learn an approximation $V_w(s)$, where w is a vector of weights. Similarly, action-value functions can be approximated as $Q_w(s,a)$. This approach offers two major advantages. First, it significantly reduces memory requirements by representing the value function compactly through a set of parameters. Second, it enables generalization across states: the function can provide reasonable estimates for states that have not been explicitly visited during training. This is particularly important in environments where the state space is continuous or extremely large. In the following, we will briefly introduce methods for function approximation, including linear function approximation, kernel-based function approximation and deep neural networks (deep RL).

2.6.1 Linear Function Approximation

A fundamental and widely studied case of function approximation in RL is *linear function approximation*, where the approximate value function is expressed as a linear function of the weight vector w [4]. For each state s, we define a *feature vector* $\phi(s) = [\phi_1(s), \phi_2(s), \ldots, \phi_d(s)]^\top$, where d is the dimension of the feature space. The approximate state value function is then defined as:

$$V_{\mathbf{w}}(s) = \mathbf{w}^\top \phi(s) = \sum_{i=1}^{d} w_i \phi_i(s)$$

Each component $\phi_i(s)$ corresponds to a real-valued *feature function*, and together, these functions act as *basis functions* [4].

Similarly, when approximating the state-action value function, we define a joint feature vector $\phi(s,a)$, and write the approximation as:

$$Q_{\mathbf{w}}(s,a) = \mathbf{w}^\top \phi(s,a) = \sum_{i=1}^{d} w_i \phi_i(s,a)$$

Constructing the feature vectors $\phi(s)$ or $\phi(s,a)$ is equivalent to selecting a set of basis functions, which plays a crucial role in the performance of linear methods. There is a long line of classical work on RL with linear function approximation, encompassing both diverse strategies for constructing feature vectors and a variety of algorithms for learning the weight vector w; we refer the reader to [4] for further reading on the subject.

2.6.2 Kernel-Based Reinforcement Learning

Kernel-based methods form a powerful class of function approximation techniques in RL, extending linear models to infinite-dimensional Reproducing Kernel Hilbert Spaces (RKHS) induced by positive definite kernels. These methods serve as a conceptual bridge between well-understood linear approaches and more complex neural-network-based models, particularly in light of recent insights from Neural Tangent kernel (NTK) theory [31]. Their appeal lies in combining expressive nonlinear representation capacity with strong theoretical foundations, making them especially suitable for value function approximation in RL [32] .

To illustrate this, consider the common value estimation task in RL: computing the expected value of the next state given a current state-action pair. Formally, we define the target function as $f(s,a) = \mathbb{E}_{s' \sim P(\cdot|s,a)}[V(s')]$. To approximate f, we introduce a kernel function $k : (\mathcal{S} \times \mathcal{A}) \times (\mathcal{S} \times \mathcal{A}) \to \mathbb{R}$ that captures similarity between state-action pairs. This kernel implicitly defines a feature map ϕ into a high- or infinite-dimensional RKHS $\mathcal{H}_k$, such that

$$k((s,a),(s',a')) = \langle \phi(s,a), \phi(s',a') \rangle_{\mathcal{H}_k}.$$

In this lifted feature space, we can apply linear methods, even though the resulting approximation is nonlinear in the original state-action space. This makes kernel methods particularly attractive in RL, since we often expect similar state-action pairs to yield similar future returns.

Given a dataset of n transitions $\{(s_i, a_i), s'_i\}_{i=1}^n$, where $s'_i \sim P(\cdot \mid s_i, a_i)$, kernel ridge regression [33] provides a principled way to estimate $f(s,a)$. Formally, the kernel ridge regression solution minimizes the regularized empirical risk:

$$\hat{f} = \arg\min_{f \in \mathcal{H}_k} \sum_{i=1}^{n} \Big(f(s_i, a_i) - V(s'_i) \Big)^2 + \tau^2 \|f\|^2_{\mathcal{H}_k},$$

where $\tau^2 > 0$ is a regularization parameter and $||.||_{\mathcal{H}_k}$ denotes the RKHS norm.

The key advantage of the kernel function is that it enables generalization to unseen state-action pairs by assuming that similar inputs should produce similar outputs. If k is smooth (like a Squared Exponential (SE) kernel), the learned function will also be smooth, meaning small changes in the input lead to small changes in the prediction.

A more detailed treatment of kernel ridge regression—including its connections to Bayesian Optimization (BO) and Gaussian Process (GP) regression—is deferred to Section 3.3.3. In the RL context, we focus on its role in value function approximation, with the explicit form of the estimator and its associated uncertainty quantification presented in Section 5.3.3.

2.6.3 Deep Reinforcement Learning

Deep RL refers to the use of Artificial Neural Networks (ANNs) for nonlinear function approximation in RL. By leveraging the representational power of neural architectures, deep RL extends the capabilities of linear and kernel-based methods, enabling the modeling of complex, high-dimensional patterns. ANNs support automatic feature extraction directly from raw input data, which is especially beneficial in environments with unstructured or high-dimensional observations. Recent advancements in deep neural network training have driven significant progress in RL, leading to notable empirical successes across diverse domains—including game playing [8, 34, 35], robotic control [36], autonomous driving [37], microchip design [38], and algorithmic problem solving [9]. Despite these achievements, deep RL methods remain theoretically challenging to analyze due to their high model capacity and the non-convex nature of their optimization landscapes.

In the next paragraph, we describe Deep Q-Networks (DQN), a foundational algorithm in deep RL, first introduced by [39], using a deep neural network to approximate the state-action value function. DQN marked a major milestone by demonstrating the powerful synergy between RL and modern deep learning techniques.

Deep Q-Networks. DQN approximate the state-action value function $Q_w(s,a)$ using a deep neural network parameterized by w, which takes a state s as input and outputs a value for each possible discrete action a. Unlike supervised learning, the target values used to train the network are not fixed ground-truth labels but are instead bootstrapped from the network's own prior estimates. This reliance on self-generated targets gives rise to *semi-gradient descent*, where the target is treated as fixed and the gradient is taken only with respect to the parameters of the current Q-network. The network is trained to minimize the TD error, leading to the following update rule:

$$w \leftarrow w + \alpha \left[r_{t+1} + \gamma \max_{a'} Q_w(s_{t+1}, a') - Q_w(s_t, a_t) \right] \nabla_w Q_w(s_t, a_t). \tag{2.16}$$

The original DQN algorithm by [39], combined RL with convolutional neural networks (CNNs) to learn directly from high-dimensional sensory inputs, such as raw pixels in Atari games. It achieved superhuman performance on many games but exhibited instability during training due to the non-stationarity of targets and strong correlations in sequential data.
Several improvements were proposed to address these issues, including:

- **Target Network:** A separate, periodically updated network $\tilde{Q}$ is used to compute target values, which stabilizes learning by decoupling the target calculation from the current network parameters. The target network is synchronized with the main network every fixed number of steps: $\tilde{Q} \leftarrow Q$.

- **Double Q-Learning:** To mitigate overestimation bias in Q-values, Double DQN uses the *main network* to select the best action in the next state, but evaluates that action using the *target network*:

  $$y = r_{t+1} + \gamma \tilde{Q}_{\tilde{w}}(s_{t+1}, \arg\max_a Q_w(s_{t+1}, a)).$$

  This separation improves the accuracy of value estimates and contributes to more stable learning.

In the following, we present the **Double DQN algorithm** (see Algorithm 3), which incorporates both the target network and Double Q-learning to improve training stability and reduce overestimation errors.

2.7 Policy-Based Methods

The algorithms presented in the previous sections focused primarily on learning the value function (VF). In this section, we introduce policy gradient al-

Algorithm 3 Double Deep Q-Networks (DDQN)

```
Initialize $Q_w$ with random weights $w$
Initialize $\tilde{Q}_{\tilde{w}}$ with random weights $\tilde{w} = w$
Initialize replay buffer $D$ to $\emptyset$
for each episode do
  $s \leftarrow s_{init}$
  for each step of the episode do
    Choose $a$ from $s$ using policy derived from $Q_w$ (e.g $\epsilon$-greedy)
    Take action $a$, observe r and $s'$
    Store the transition $(s, a, r, s')$ in $D$
    Sample a minibatch $B$ from $D$ of size $N$
    $w \leftarrow w - \alpha \frac{1}{N} \sum_{(s,a,r,s') \sim B} \nabla_w [r + \gamma \tilde{Q}_{\tilde{w}}(s', \arg\max_a Q_w(s', a)) - Q_w(s, a)]^2$
    every K steps, set $\tilde{w} = w$
    $s \leftarrow s'$
  end for
end for
```

gorithms which aim to optimize the policy directly, without relying on value functions, by modeling the policy as a parameterized function $\pi_\theta(a \mid s)$.

2.7.1 Policy Gradient Theorem

Policy gradient methods seek to maximize performance based on the gradient of an objective function $J(\theta)$ which can be written as:

$$\nabla_\theta J(\theta) \propto \sum_s \rho(s) \sum_a Q^\pi(s, a) \nabla_\theta \pi_\theta(a \mid s), \tag{2.17}$$

where $\rho(s)$ denotes the on-policy state distribution under π (i.e., the normalized fraction of time the agent spends in state s), and the gradients are taken with respect to the policy parameter θ. The symbol $\propto$ means "proportional to". The proportionality constant is 1 in infinite-horizon tasks, and equal to the average episode length in episodic tasks. This expression is known as the **Policy Gradient Theorem**.

2.7.2 REINFORCE Algorithm

To derive the first policy-gradient learning algorithm (REINFORCE), we use stochastic gradient ascent, which relies on sample gradients that are proportional to the true gradient of the objective function. The policy gradient theorem provides such an expression proportional to the true gradient:

$$\nabla_\theta J(\theta) \propto \sum_s \rho(s) \sum_a Q^\pi(s,a) \nabla_\theta \pi_\theta(a \mid s) \tag{2.18}$$

$$= \mathbb{E}_\pi \Big[\sum_a Q^\pi(s_t,a) \nabla_\theta \pi_\theta(a|s_t) \Big] \tag{2.19}$$

$$= \mathbb{E}_\pi \Big[\sum_a \pi_\theta(a|s_t) Q^\pi(s_t,a) \frac{\nabla_\theta \pi_\theta(a|s_t)}{\pi_\theta(a|s_t)} \Big] \tag{2.20}$$

$$= \mathbb{E}_\pi \Big[G_t \nabla_\theta \log \pi_\theta(a_t|s_t) \Big] \tag{2.21}$$

Here, the last equality follows from replacing a by a sample $a_t \sim \pi_\theta(\cdot \mid s_t)$ and by applying $\mathbb{E}_\pi \Big[G_t \mid s_t, a_t \Big] = Q^\pi(s_t, a_t)$. Hence, the policy parameter update equation in the general discounted case[1] is:

$$\theta \leftarrow \theta + \alpha \gamma^t G_t \log \pi_\theta(a_t|s_t), \tag{2.22}$$

where α is the step size. This shows that REINFORCE algorithm updates the policy parameters in the direction that favors actions with the highest return. Note that REINFORCE relies on the full return from time t, incorporating all future rewards until the episode ends. This makes it an MC method, as it uses complete episodes to estimate returns.

2.7.3 REINFORCE with Baseline

REINFORCE algorithm has low bias but high variance due to the use of MC estimation. One common approach to reduce this variance is to subtract a baseline $b(s_t)$ from the return G_t. A commonly chosen baseline is the estimate of the state value function denoted by $V_w(s_t)$, where w represents a learned parameter:

$$b(s_t) = \mathbb{E}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T] = V_w(s_t). \tag{2.23}$$

[1]The discount factor was omitted in the previous derivations for simplicity and is reintroduced here in the update rule.

Using this baseline, the log-probability of an action increases proportionally to how much its return exceeds the expected return. In the following, the pseudo-code is provided for the REINFORCE algorithm with baseline:

Algorithm 4 REINFORCE with Baseline

```
Input: a differentiable policy parameterized by $\theta$: $\pi_\theta(a|s)$
Input: a differentiable value function parameterized by $w : V_w(s)$
Algorithm parameters: step sizes $\alpha_\theta > 0$, $\alpha_w > 0$
Initialize policy parameter $\theta \in \mathbb{R}^{d'}$ and state value function parameter $w \in \mathbb{R}^d$

for each episode do
    Generate an episode $s_0, a_0, r_1, \ldots, s_{T-1}, a_{T-1}, r_T$ following the policy $\pi_\theta$
    for each step t of the episode do
        $G_t \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k$
        $\delta \leftarrow G_t - V_w(s_t)$
        $w \leftarrow w + \alpha_w \delta \nabla V_w(s_t)$
        $\theta \leftarrow \theta + \alpha_\theta \gamma^t \delta \nabla \log \pi_\theta(a_t|s_t)$
    end for
end for
```

2.7.4 Actor-Critic Algorithms

Although the REINFORCE with baseline method involves learning both a policy and a state value function, it is not typically classified as an Actor-Critic (AC) method, since the value function serves only as a baseline for variance reduction rather than a critic.

AC algorithms, by contrast, simultaneously learn both the policy and the value function in a tightly coupled manner and leverage bootstrapping to estimate the value function, making the TD error central to learning. The main components of AC are:

- **Critic**: it learns the parameters w for the state value function $V_w(s)$ or the state-action value function $Q_w(s,a)$.
- **Actor**: it learns the policy parameters θ for $\pi_\theta(s|a)$, guided by the value estimates provided by the critic.

Unlike methods that must wait until the end of an episode to compute the return, AC algorithms update incrementally using the TD error. The pseudocode for an AC algorithm is as follows:

Algorithm 5 One-step Actor-Critic (episodic) algorithm

```
Input: a differentiable policy parameterized by $\theta$: $\pi_\theta(a|s)$
Input: a differentiable value function parameterized by $w : V_w(s)$
Algorithm parameters: step sizes $\alpha_\theta > 0$, $\alpha_w > 0$
Initialize policy parameter $\theta \in \mathbb{R}^{d'}$ and state value function parameter $w \in \mathbb{R}^d$

for each episode do
    Initialize the start state s
    $I \leftarrow 1$
    for each step of the episode do
        Sample action $a$ from $\pi_\theta(.|s)$
        Take action $a$, observe reward $r$ and next state $s'$
        $\delta \leftarrow r + \gamma V_w(s') - V_w(s)$
        $w \leftarrow w + \alpha_w \delta \nabla V_w(s)$
        $\theta \leftarrow \theta + \alpha_\theta I \delta \nabla \log \pi_\theta(a|s)$
        $I \leftarrow \gamma I$
        $s \leftarrow s'$
    end for
end for
```

Asynchronous Advantage Actor-Critic (A3C) [40] is a variant of Actor-Critic methods in which multiple agents (actors) interact with separate instances of the environment and are trained in parallel, typically across different CPU cores. Each actor periodically synchronizes with the global network parameters. After interacting with its environment, an actor accumulates gradients using its local parameters and sends them to the global network, which updates its weights slightly in the direction of each training thread's gradients. The actor's parameters are then reset to match the updated global parameters. This process is asynchronous because updates occur at different times for different actors.

More recently, researchers introduced Advantage Actor-Critic (A2C), a synchronous and deterministic version of A3C. The key difference is the inclusion of a coordinator that synchronizes all actors and updates the global parameters only after all actors have completed their interactions. This removes the inconsistency caused by asynchronous updates. Empirical results have shown that A2C converges faster and achieves better performance than A3C [41].

Proximal Policy Optimization (PPO) [42] is a widely-used AC method designed to achieve reliable and stable policy updates. Unlike A3C or A2C, PPO focuses on improving the training stability of policy gradients through a surrogate objective function that limits large policy updates. The core idea is to ensure that the new policy does not diverge significantly from the old one by clipping the probability ratio between the new and old policies. The main clipped surrogate objective objective is defined as follows [42]:

$$L_t^{\text{CLIP}}(\theta) = \mathbb{E}_t\left[\min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t,\ \text{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon_{clip}, 1+\epsilon_{clip}\right)\hat{A}_t\right)\right],$$

where $\hat{A}_t$ is an estimator of the advantage function and ϵ_{clip} is a small positive hyperparameter (e.g., $\epsilon_{clip} = 0.2$). The clipping prevents the probability ratio from deviating too far from 1, ensuring conservative updates.

In practice, PPO combines this clipped objective with a value function loss and an entropy bonus, forming the total objective function:

$$L_t^{\text{CLIP+VF+Entropy}}(\theta) = \mathbb{E}_t\left[L_t^{\text{CLIP}}(\theta) - c_1 L_t^{\text{VF}}(\theta) + c_2 H(\pi_\theta(.|s_t))\right],$$

where:

- $L_t^{\text{VF}}(\theta) = \left(V_\theta(s_t) - V^{\text{target}}(s_t)\right)^2$ is the squared error loss between $V_\theta(s_t)$ and the target value function $V^{\text{target}}(s_t)$.
- $H(\pi_\theta(.|s_t))$ is the entropy of the policy, encouraging exploration.
- c_1, c_2 are coefficients for the value loss and entropy bonus, respectively.

Note that we used the notation $V_\theta(s)$ for simplicity, since PPO typically uses a shared parameter vector θ for both the policy and value function.
For the advantage estimate $\hat{A}_t$, it can be computed using Generalized Advantage Estimation (GAE) [42]. Assuming the policy is run for a fix number of T

time steps, the advantage function can be written as:

$$\hat{A}_t = \delta_t + (\gamma\lambda_{gae})\delta_{t+1} + \cdots + (\gamma\lambda_{gae})^{T-t+1}\delta_{T-1},$$

where $\delta_t = r_t + \gamma V_\theta(s_{t+1}) - V_\theta(s_t)$, and $\lambda_{gae} \in [0,1]$ controls the trade-off between bias and variance.

The PPO algorithm proceeds as follows [42]:

Algorithm 6 PPO, Actor-Critic Style

```
for iteration = 1, 2, ... do
    for actor = 1, ..., N do
        Run policy $\pi_{\theta_{\text{old}}}$ in the environment for $T$ time steps
        Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
    end for
    Optimize the objective $L^{\text{CLIP+VF+Entropy}}$ w.r.t. $\theta$, using minibatch
    stochastic gradient descent (SGD) for $K$ epochs
    Update $\theta_{\text{old}} \leftarrow \theta$
end for
```

2.8 Summary and Conclusion

This chapter offered an overview of foundational RL concepts, including MDPs and POMDPs, value functions and Bellman optimality, model-based vs. model-free approaches, function approximation from linear to deep RL, value- and policy-based methods, and key algorithm pseudocode central to RL research. While much of the material draws from established sources such as [4], it establishes the essential language and tools needed to understand and situate contemporary RL research. In particular, the frameworks and algorithms presented in this chapter directly frame the context for the studies discussed later in this thesis, including the impact of intrinsic rewards on exploration in deep RL (Chapter 4) and the analysis of sample complexity in reward-free, kernel-based RL (Chapter 5). By providing these foundational principles, this chapter ensures that the subsequent work can be interpreted both rigorously and intuitively within the broader RL framework.

Chapter 3

Background on Bandits

Another fundamental framework for sequential decision-making under uncertainty is the family of bandit problems. In contrast to RL problems, which involve multiple states and transitions among them, bandit problems focus on a single state with multiple actions (referred to as arms), each producing stochastic rewards. They also serve as a key theoretical foundation for RL, often representing the first step toward understanding more complex RL problems. This simplified setting allows researchers to isolate and study the *exploration-exploitation tradeoff*, a central challenge in machine learning and decision theory. In addition to serving as a foundational framework, bandits have found broad applications in a variety of domains, such as clinical trials, dynamic spectrum allocation in wireless networks, online advertising, web search optimization, and social networking analysis [43]. This wide range of applications underscores the importance of bandits as a critical area of study.

In this chapter, we cover several main classes of bandit models. We begin with *multi-armed bandits (MAB)*, characterized by a finite set of actions, each providing independent stochastic rewards. We then discuss *linear bandits*, where the expected reward of each action is modeled as a linear function of the action's feature vector. Next, we cover *Gaussian Process (GP) bandits*, which model the unknown reward function using a GP and form the core of many *Bayesian Optimization (BO)* algorithms. Finally, we present *dueling bandits*, a variant where feedback is provided through pairwise comparisons instead of absolute rewards.

Together, these classes of bandits provide the foundation for much of modern bandit theory and its applications. They also serve as essential building blocks for our theoretical work in subsequent chapters.

3.1 Multi-Armed Bandits

In the classic MAB setup, a player sequentially selects one arm $a_t \in \mathcal{A}$ at each time step $t \in \{1, \ldots, T\}$, where $\mathcal{A}$ is a finite set of K arms. Each arm $a \in \mathcal{A}$ yields a random reward $r_t \in \mathbb{R}$, drawn independently from an unknown distribution with mean $\mu_a = \mathbb{E}[r_t \mid a_t = a]$. Rewards from different arms are assumed to be independent. The goal is to design an arm selection policy that maximizes the total expected reward over a finite horizon T.

A major challenge in MAB problems is managing the exploration–exploitation tradeoff: *exploration* involves selecting under-sampled arms to better estimate their reward distributions, while *exploitation* focuses on choosing the arm believed to offer the highest expected reward based on historical data.

The performance of a policy is commonly evaluated through *cumulative regret*, defined as the expected loss in cumulative reward compared to an omniscient player who always selects the arm $a^\star$, which is the arm with the highest expected reward:

$$R(T) = T\mu_{a^\star} - \mathbb{E}\left[\sum_{t=1}^{T} r_t\right],$$

where $a^\star$ is given by

$$a^\star = \arg\max_{a \in \mathcal{A}} \mu_a \,.$$

Achieving sublinear regret in T guarantees that the policy's reward converges to the optimal reward as the number of plays grows.

In [44], authors showed that the minimum regret has a logarithmic order in T. Under the assumption that the reward distribution family is known, [44] have introduced policies achieving this logarithmic order for several reward distributions, including Bernoulli, Poisson, Gaussian, and Laplace. Later, [45] proposed the *Upper Confidence Bound (UCB)* algorithm known for its simplicity and strong theoretical guarantees. At each time step t, the UCB algorithm selects the arm

$$a_t = \arg\max_{a \in \mathcal{A}} \left[\hat{\mu}_a(t) + \sqrt{\frac{2 \log t}{N_a(t)}} \right],$$

where $\hat{\mu}_a(t)$ denotes the empirical mean reward of arm a, and $N_a(t)$ is the number of times arm a has been selected up to time t. The confidence term encourages exploration by favoring arms with greater uncertainty. UCB ensures each arm is sampled sufficiently—on the order of $\log t$—to achieve logarithmic regret. The original UCB algorithm assumes reward distributions with bounded support [45]. This approach has since been extended to light-tailed distributions [46] and a later variant of UCB was developed to achieve optimal logarithmic regret for heavy-tailed rewards with finite p-th moments [47]. Moreover, a different approach, termed the deterministic sequencing of exploration and exploitation (DSEE), was proposed by [48] to achieve logarithmic regret for both light-tailed and heavy-tailed cases.

Another important class of algorithms for the MAB problem is *Thompson Sampling (TS)*, also known as *posterior sampling*. First introduced by W. R. Thompson in 1933 [49], it is a Bayesian approach that maintains a posterior distribution over the expected reward of each arm. At each time step, the algorithm samples a value from each posterior and selects the arm with the largest sampled value. This posterior reflects all rewards observed so far and is updated after each time step, becoming the prior for the next time step. This means that actions are chosen with a probability proportional to their likelihood of being optimal under the current posterior. TS has been analyzed in the literature [50, 51], with theoretical guarantees demonstrating its effectiveness. In particular, the analysis by [50] showed that TS achieves logarithmic expected cumulative regret in the stochastic MAB setting. Furthermore, for the specific case of Bernoulli bandits, TS was shown in [51] to attain the asymptotic lower bound on regret established by [44].

While the standard stochastic MAB problem is well understood in the case of finite set of arms, it becomes intractable as the number of arms grows large—and essentially unmanageable for infinite arm sets.

3.2 Linear Bandits

Stochastic linear bandits extend the classical MAB setting by assuming that the reward is a linear function of the chosen action. Specifically, the action set is $\mathcal{A} \subset \mathbb{R}^d$, and at each time step t, the learner selects an action $a_t \in \mathcal{A}$ and receives a reward

$$r_t = \langle a_t, \theta \rangle + \varepsilon_t,$$

where $\theta \in \mathbb{R}^d$ is an unknown parameter vector and ε_t is random zero-mean noise term.

The linear bandit problem was first introduced by [52] under the name *linear reinforcement learning*. In this formulation, the set of available actions changes from time step to time step but has a fixed finite cardinality. Their work proposed two algorithms: LinRel, a simpler algorithm without formal regret analysis, and SupLinRel, which achieves a regret upper bound of $\tilde{O}(\log^{3/2} K\sqrt{dT})$[1], where d is the dimensionality of the unknown parameter and K is the number of actions. This approach was later studied by [53] and [54] in the context of web advertisement.

A more general setting, where the action set remains fixed over time but may be infinite (subject to being a bounded subset of a finite-dimensional vector space), was studied by [55, 56, 57, 58]. For instance, [55] introduced the Confidence Ball algorithm and proved a regret bound of $\tilde{O}(d\sqrt{T})$, also showing this bound to be tight by proving a lower bound of the same order. The apparent difference from the earlier upper bound $\tilde{O}(\sqrt{dT})$ by [52] arises from variations in the underlying settings and constraints.

A unifying concept across all these formulations is the *Optimism in the Face of Uncertainty (OFU)* principle. The learner maintains a confidence set

[1]Throughout this thesis, we use the $\mathcal{O}$ and $\tilde{\mathcal{O}}$ notations to hide constants and logarithmic terms, respectively, for simplicity of presentation.

for the unknown parameter vector based on past observations and, at each time step, selects the action that would yield the highest reward under the optimistic plausible parameter. Building on the work of [55], [58] introduced smaller confidence set constructions by addressing the dependence between arms through a novel tail inequality for vector-valued martingales and techniques from the theory of *self-normalized processes* [59, 60]. Their algorithm, called Optimism in the face of Uncertainty Linear bandits (OFUL), produced tighter confidence sets that hold uniformly over time, resulting in improved regret bounds and enhanced empirical performance.

Another family of approaches applies TS to the linear bandit setting. A study by [61] considered the contextual linear bandit problem, where each arm is associated with a d-dimensional feature vector (context). The authors proposed an algorithm which maintains a Gaussian posterior over the unknown parameter vector, updating it with observed rewards and contexts. At each time step, it selects the arm with the highest expected reward under a sample from this posterior. A regret bound of $\mathcal{O}(d^{3/2}\sqrt{T})$ was established for this algorithm. In related work, [62] provided an alternative proof of the regret bound for TS in the stochastic linear bandit setting. They achieved the same regret bound, while offering new insights into the behavior and underlying mechanisms of TS.

3.3 Gaussian Process Bandits / Bayesian Optimization

An alternative and powerful approach to modeling bandit problems with a continuum set of arms is based on the framework of GPs [63]. GP bandit optimization has developed under two distinct approaches with different philosophies and terminologies (see, e.g., [64]). On the one hand, the Bayesian approach models the unknown reward function as a GP, treats the problem probabilistically, and produces a posterior distribution with predictive uncertainty. This viewpoint is commonly referred to as BO [65]. On the other hand, the frequentist approach assumes that the unknown reward function is deterministic and lies in a Reproducing Kernel Hilbert Space (RKHS) associated with a known positive definite kernel. It leverages kernel ridge regression, where predictions are obtained by minimizing a regularized empirical loss. Although historically distinct, these approaches are intimately connected and lead to the same regression solution.

Theoretically, GP bandits can also be seen as a generalization of the linear bandit setting, extending it to infinite-dimensional feature spaces and allowing nonlinear functions to be treated as linear in a high-dimensional RKHS [66]. In what follows, we first review background on GPs, RKHS, and kernel ridge regression. We then discuss common kernels and complexity measures such as information gain, summarize existing confidence intervals, and finally present algorithms for GP bandits along with their regret bounds.

3.3.1 Gaussian Processes (Bayesian View)

GPs provide a flexible nonparametric framework for modeling functions with uncertainty. A GP $\{f(x)\}_{x\in\mathcal{X}}$ is a collection of random variables, any finite subset of which is jointly Gaussian. A GP is fully specified by a mean function $m:\mathcal{X}\to\mathbb{R}$ and a covariance (kernel) function $k:\mathcal{X}\times\mathcal{X}\to\mathbb{R}$, such that:

$$\mathbb{E}[f(x)] = m(x), \quad \mathbb{E}[(f(x_i)-m(x_i))(f(x_j)-m(x_j))] = k(x_i,x_j),$$

where $x_i, x_j \in \mathcal{X}$. In practice, the mean is often assumed to be zero without loss of generality.

The Bayesian view of GP-bandit optimization generally considers the problem of optimizing a fixed and unknown function $f:\mathcal{X}\to\mathbb{R}$, where $\mathcal{X}\subset\mathbb{R}^d$ is a compact domain. The function f is assumed to be a sample from a GP prior. At each time step, the algorithm selects a point $x_t\in\mathcal{X}$ and observes a noisy evaluation $y_t = f(x_t)+\varepsilon_t$ where ε_t is i.i.d. zero-mean Gaussian noise. The goal is to rapidly identify the maximum of f through the sequence of noisy observations. To guide the search for the maximizer of f, the algorithm must continually update its belief about the function as data accumulate. Conditioning on the history of observations yields a posterior distribution for the function f, characterized by a mean and variance. This procedure is referred to as GP regression. These posterior quantities (mean and variance) coincide with those obtained from kernel ridge regression; the explicit forms are presented in Section 3.3.3.

3.3.2 Reproducing Kernel Hilbert Spaces (frequentist view)

Let k be a positive definite kernel, and let $\mathcal{H}_k$ be the RKHS associated with k. The RKHS consists of functions $f : \mathcal{X} \to \mathbb{R}$ with finite norm $\|f\|_{\mathcal{H}_k}$, and satisfies the reproducing property:

$$f(x) = \langle f, k(\cdot, x) \rangle_{\mathcal{H}_k}, \quad \forall f \in \mathcal{H}_k.$$

Mercer's theorem guarantees that k admits a spectral decomposition:

$$k(x, x') = \sum_{m=1}^{\infty} \gamma_m \varphi_m(x) \varphi_m(x'),$$

with eigenvalues $\gamma_m > 0$ and orthonormal eigenfunctions $\{\varphi_m\}$. Functions $f \in \mathcal{H}_k$ can be written as:

$$f = \sum_{m=1}^{\infty} w_m \sqrt{\gamma_m} \varphi_m, \quad \text{with } \|f\|_{\mathcal{H}_k}^2 = \sum_{m=1}^{\infty} w_m^2.$$

The frequentist view of GP bandit optimization assumes that the objective function f has a bounded norm in the RKHS: $\|f\|_{\mathcal{H}_k} \leq B$, and that the noise terms ε_t are sub-Gaussian random variables. This approach leverages kernel ridge regression, which combines least-squares fitting with RKHS regularization. We introduce kernel ridge regression next and then highlight its connection to GP regression.

3.3.3 Kernel Ridge Regression

Given t noisy observations $\{(x_i, y_i)\}_{i=1}^t$, where $y_i = f(x_i) + \varepsilon_i$, the kernel ridge regression estimator is defined as the solution to the regularized least-squares problem:

$$\hat{f}_t = \arg\min_{f \in \mathcal{H}_k} \left\{ \sum_{i=1}^t (f(x_i) - y_i)^2 + \tau^2 \|f\|_{\mathcal{H}_k}^2 \right\},$$

where $\tau^2 > 0$ is a regularization parameter. The resulting closed-form predictor and uncertainty estimate are:

$$\begin{aligned} \hat{f}_t(x) &= k_t^\top(x) \left(K_t + \tau^2 I\right)^{-1} \mathbf{y}_t, \\ \sigma_t^2(x) &= k(x,x) - k_t^\top(x) \left(K_t + \tau^2 I\right)^{-1} k_t(x), \end{aligned} \tag{3.1}$$

where $k_t(x) = [k(x,x_1), \ldots, k(x,x_t)]^\top$, $K_t = [k(x_i,x_j)]_{i,j=1}^t$ is the kernel matrix, $\mathbf{y}_t = [y_1, \ldots, y_t]^\top$, and I is the identity matrix.

Equivalence with GP Regression. These expressions for $\hat{f}_t(x)$ and $\sigma_t^2(x)$ coincide exactly with the posterior mean and variance of a zero-mean GP with the same covariance function and Gaussian noise of variance τ^2 [63]. In this thesis, we adopt the RKHS-based frequentist view, framing estimators and regret bounds in terms of kernel ridge regression. Nonetheless, the GP interpretation remains valuable for intuition, particularly in algorithm design.

3.3.4 Common Kernels and their Properties

In practice, the Matérn and Squared Exponential (SE) kernels are among the most widely used choices in BO. These kernels are defined as follows:

$$k_{\text{Matérn}}(x,x') = \frac{1}{\mathcal{G}(\nu)2^{\nu-1}} \left(\frac{\sqrt{2\nu}\rho}{l}\right)^\nu B_\nu\left(\frac{\sqrt{2\nu}\rho}{l}\right), \tag{3.2}$$

$$k_{\text{SE}}(x,x') = \exp\left(-\frac{\rho^2}{2l^2}\right), \tag{3.3}$$

where $l > 0$ is the lengthscale, $\rho = \|x - x'\|_2$ is the Euclidean distance between x and x', $\nu > 0$ is the smoothness parameter, $\mathcal{G}$ denotes the Gamma function, and B_ν is the modified Bessel function of the second kind. The SE kernel can be interpreted as a special case of the Matérn kernel as $\nu \to \infty$.

3.3.5 Maximum Information Gain and Eigendecay

We introduce a kernel specific complexity term referred to as maximum information gain that helps characterize the regret of algorithms in bandits and kernel-based RL. It is defined as follows[2] [67, 68]:

$$\Gamma(t) = \max_{x_1,\ldots,x_t} \frac{1}{2} \log\det(I + \tau^{-2} K_t) \tag{3.4}$$

where K_t is the kernel matrix, and τ^2 is the regularization parameter (as defined in Section 3.3.3). The maximum information gain depends on the eigendecay defined as follows.

Definition 1. *A kernel k is said to have a polynomial (resp. exponential) eigendecay if $\gamma_m = \mathcal{O}(m^{-p})$ (resp. $\gamma_m = \mathcal{O}(c^m)$), for some $p > 1$ ($c < 1$), where γ_m are the Mercer eigenvalues in decreasing order.*

In particular, for smooth kernels with exponentially decaying eigenvalues—such as the SE kernel—the maximum information gain grows polylogarithmically in T [68]. In contrast, for broader classes of kernels with polynomial eigendecay that are of both practical and theoretical importance, including the Matérn family [69], the information gain grows polynomially in T. In such cases, the resulting regret bounds may no longer be sublinear (i.e., become vacuous), and therefore do not guarantee that the algorithm's performance improves over time.

3.3.6 Confidence Intervals

GP modeling enables the construction of confidence intervals for complex functions defined over continuous domains. In particular, we utilize the GP's predictive mean $\hat{f}_t(x)$ and uncertainty estimate $\sigma_t(x)$ to construct confidence

[2]In later chapters, we use slight variations of the maximum information gain notation to reflect local context. In **Chapter 5**, we denote the number of observations by n instead of t. In **Chapter 6**, we replace τ^2 by λ. This is purely a notational change; the definitions are equivalent.

bounds of the form:

$$|f(x) - \hat{f}_t(x)| \leq \beta_t(\delta)\,\sigma_t(x),$$

where $\beta_t(\delta)$ is a confidence width multiplier corresponding to confidence level of $1-\delta$. Confidence intervals formally quantify our uncertainty about the unknown function at any input, with high probability guarantees. The key goal of the theoretical analysis is to derive expressions for $\beta_t(\delta)$ that are as tight as possible while still ensuring the bound holds with high probability.

Under the assumption that the observation noise is ς-sub-Gaussian and the function $f \in \mathcal{H}_k$, Theorem 1 in [70] provides a high-probability bound in the *non-adaptive* setting—where the observation points $\{x_j\}_{j=1}^t$ are independent of the observation noise. Specifically, for any fixed $x \in \mathcal{X}$, with probability at least $1-\delta$, the confidence width multiplier is:

$$\beta_t(\delta) = \|f\|_{\mathcal{H}_k} + \frac{\varsigma}{\tau}\sqrt{2\log\left(\frac{1}{\delta}\right)}. \tag{3.5}$$

In the *adaptive* setting—where the observation points $\{x_j\}_{j=1}^t$ are adaptively selected based on prior observation values—the standard concentration inequalities cannot be applied directly because the data exhibit temporal dependence. This challenge was addressed in the *linear case* where $f(x) = \theta^\top x$, by the work of [58], who derived high-probability confidence intervals using self-normalized concentration inequalities for vector-valued martingales. Specifically, for all $x \in \mathcal{X}$, with probability at least $1-\delta$, the confidence width multiplier is given by:

$$\beta_t(\delta) = \|f\|_{\mathcal{H}_k} + \frac{\varsigma}{\tau}\sqrt{d\log\left(\frac{1+t\bar{x}^2/\tau^2}{\delta}\right)}, \tag{3.6}$$

and $\bar{x} = \max_{x\in\mathcal{X}}\|x\|_2$. Their analysis is based on bounding the martingale sequence $S_t = \sum_{i=1}^t \varepsilon_i x_i$, leading to the following confidence *ellipsoid* for the unknown parameter θ:

$$\|\theta - \hat{\theta}_t\|_{V_t} \leq \tau\beta_t(\delta),$$

where $V_t = \tau^2 I + \sum_{i=1}^{t} x_i x_i^\top$. This confidence ellipsoid for θ can then be represented in terms of the confidence interval for f [71].

Building on this framework, an extension to the RKHS (kernel) setting was provided by [66] through an analogous self-normalized bound for vector-valued martingales. Under the assumptions on sub-Gaussian noise and bounded RKHS norm, it is shown that, with probability at least $1-\delta$, the confidence interval bound holds uniformly for all $x \in \mathcal{X}$ [72, 32]:

$$\beta_t(\delta) = \|f\|_{\mathcal{H}_k} + \frac{\varsigma}{\tau}\sqrt{2\log\left(\frac{1}{\delta}\right) + \Gamma(t)}, \tag{3.7}$$

where $\Gamma(t)$ defined in (3.4), denotes the maximum information gain at time t.

These results highlight a key difference between non-adaptive and adaptive sampling. In the non-adaptive case, $\beta_t(\delta)$ scales only with $\sqrt{\log(1/\delta)}$ (see Equation (3.5)), giving tighter confidence intervals since observation points are independent of the noise. In adaptive sampling, each x_t depends on past data, creating statistical dependencies that standard concentration results cannot handle. To compensate, $\beta_t(\delta)$ must also scale with the maximum information gain $\Gamma(t)$, which measures how much information t samples can reveal about the function under the GP prior. Consequently, adaptive confidence intervals are inflated by an additional $\sqrt{\Gamma(t)}$ factor (see Equation (3.7)). Intuitively, larger $\Gamma(t)$ means greater potential dependence among observations, requiring more conservative bounds to ensure the true function remains within them with high probability.

These high-probability confidence intervals are not merely theoretical constructs; they form the core mechanism behind many BO and bandit algorithms. They quantify how much we can trust GP predictions and how conservative we must be when selecting points adaptively. The guarantee that the true function f lies within the bounds allows these algorithms to strategically balance exploration and exploitation, with formal regret guarantees. Tighter confidence intervals directly translate to faster convergence and better performance of these algorithms.

3.3.7 Algorithms for GP Bandits and Regret Bounds

Among the most widely studied algorithms for GP bandits is GP-UCB [67], which maintains a kernel ridge regression estimate of the unknown function together with a confidence ellipsoid around this estimate. At each time step, it optimistically selects the point that maximizes the predicted mean plus a scaled standard deviation, thereby balancing exploration and exploitation. Alternative acquisition strategies, such as Expected Improvement (EI) [73, 74, 75], Probability of Improvement (PI) [76], and Thompson Sampling (GP-TS) [66], follow different selection rules but achieve comparable cumulative regret guarantees of order $\mathcal{O}(\Gamma(T)\sqrt{T})$, where $\Gamma(T)$ is the maximum information gain of the kernel [67].

This bound, however, is often loose [77]. For common kernels such as Matérn, $\Gamma(T)$ can grow polynomially with T, leading to regret that may fail to be sublinear and is known to be a factor $\sqrt{\Gamma(T)}$ worse than the minimax rate [65].

To mitigate this limitation, several algorithmic refinements have been proposed. For instance, Local Polynomial GP-UCB (LP-GP-UCB) [78] augments GP models with local polynomial estimators, improving performance in certain regimes while retaining the same regret bound $O(\Gamma(T)\sqrt{T})$. The Partioned Improved GP-UCB (π-GP-UCB) algorithm [79] instead partitions the search space into hypercubes and fits an independent GP to each, achieving a sublinear regret across all parameters of the Matérn kernel family.

More substantial progress was achieved by SupKernelUCB [80], which achieves $\tilde{\mathcal{O}}(\sqrt{\Gamma(T)T})$ regret on discrete domains, and extends to continuous domains under mild Lipschitz-style assumptions. When combined with tighter bounds on $\Gamma(T)$ [68], this nearly matches the lower bounds for SE and Matérn kernels. However, SupKernelUCB is often regarded as impractical [81]. Subsequent works proposed more practical algorithms that achieve the same $\tilde{\mathcal{O}}(\sqrt{\Gamma(T)T})$ scaling: Robust Inverse Propensity Score (RIPS) [82] achieves the bound while requiring only $O(\log T)$ batches and being robust to model misspecification. GP-Thresholded Domain Shrinking (GP-ThreDS) [83] adopts a tree-based domain shrinking strategy that performs well empirically while meeting the theoretical guarantee. Batched Pure Exploration (BPE) [84] employs a structured, multiround batching scheme that leverages uncertainty information across rounds to balance exploration and exploitation.

Beyond cumulative regret, another performance metric studied in the GP bandit literature is *simple regret*, which evaluates the quality of the final recommendation rather than the entire sequence of actions. Formally, it measures the gap between the optimal reward and the reward obtained at the point selected as best after T iterations. Although [67] did not analyze simple regret explicitly, their cumulative regret bound of $\mathcal{O}(\Gamma(T)\sqrt{T})$ for GP-UCB implies a simple regret bound of $\mathcal{O}(\Gamma(T)/\sqrt{T})$ [65]. This rate of simple regret was later established directly under the RKHS setting with noisy observations in several works [65, 78, 85]. A sharper analysis was provided by [70], who introduced the Maximum Variance Reduction (MVR) algorithm and proved a $\sqrt{\Gamma(T)}$-factor improvement, bringing the achievable simple regret closer to the known information-theoretic lower bounds. Importantly, these results are often framed in terms of sample complexity, i.e., the number of evaluations required to guarantee that the simple regret is below a target threshold with high probability.

3.4 Dueling Bandits

The assumption of a numerical reward signal is a potential limitation of the standard MAB setting. In many real-world applications, it is hard—or even impossible—to quantify the quality of an option numerically. For instance, consider crowdsourcing services like Amazon Mechanical Turk, where annotators are asked to make pairwise comparisons between alternatives. The goal is to approximate an underlying preference ordering based on these (possibly noisy and inconsistent) comparisons [86]. Preferential feedback also arises in various online learning tasks such as information retrieval [87], recommendation systems [88], and skill rating/player ranking [89].

To address such scenarios, the *dueling bandits* framework extends the MAB framework to handle preference-based feedback [87]. In this setting, the learner selects two arms at each time step and receives binary feedback indicating which arm is preferred, in contrast to the traditional MAB setting where a single arm is chosen and a numerical reward is observed. We refer the reader to [90] for a comprehensive survey. Formally, at each time step $t = 1, \dots, T$, the interaction in the dueling bandits setting proceeds as follows:

- The algorithm selects a pair of actions/arms a_i and a_j from the available set of actions.

- The environment returns preference-based feedback: action a_i is preferred over a_j with probability $\mathbb{P}(a_i \succ a_j)$, and vice versa with probability $\mathbb{P}(a_j \succ a_i) = 1 - \mathbb{P}(a_i \succ a_j)$.

The central goal in the dueling bandits framework is to design algorithms that identify the "optimal" arm as efficiently as possible, typically measured in terms of *cumulative regret*. Unlike standard MAB, defining optimality in dueling bandits is more nuanced due to the preference-based feedback. Several notions of optimality have been proposed in the literature, including the *Copeland Winner* [91], *Borda Winner* [92], and *von Neumann Winner* [93], each offering a different interpretation of the optimal arm. In utility-based dueling bandits, preferences are assumed to stem from latent utility values, and the optimal arm is defined as the one maximizing this utility. Consequently, regret is defined relative to the chosen notion of optimality. In general, cumulative regret measures how much the learner's selected arms underperform compared to the optimal arm over time. Depending on the model, regret may be expressed in terms of the utility gap, Borda score, Copeland score, or expected outcomes against the von Neumann winner.

A wide range of algorithms has been developed to address these problems under various assumptions. The initial works on dueling bandits focused on scenarios with a finite number K of arms. Here, the problem instance can be fully described by a $K \times K$ preference probability matrix, where each entry $p_{i,j} = \mathbb{P}(a_i \succ a_j)$ denotes the probability that arm i is preferred over arm j. Many algorithms attempted to learn this pairwise preference matrix by leveraging noisy sorting or tournament procedures, or by generalizing classical MAB algorithms to the preference-based setting. Examples include Interleaved Filter [94], Beat the Mean (BTM) [95], Relative Confidence Sampling [96], Relative Upper Confidence Bound (RUCB) [97], Doubler, MultiSBM, Sparring [98], and mergeRUCB [99]. These methods typically impose various structural assumptions on the problem, such as strong stochastic transitivity (SST) [94, 100], stochastic triangle inequality (STI) [95], weak or relaxed stochastic transitivity [101, 95], or the presence of a Condorcet winner (an arm preferred over every other arm) [97, 99, 96]. These algorithms usually guarantee regret bounds that grow logarithmically in the number of time steps but at least linearly with respect to the number of arms, limiting their scalability to large or infinite arm sets.

To address settings with infinitely many arms, utility-based dueling bandits have gained popularity. Instead of a preference matrix, preferences are modeled via an unknown latent utility function. Preference feedback is interpreted as arising from the difference in utilities of the chosen arms, passed through a link function. A common choice is the logistic function, as in the Bradley-Terry-Luce (BTL) model [102], where the probability of preferring one arm over another is given by the logistic function of the utility difference.

The simplest structured variant is the linear contextual dueling bandit, studied in [103, 104, 105, 106, 107], which allows a large action set but assumes a linear utility function. Here, the utility of an arm a is modeled as $f(a) = \theta^\top \phi(a)$ for some unknown parameter vector $\theta \in \mathbb{R}^d$ and a known feature map $\phi(\cdot) \in \mathbb{R}^d$. Preference-based bandit optimization with linear utilities is well understood and has even been extended to RL settings involving preference feedback on trajectories [108, 109, 110, 111, 112]. However, such linear models have limited applicability in practice, as they cannot capture the complex nonlinear utility functions typical of many real-world problems.

Alternatively, RKHS offer a rich class of models for representing the utility function. This has motivated several extensions of the dueling bandit problem to kernel-based settings. Recent works have investigated the convergence of kernelized algorithms for preference-based bandits. For instance, [113, 114, 115] focus on the Borda score, which quantifies the probability that a given action is preferred over a uniformly sampled action from the domain. However, these approaches typically rely on strong assumptions about the Borda function, effectively reducing the problem to a conventional BO setting. Moreover, these works model the problem using a regression likelihood, which assumes that both the latent utility function and the probability of preference lie in an RKHS. This regression-based modeling approach is somewhat misaligned with the inherently classification-based nature of preference learning. While the model is valid, it often does not lead to sample efficient algorithms.

In contrast, other methods [116, 117] adopt a kernelized logistic negative log-likelihood loss to infer the utility function. These works provide confidence sets for the minimizer of the logistic loss and establish tighter regret guarantees, offering more principled and efficient approaches.

Finally, a recent extension [118] introduces neural dueling bandits, where wide neural networks are used for preference prediction instead of kernels. Although the theoretical analysis relies on connections to the Neural Tangent kernel (NTK), this approach demonstrates an alternative modeling strategy. Importantly, the guarantees and assumptions remain largely analogous to those in kernel-based methods.

3.5 Summary and Conclusion

This chapter provided a structured overview of bandit models, tracing their development from classical MAB to more advanced frameworks. We started with finite MAB, progressed to linear bandits and then to GP bandits, which allow more general nonlinear rewards. Finally, we covered dueling bandits, where scalar feedback is replaced by binary comparisons.

This progression from simple to complex bandit models lays the foundation for the techniques developed in this thesis. We introduced core algorithmic families and key principles—such as the exploration–exploitation trade-off, regret, OFU, GPs, kernel ridge regression, confidence intervals, and complexity measures—that underpin the theoretical analysis of these models. These concepts are essential for the subsequent chapters: near-optimal sample complexity in reward-free, kernel-based RL (Chapter 5) and Bayesian Optimization from Human Feedback (Chapter 6), with dueling bandits providing crucial background for the latter. Overall, this chapter equips the reader with the theoretical framework and technical tools that connect classical bandit theory to the advanced methods explored in this thesis.

Chapter 4

The Impact of Intrinsic Rewards on Exploration in RL

Following the background on RL and bandit problems, this chapter turns to the exploration problem in RL, focusing on an empirical proof-of-concept study. A persistent challenge in deep RL is exploration in sparse-reward environments, where rewards are provided to the agent only rarely—for example, when accomplishing a task. As a consequence, the agent often fails to discover rewarding behaviors. To address this challenge, researchers have proposed various types of intrinsic rewards that encourage diversity in exploration. Such diversity can be imposed at different levels, favoring the agent to explore different states, policies, or behaviors (State, Policy, and Skill level diversity, respectively). Yet, the actual impact of these different types of diversity on exploration behavior remains unclear. In this chapter, we aim to fill this gap by studying how intrinsic rewards operating at different levels of diversity affect the exploration patterns of RL agents. We select four intrinsic rewards (State Count, Intrinsic Curiosity Module (ICM), Maximum Entropy, and Diversity is All You Need (DIAYN)), each favoring different levels of diversity. We conduct an empirical study on MiniGrid environments to compare their impact on exploration considering various metrics related to the agent's exploration. This study sheds light on how diversity influences exploration and offers practical implications for selecting and applying intrinsic rewards in environments with varying levels of exploration difficulty.

4.1 Introduction

The sparsity of rewards is a major hurdle for RL algorithms [25, 119]. With infrequent feedback, the probability of the agent randomly discovering a rewarding sequence of actions becomes low. Therefore, a large number of samples is needed to explore and stumble into a successful sequence of actions leading to the desired outcome [120]. This is known as the hard exploration problem [25]. Classical exploration strategies, e.g., epsilon-greedy and Boltzmann distribution [121] fail to explore the environment efficiently enough to find the optimal solution when the feedback is sparse [122]. Among the possible solutions to address this limitation [25, 123, 124], intrinsic rewards [125, 126] have been proposed. Intrinsic rewards are signals that encourage the agent to explore novel experiences, with the aim of enhancing learning efficiency in environments with sparse external rewards [125, 126]. They are a part of the larger notion of intrinsic motivation defined by [127] as the tendency to "*seek out novelty and challenges, to extend and exercise one's capacity, to explore, and to learn*". Intrinsic rewards are often categorized in the literature into *knowledge-based* and *competence-based* [128, 129, 124, 130]. The first category encourages the agent to gain new knowledge about the environment. It compares the agent's experiences to its existing knowledge, and rewards the agent for encountering unexpected situations. This includes methods that reward novelty in states or state transitions [131, 132, 133, 134], the prediction error [135] or the information gain [136]. The second category, also called "*skill learning*" in [124, 137], rewards the agent for learning a diverse repertoire of skills in an unsupervised way. It mainly includes goal-conditioned RL approaches, which generate and achieve their own goals to explore the environment [138, 139, 140]. In [129], a detailed survey on goal-conditioned RL is presented, highlighting the different types of goal representations and goal-sampling strategies.

This categorization uncovers a potential link between diversity and exploration, where intrinsic rewards promote diverse agent behaviors to efficiently explore the environment. While diversity is acknowledged as crucial in RL, it has mainly been explored in relation to robustness, generalization, hierarchical learning or generation tasks [141, 142, 143, 144, 145, 146, 147, 148]. However, its role in driving effective exploration remains underexplored and not empirically validated yet. In this chapter, we take an initial step toward understanding whether mechanisms that encourage diversity through skill discovery can also drive more effective exploration. To address this gap, we propose a rigorous methodology to empirically compare knowledge-based and competence-based intrinsic rewards, which has not been thoroughly investigated in prior research. Our work focuses on examining how different levels of diversity in exploration behavior impact exploration, driven by the need to address the following open questions: *i)* What is, in practice, an effective exploration in environments with low- and high-dimensional state spaces? *ii)* How does the level of diversity imposed by intrinsic rewards affect exploration performance across different scenarios? *iii)* Does behavioral diversity through skill discovery, known to help robustness and fast adaptation [143, 144], also helps exploration? Our key contributions are as follows:

1. We introduce a refined categorization of intrinsic rewards based on four diversity levels—State, State + Dynamics, Policy, and Skill—offering a more granular understanding of their influence on exploration.

2. We design an empirical study that assesses how these different diversity levels impact exploration by incorporating multiple complementary exploration metrics such as return, coverage, entropy, reward findings, and state visitation maps.

3. We provide empirical insights into the role of diversity in exploration, offering practical guidance to leverage intrinsic rewards for environments with varying exploration challenges.

Table 4.1: Comparison of existing works on intrinsic rewards (IR).

Study	Paper Category	Pros	Cons/Limitations
[25]	Categorization of IR	Categorized IR into reward novel states and reward diverse behaviors.	Lack of empirical testing in a common framework.
[124]	Categorization of IR	Categorized IR into knowledge acquisition and skill-learning.	Lack of empirical testing in a common framework.
[137]	Categorization of IR	Categorized IR into surprise, novelty and skill-learning.	Lack of empirical testing in a common framework.
[123]	Categorization of IR	Categorized IR into blind, uncertainty, space coverage and self-generated goals.	Lack of empirical testing in a common framework.
[129]	Categorization of IR	Categorized IR into knowledge and competence-based (focused on goal-conditioned RL).	Lack of empirical testing in a common framework.
[149]	Categorization of IR	Categorized IR into prediction-error, novelty and information gain.	Lack of empirical testing in a common framework.
[130]	Categorization of IR	Categorized IR into knowledge-based and competence-based.	Lack of empirical testing in a common framework.
[150]	Empirical study on IR	Evaluated performance of knowledge-based IR (State Count, RND, ICM, RIDE) on MiniGrid. Analyzed different weighting methods for IR and different neural network architectures.	Did not include any skill-learning methods from the competence-based category. Focused on return performance without directly studying exploration.
[151]	Empirical study on IR	Compared the performance of knowledge-based IR (pseudo-counts, RND, ICM, Noisy Networks) on ALE environment.	Did not include any skill-learning/goal-conditioned methods. Evaluated performance solely based on return with no specific focus on the exploration behavior.
[152]	Empirical study on IR	Compared the performance of knowledge-based IR (ICM, RND, Disagreement, NGU, pseudo-counts, RIDE, RE3, and E3B). Addressed key design and optimization details of IR to establish standardized implementations.	Did not include skill-learning/goal-conditioned methods. Did not provide any link between diversity and exploration.
[153]	Categorization and empirical study on IR	Tested methods from knowledge-based (ICM, Disagreement, RND), competence-based (DIAYN, SMM, APS) and data-based (APT, ProtoRL) categories on continuous control tasks. Evaluated their generalization capabilities in an unsupervised pretraining followed by supervised finetuning framework.	Did not consider the joint optimization of IR and extrinsic rewards. Focused on generalization and fast adaptation rather than studying the impact of diversity on exploration.
[154]	Categorization and empirical study on IR	Divided IR between lifelong (global) and episodic bonuses. Tested different combinations of global and episodic bonuses on MiniGrid, in sparse reward and pure exploration settings. Analyzed why lifelong IR do not contribute much in improving exploration.	Focused on global vs episodic perspective, not on diversity and its impact on exploration.
[155]	Categorization and empirical study of IR	Studied the advantages and disadvantages of global and episodic IR for exploration in contextual MDPs.	Interpreted IR from the global/episodic perspective, but not from a diversity perspective.
[156]	General framework unifying IR	Interpreted IR as special cases of conditional prediction with different mask distributions.	Lack of a comparative empirical study.
[157]	General framework unifying IR	Reformulated the convex MDP problem as a convex-concave game and interpreted several RL algorithms (including skill-based IR) as instances of it.	Lack of a comparative empirical study.

To achieve this, we evaluate representative intrinsic reward methods from each diversity level on MiniGrid [158], using both grid encodings and RGB (Red, Green, Blue) observations. This setup allows us to analyze how diversity shapes agent behavior in exploration-critical environments. To the best of our knowledge, this is the first systematic evaluation of diversity levels in intrinsic rewards within a unified framework, offering novel insights into their influence on exploration and performance.

4.2 Related Works

While numerous intrinsic reward formulations have been proposed to address complex sparse-reward tasks, a comprehensive understanding of their comparative advantages and challenges remains elusive, leaving this an open question in the field. Here, we review previous works that have attempted to categorize or empirically compare intrinsic rewards. Table 4.1 provides an overview of these studies, highlighting the pros and cons of each approach.

Existing surveys [137, 25, 124, 123, 129, 149, 130] offer slightly different taxonomies of intrinsic rewards, often using varied terminology. However, most include two broad categories: one focused on increasing knowledge about the environment (e.g., prediction error, information gain, learning progress, and state novelty), and another focused on learning diverse skills. Yet, these surveys lack empirical validation and none of them explore the different levels of diversity that these intrinsic rewards can introduce within each category. In this work, we build on the categorization proposed by [129], which clearly distinguishes between knowledge-based and competence-based intrinsic rewards, and we further subdivide them into different diversity levels (state/state+dynamics/policy/skill).

We are now interested in the works provided in the literature aimed at benchmarking different intrinsic rewards. A few studies have compared methods within the knowledge-based category. For instance, [150] compared State Count [159], Random Network Distillation (RND) [133], Intrinsic Curiosity Module (ICM) [135], Reward Impact Driven Exploration (RIDE) [160] on MiniGrid environment. The study aimed to evaluate the impact that weighting intrinsic rewards has on performance, as well as the effect of using different neural network architectures. The main insight from this work was that no single intrinsic reward method consistently outperforms the others across all tasks, and the performance is highly sensitive to the choice of network architecture and reward scaling. Another study by [151] evaluated pseudo-counts [131], RND, ICM and Noisy Networks [161] within the Arcade Learning Environment (ALE) [162], and suggested that none of these methods outperform the epsilon-greedy exploration. A more recent work by [152] introduced RLeXplore, a comprehensive plug-and-play framework that implements ICM [135], RND [133], Disagreement [163], Never Give Up (NGU) [134], pseudo-counts [131], RIDE [160], Random Encoders for Efficient Exploration (RE3) [164], and Exploration via Elliptical Episodic Bonuses (E3B) [165]. Their framework addressed critical design, implementation, and optimization issues related to intrinsic rewards, including reward and observation normalization, co-learning dynamics of policies and representations, weight initialization, and the combined optimization of intrinsic and extrinsic rewards. The study most similar to ours is by [153] which evaluated intrinsic rewards across knowledge-based (ICM, Disagreement, RND), competence-based (DIAYN, State Marginal Matching (SMM), Active Pretraining with Successor Features (APS)), and data-based (Active Pretraining (APT), ProtoRL) categories on the DeepMind Control Suite. However, their primary objective was to assess the generalization of unsupervised RL algorithms by measuring how quickly they adapted to diverse downstream tasks. To achieve this, they used a reward-free pretraining phase followed by supervised finetuning. In contrast, our study focuses on the standard RL setting, where both intrinsic and extrinsic rewards are optimized simultaneously (except for skill-based learning). Instead of concentrating on adaptation, we address the exploration challenge, evaluating intrinsic rewards from a diversity perspective and employing various metrics to measure exploration quality.

Other works have examined a different taxonomy of intrinsic rewards: global vs. episodic bonuses. Global bonuses are calculated using the entire training experience, while episodic bonuses are calculated using only the experience from the current episode. The work by [154] found that episodic bonuses are more crucial than global bonuses to improve exploration in procedurally generated environments such as MiniGrid. A later study by [155] found that episodic bonuses tend to yield better results in situations where there is minimal shared structure across various contexts in MiniHack [166], while global bonuses tend to be effective in cases where there is a greater degree of shared structure.

Additionally, some works aimed to unify different intrinsic reward formulations under a general framework. For instance, [156] proposed a unified framework for intrinsic rewards, showing that existing methods can be viewed as special cases of conditional prediction with different mask distributions. Building on this, they introduced a novel trajectory-level exploration intrinsic reward, which extends beyond the typical one-step future prediction to capture transition dynamics across longer time horizons. In a related line of work, [157] reformulated the convex MDP problem as a convex-concave game between an agent and an adversarial player generating costs (negative rewards). They unified a broad range of RL algorithms, including methods for unsupervised skill discovery, by interpreting them as instances of this generalized game-theoretic framework.

Despite these significant advances in categorizing, evaluating, and interpreting intrinsic rewards in RL, a critical gap remains: the impact of diversity in intrinsic rewards on the exploration performance has not been thoroughly examined. Specifically, it is unclear how the exploration performance of competency-based methods, which encourage various behaviors, compares to knowledge-based methods that promote various states. In this study, we provide an initial empirical investigation into the impact of different levels of diversity on exploration across several MiniGrid environments, serving as a preliminary effort to understand the complex relation between diversity and exploration in RL.

4.3 Methodology

In the following, we subclassify the knowledge and competence-based intrinsic reward methods according to the level of diversity they impose on the agent's exploration (Section 4.3.1). Then, we select four intrinsic rewards, one for each level (Section 4.3.2), and we test them empirically on MiniGrid environment, explained and motivated in Section 4.3.3. Section 4.3.4 outlines the experimental protocol used in the study, while Section 4.3.5 details the model architecture. Finally, Section 4.3.6 introduces the evaluation metrics.

4.3.1 Taxonomy of Diversity Levels Imposed by Intrinsic Reward

We systematize the types of diversity imposed by intrinsic rewards into four levels: **State level diversity** encourages exploration of unseen states, pushing the agent towards areas where its knowledge is most limited. **State + Dynamics level diversity** also focuses on diverse states, but additionally considers the novelty of the dynamics between those states for a more comprehensive exploration. **Policy level diversity** explores the impact of different actions from given states, while **Skill level diversity** explores the effectiveness of diverse skills (policy-goal association) in achieving goals [129]. For a more detailed description of these diversity levels, please refer to Appendix A.1.

4.3.2 The Selected Intrinsic Reward Algorithms

We augment the task reward with an intrinsic reward such that the total reward becomes: $r^{total} = r^{ext} + \beta^{int} * r^{int}$, where r^{ext} is the extrinsic reward, r^{int} is the intrinsic reward and β^{int} is the intrinsic reward coefficient [25]. The best-performing β^{int} values, either sourced from the literature [150] or determined through a grid search (details provided in Appendix A.3), are presented in

Table A.1, also located in Appendix A.3. We select four different intrinsic reward methods, each representative of one of the four diversity levels:

State Count (State level diversity) builds an intrinsic reward inversely proportional to the state visitation count [159]. For a transition (s_t, a_t, s_{t+1}), where s_t is the current state, a_t is the current action and s_{t+1} is the next state, $r_t^{int} = 1/\sqrt{N(s_{t+1})}$, where $N(s_{t+1})$ represents the number of times s_{t+1} has been visited during training. This algorithm considers only discrete, low-dimensional state space. However, for RGB observations, where the state space is much larger and State Count is not feasible, we use SimHash [132] to hash states before counting them. SimHash maps the pixel observations to hash codes according to the following equation, with ψ as the hashing function: $\psi(s_{t+1}) = sgn(A * \phi(s_{t+1})) \in \{-1, 1\}^m$. Here, ϕ is an embedding function, A is a matrix with i.i.d. entries drawn from a standard normal distribution, m is the size of the hashed key, and $sgn(\cdot)$ maps a number to its sign. Then, the same intrinsic reward formula is applied but using the hashed observation: $r_t^{int} = 1/\sqrt{N(\psi(s_{t+1}))}$.

ICM (State + Dynamics level diversity) uses curiosity as an intrinsic reward. Curiosity is formulated as the error in the agent's ability to predict the outcome of its own actions in a learned state embedding space [135]. Specifically, ICM trains a state embedding network, a forward and an inverse dynamic model. For a transition tuple (s_t, a_t, s_{t+1}), the embedding network $\phi : \mathcal{S} \rightarrow \mathcal{F}$ projects the current state s_t and next state s_{t+1} into the feature space $\mathcal{F}$ to get the embeddings $\phi(s_t)$ and $\phi(s_{t+1})$ respectively. Then, the inverse dynamics model $g : \mathcal{F} \times \mathcal{F} \rightarrow \mathcal{A}$ takes as input the current and next state embeddings, $\phi(s_t)$ and $\phi(s_{t+1})$ respectively, and predicts the action a_t taken by the agent to move from state s_t to state s_{t+1}. The state embedding network is updated, such that it only captures the features of the environment that are controlled by the agent's actions, and ignores the uncontrollable factors. The forward dynamics model $f : \mathcal{F} \times \mathcal{A} \rightarrow \mathcal{F}$ predicts the next state embedding $\phi(s_{t+1})$ given the current state embedding $\phi(s_t)$ and current action a_t.

The intrinsic reward is the prediction error of the forward dynamics model: $r_t^{int} = \|f(\phi(s_t), a_t) - \phi(s_{t+1})\|_2^2$ [135].

Max Entropy RL (Policy level diversity) augments the extrinsic reward with the policy entropy $r_t^{int} = H(\pi(.|s_t))$ to favor stochastic policies [167, 168].

DIAYN (Skill level diversity) aims to discover a set of diverse skills without supervision [138]. A skill is defined as a policy $\pi(a|s,z)$ conditioned on the state s and latent variable/goal z[1]. DIAYN's objective is to maximize the mutual information (MI) between z and every state in the trajectory generated by $\pi(a|s,z)$. The intuition is to infer the skill from the state. At the start of each episode, a latent variable z is sampled from a uniform distribution $p(z)$, then the agent acts according to that skill $\pi(a|s,z)$ throughout the episode. A discriminator $q_\alpha(z|s)$ parametrized by α is trained to estimate the skill z from the state s. The intrinsic reward, defined by $r_t^{int} = log(q_\alpha(z|s_{t+1})) - log(p(z))$, is used to push the agent to visit states that are easily distinguishable in terms of skills. Then, the discriminator is updated to better predict the skill, and the policy is updated to maximize r^{int} using any RL algorithm. It is worth mentioning that DIAYN has been proposed as an unsupervised skill discovery method to favor robustness, fast adaptation to new tasks and hierarchical learning. Therefore, exploration is not the main goal of DIAYN. As a consequence, DIAYN's intrinsic reward can conflict with the agent's extrinsic reward, potentially jeopardizing convergence if combined directly. To address this, we split the training budget between pretraining and finetuning phases. During pretraining, skills are learned using only intrinsic rewards. The learned weights are then used to initialize the policy and value networks for the finetuning phase with task-specific extrinsic rewards. The rationale for this approach is further explained in Appendix A.5, where we evaluate the performance of DIAYN when combined with extrinsic rewards.

[1]In this chapter, z denotes a latent skill variable. The symbol z may have different meanings in other chapters: in Chapter 5, it denotes a state-action pair (s,a); in Chapter 6, it denotes a pair of actions (x,x').

4.3.3 Environment

We test on MiniGrid [158], a widely used procedurally generated environment in RL exploration benchmarks [160, 150, 154] suitable for experimenting with sparse rewards, as its tasks provide rewards only upon reaching goals, encouraging agents to explore efficiently. We consider two types of observations: partially observable grid encodings, and partially observable RGB images (see Appendix A.2). The latter has a much larger state space, allowing us to investigate the scenarios challenging for State level diversity algorithms. To study the impact of different diversity levels on exploration, we select four environments with varying grid layouts and tasks, that highlight the strengths and weaknesses of various intrinsic reward methods:

1. **Empty**: We choose this environment as a control and it is the only one not procedurally generated (fixed initial and goal positions). The setup imposes minimal constraints, providing freedom to solve the task in different ways. Consisting of one big homogeneous room, this environment is interesting since it can lead to state aliasing: different MDP states, for example, different (x, y) positions of the agent, appear as identical observations [169]. This creates a challenge for state count-based methods, which count the observations they encounter and therefore cannot differentiate between the true underlying states.

2. **DoorKey**: This environment requires strategic exploration to locate keys and unlock doors. It stresses the importance of a trajectory to visit states in particular order. Methods that can learn skills and recognize these dependencies might perform better than state count-based methods, which treat all state visits equally without taking into account the order.

3. **FourRooms**: This environment is characterized by its sparsity of rewards. The presence of multiple rooms encourages the agent to devise different strategies for navigation, fostering diversity in the trajectories

or paths taken by the agent to achieve the goal.

4. **RedBlueDoors**: This environment also requires strategic exploration, but it is an easier task than DoorKey. It introduces color-coded doors, requiring agents to exhibit high levels of cognitive flexibility.

More details about the tasks, observation and action spaces are included in Appendix A.2.

4.3.4 Experimental Protocol

We test the four algorithms in each environment for each observation space. We select Proximal Policy Optimization (PPO) [42] as our baseline algorithm. PPO is a widely accepted and popular choice in RL research, known for its stability, robustness, and relatively high sample efficiency. Its simple implementation offers manageable computational costs, which enhances reproducibility and facilitates validation of results. Beyond its theoretical strengths, PPO has demonstrated success in complex real-world applications, such as Large Language Model (LLM) research, underscoring its versatility and reliability for our study. For the pseudocode of the algorithm, please refer to Algorithm 6 in Chapter 2.

We adopt the default hyperparameters from [150], listed in Table A.2 of Appendix A.3. This baseline algorithm comes with an entropy regularization in the objective function to encourage a minimum level of exploration [170]. Such regularization is essential to avoid overfitting [171] and to stabilize the training process [168]. We set the entropy regularization coefficient to 0.0005 in all simulated algorithms. The selected value is large enough to guarantee a minimum level of stable convergence but small enough to not affect our experiments. We train each algorithm for 40M frames on all environments. For DIAYN exclusively, we use 25M for pretraining and 15M for finetuning. Training curves, averaged over five runs with different seeds, are provided for all algorithms. The simulations in this study were conducted on a high-performance computing node equipped with an NVIDIA TITAN X (Pascal)

GPU featuring 12 GB of VRAM, an Intel Xeon E5-2640 v4 CPU operating at 2.40 GHz with 40 cores, and 62 GB of RAM.

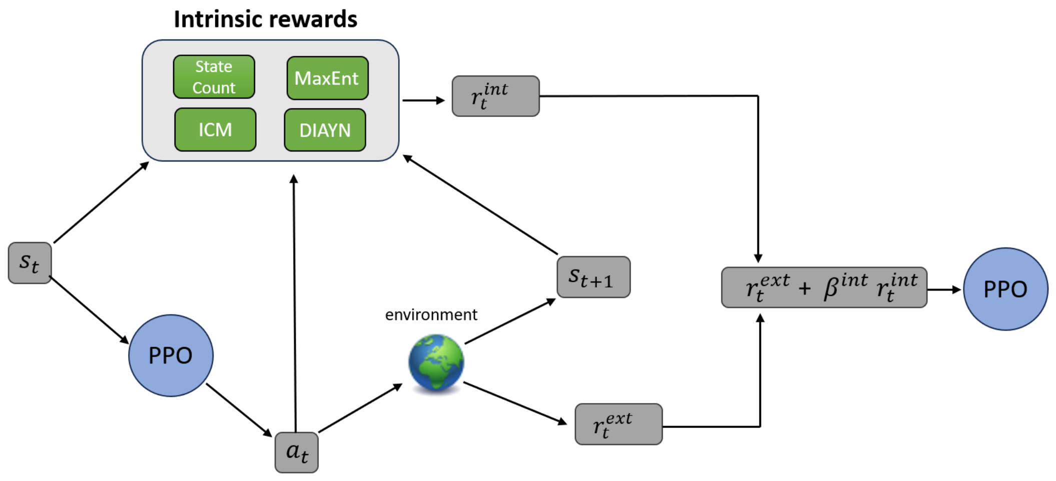
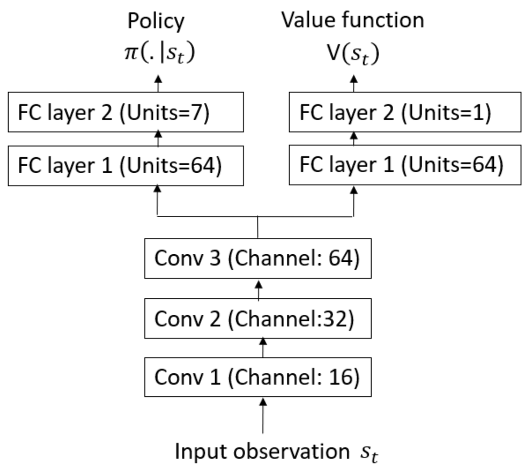


Figure 4.1: Overview of the empirical study pipeline, illustrating the flow from input observations to action selection, and reward computation (both extrinsic and intrinsic) within the PPO framework.

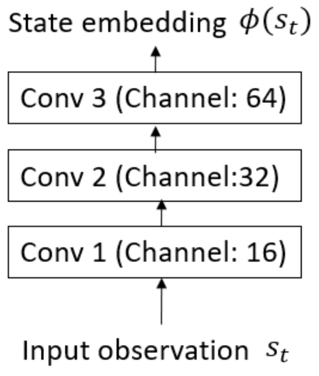
4.3.5 Model Architecture

Figure 4.1 illustrates the pipeline of the empirical study, depicting the sequential flow of inputs, outputs, and reward computations within the model. At each time step t, the input observation s_t (either a grid encoding or an RGB image) is processed by the PPO algorithm, which outputs an estimated policy and value function. The agent then takes an action a_t based on the estimated policy, transitions to the next state s_{t+1}, and receives an extrinsic reward r_t^{ext}. Depending on the intrinsic reward method applied (State Count/SimHash, ICM, DIAYN, or Max Entropy), the agent computes an intrinsic reward r_t^{int} for the transition (s_t, a_t, s_{t+1}), following the formulations in Section 4.3.2. The intrinsic and extrinsic rewards are combined $r_t^{total} = r_t^{ext} + \beta^{int} * r_t^{int}$ and fed back into the Actor-Critic PPO network to refine the policy and value function. For DIAYN exclusively, we avoid combining intrinsic and extrinsic rewards, as discussed in Section 4.3.2.

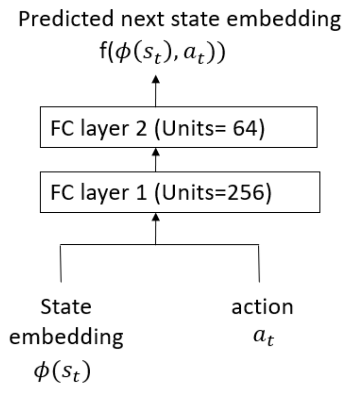
The Actor-Critic (AC) model architecture used in PPO employs a shared CNN to process observations, which can be either grid encodings or RGB



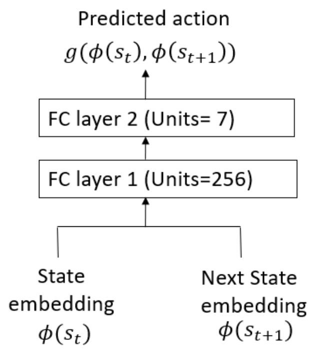
(a) AC network



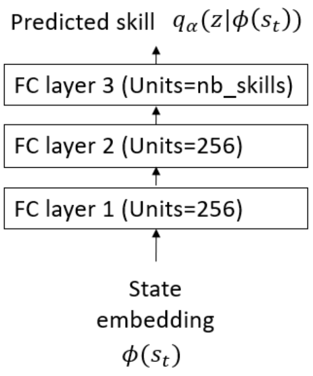
(b) State Embedding



(c) Forward dynamics network


Predicted action
$g(\phi(s_t), \phi(s_{t+1}))$
FC layer 2 (Units= 7)
FC layer 1 (Units=256)
State embedding $\phi(s_t)$
Next State embedding $\phi(s_{t+1})$

(d) Inverse dynamics network


Predicted skill $q_\alpha(z|\phi(s_t))$
FC layer 3 (Units=nb_skills)
FC layer 2 (Units=256)
FC layer 1 (Units=256)
State embedding $\phi(s_t)$

(e) Discriminator network

Figure 4.2: Neural Network Architectures.

images. This CNN consists of three convolutional layers: the first layer has 16 filters of size 2×2 with Rectified Linear Unit (ReLU) activation, followed by a 2×2 max-pooling layer; the second layer has 32 filters of size 2×2 with ReLU activation; and the third layer has 64 filters of the same size and activation function. The CNN output then branches into two fully connected networks, designated as the actor and critic networks. Each network includes a hidden layer with 64 units and Tanh activation. The actor network produces action probabilities, while the critic network outputs the value function. Figure 4.2a provides an overview of this architecture.

The PPO architecture remains consistent across all intrinsic rewards. Some methods, however, require additional auxiliary networks, such as the embedding networks ϕ for ICM, DIAYN, and SimHash (see Figure 4.2b), forward (f) and inverse (g) dynamics networks in ICM (Figures 4.2c and 4.2d), and the discriminator network q_α in DIAYN (Figure 4.2e). For ICM and DIAYN, the state embedding network follows the same CNN architecture as PPO to extract features from observations (see Figure 4.2b). SimHash further appends a fully connected layer to the embedding network, reducing the RGB image embedding to a 512-dimensional vector prior to hashing.

4.3.6 Evaluation Metrics

We analyze each of the intrinsic rewards, according to five metrics:

Episodic return: This metric measures the total extrinsic reward accumulated within a single episode: $\sum_{t=1}^{H} r_t^{\text{ext}}$, where r_t^{ext} is the extrinsic reward received at timeframe t and H is the length of the episode. We report the average episodic return for all actors. This metric captures the convergence speed and learning ability of the intrinsic reward method.

Observation coverage: This metric offers insight into the extent of exploring the observation space. We count how many unique observations (grid encodings or RGB) have been visited during training. We normalize this metric over the highest coverage achieved by the intrinsic reward methods.

Agent's position coverage: This metric indicates the proportion of (x,y) grid positions visited by the agent so far during training, calculated as: $\frac{N_{visited}(x,y)}{N_{total}(x,y)}$. Here, $N_{visited}(x,y)$ is the number of unique grid positions the agent has visited, and $N_{total}(x,y)$ is the total number of possible grid positions the agent can visit. This metric captures how well the agent has explored the position space, which is different from the observation space in a partially observable framework.

Policy Entropy: This metric evaluates the stochasticity of the policy. For a given state s_t, the Shannon entropy of the policy is defined as $H(\pi(\cdot \mid s_t)) = -\sum_{j=1}^{|A|} \pi(a^j \mid s_t) \log \pi(a^j \mid s_t)$, where a^j denotes one of the $|A|$ possible actions in the MiniGrid environment ($|A| = 7$ in our case). In practice, we report the average of $H(\pi(\cdot \mid s_t))$ across visited states and actors during training.

Time steps of the first, second, and third reward discoveries: We record the number of frames at which the agent, using a particular intrinsic reward method, successfully reaches a sparse reward for the first, second, and third time. Note that "number of frames" refers to the number of times the agent interacted with the environment throughout the training. This metric sheds light on the speed and effectiveness of the exploration method to discover the high-reward states, as well as learning to revisit these states.

Finally, we include further visualizations (heatmaps) of the state visitation count ((x,y) positions) in Appendix A.4. These heatmaps represent the proportion of visits to each grid position (x,y) relative to the total number of frames. To generate them, we train the agent for 10M frames in singleton environments, where the maze layout remains fixed across training episodes. This setup highlights the agent's exploration patterns on a consistent grid map. Figures A.3, A.4, A.5, and A.6 in Appendix A.4.1 display results for grid-encoded observations, while Figures A.7, A.8, A.9, and A.10 in Appendix A.4.2 show results for RGB observations. These visualizations illustrate the areas of the grid explored by the agent during training across the four environments: Empty, DoorKey, FourRooms, and RedBlueDoors.

4.4 Experimental Results and Discussion

We discuss the following three questions to analyze the performance of the exploration algorithms:

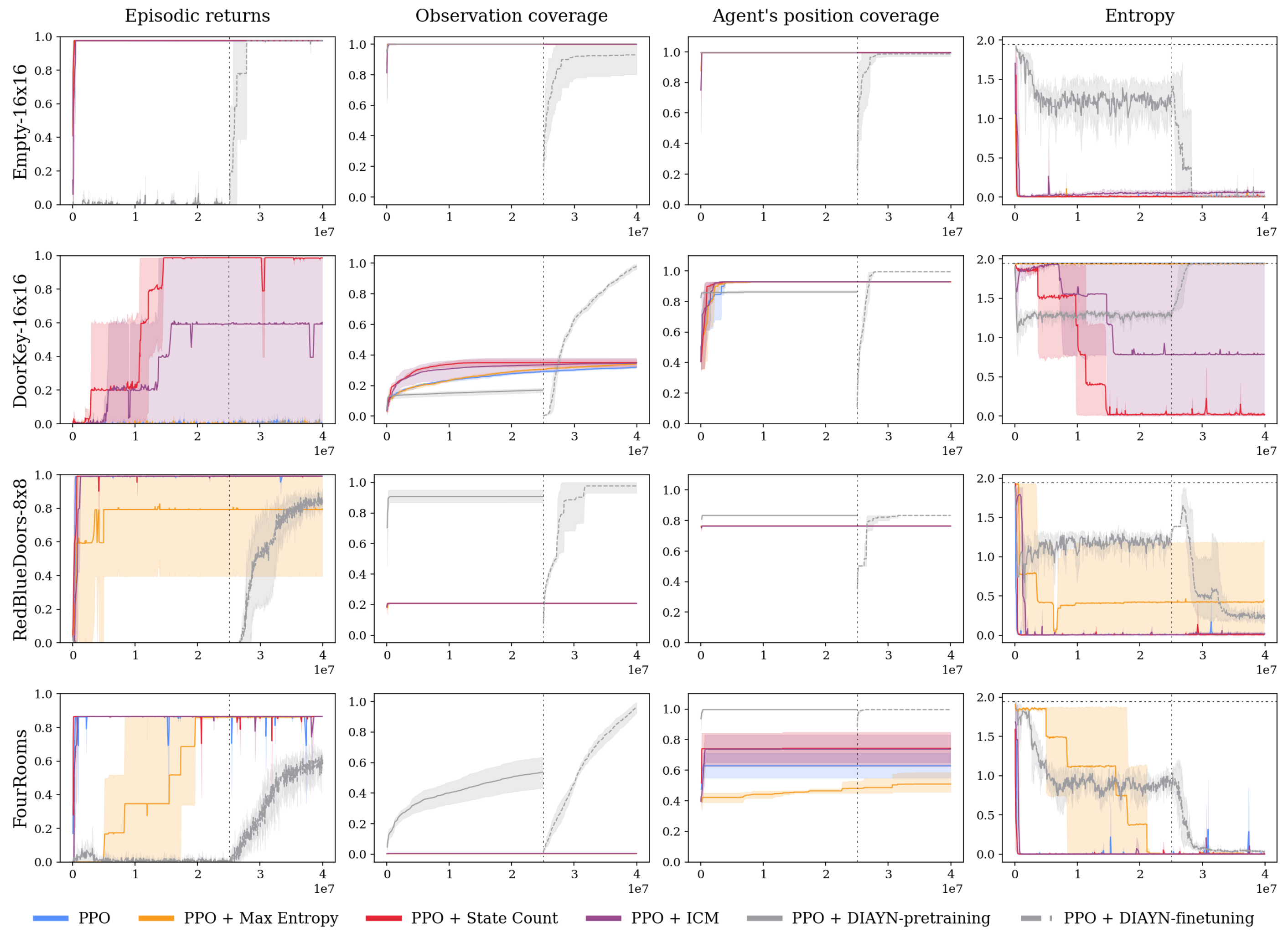


Figure 4.3: The four metrics against the number of transitions (frames) processed by the environment. Observations are grid encodings. Results are averaged over five seeds with standard deviation shading. Vertical dash-dot lines mark the start of DIAYN finetuning. Horizontal dash-dot lines mark the theoretical maximum entropy of the policy $H_{max}(\pi) = \log(|A|)$.

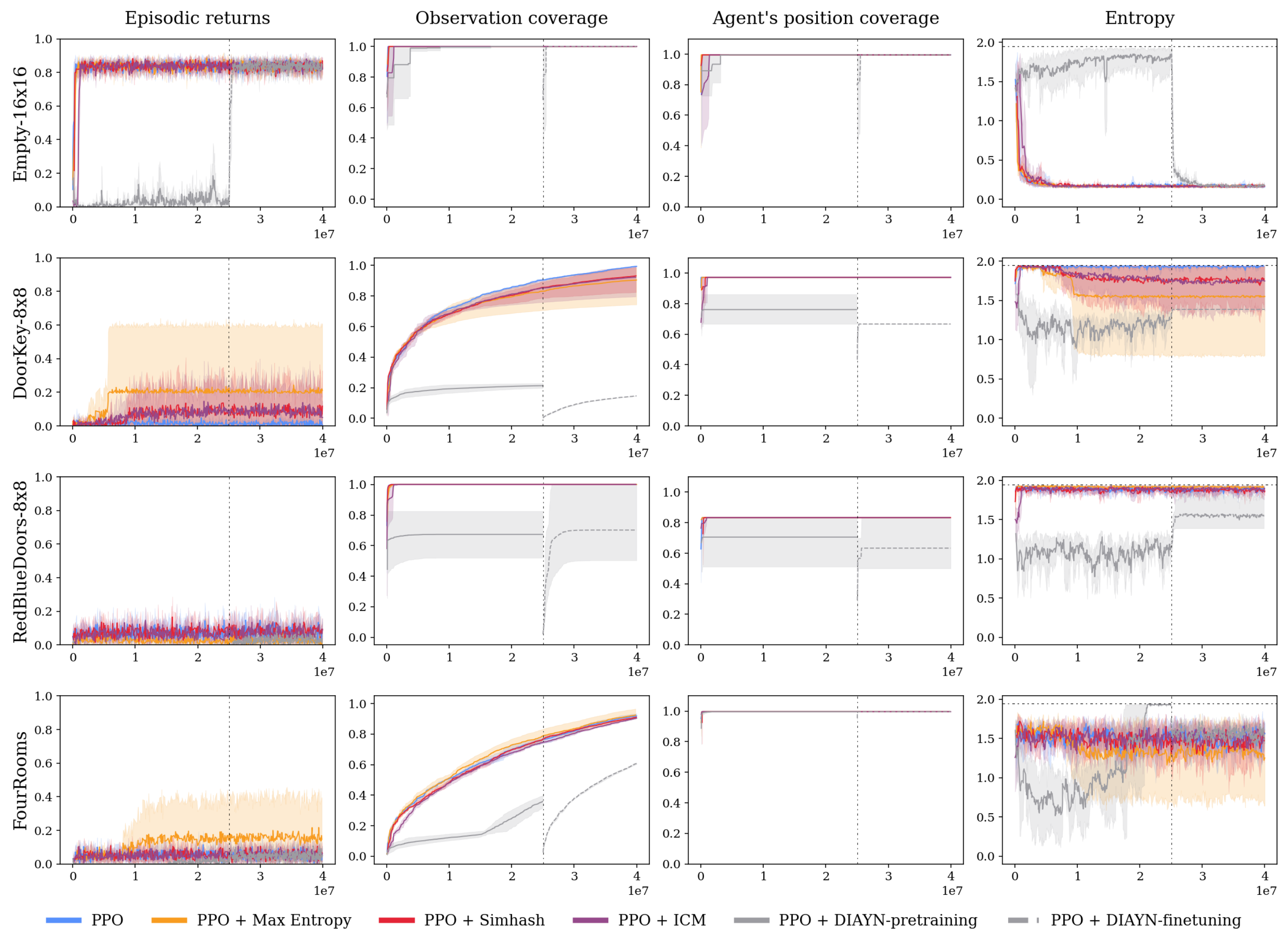


Figure 4.4: Analogous to Figure 4.3, but observations are partial RGB images.

- RQ1: Do different intrinsic rewards lead to different return performance/sample efficiency for both grid encodings and RGB partial observations?
- RQ2: What are the characteristics (strengths/weaknesses) of each intrinsic reward method, and what are the practical recommendations to select intrinsic rewards?
- RQ3: How do different intrinsic rewards impact efficiency in discovering the sparse reward? Is there any link with credit assignment?

4.4.1 RQ1: Return Performance of the Different Intrinsic Rewards

In terms of episodic return, State Count has the best performance with low-dimensional observations (grid encodings) on all environments (see column 1 of Figure 4.3). It converges to the maximum return with the least number of frames. In the case of DoorKey 16x16, where many algorithms—including PPO, Max Entropy, and DIAYN—struggle to solve the task, State Count emerges as the top performer, successfully obtaining the key and attaining the highest return. Following closely, ICM demonstrates lower sample efficiency. However, this is not the case for RGB observations (refer to column 1 of Figure 4.4), in which SimHash (equivalent to State Count) performs poorly on most environments. The failure of SimHash in the case of RGB observations can be attributed to the challenge in adequately representing the significant features present in the high-dimensional states. RGB images contain an abundance of extraneous pixel-level details that are irrelevant to the task, requiring the agent to represent only the meaningful features. SimHash, which uses a simple hashing mechanism to represent states, struggles to capture the essential features in RGB states due to their sparse and coarse encoding mechanisms. This limitation is especially evident in environments that require high level of feature abstraction and attention to object relationships, such as DoorKey 16x16, where misrepresenting critical details hinders the agent's navigation.

Max Entropy is less impacted by such representation learning difficulties. It achieves a slightly higher return on DoorKey 8x8 and FourRooms environments in the case of RGB observations (Figure 4.4). This robustness can be attributed to Max Entropy's tendency to encourage diverse policy exploration without heavily relying on specific state representations, which provides a certain level of resilience to noisy feature extraction. All other intrinsic rewards struggle to solve the tasks (except for Empty 16x16) and consistently maintain a high level of non-decreasing policy entropy. This is likely because these methods rely on high-quality state representations to produce meaningful novelty signals. In high-dimensional RGB observations, however, they tend to generate less informative intrinsic rewards. This results in difficulty differentiating between truly novel states and irrelevant pixel-level variations, causing policy learning to stagnate.

DIAYN finetuning has a worse average return compared to the baseline PPO in both grid encodings and RGB scenarios. This shows that initializing the AC weights with DIAYN skills does not improve sample efficiency compared to random initialization. Note that DIAYN pretraining does not collect any extrinsic reward because it is trained to maximize the intrinsic reward generated by the discriminator and not the true task reward. We hypothesize that the limited skill label space, compared to the vast state space, promotes the learning of static skills that lack adaptability and fail to transfer effectively to the target task. Specifically, the states encountered by different skills tend to vary only slightly, enabling skill differentiation but not necessarily the development of semantically meaningful or broadly transferable skills.

4.4.2 RQ2: Characteristics of Each Intrinsic Reward Algorithm

State Count / SimHash demonstrates the best sample efficiency in grid encodings, enabling efficient task-solving in small state/action spaces. Additionally, it ensures a fast coverage of observations and grid positions compared to other algorithms, as depicted in columns 2 and 3 of Figures 4.3 and 4.4.

Furthermore, as it converges to the optimal policy, it exhibits a fast decreasing policy entropy due to the diminishing intrinsic reward effect with increasing state counts. Examination of the heatmaps (Appendix A.4) reveals that State Count offers the most uniform coverage of the state space across all environments. This enables the algorithm to identify the optimal path, while maintaining a balanced approach between exploration and reward maximization. Remarkably, in the DoorKey environment (Figure A.4 in Appendix A.4), State Count demonstrates a tendency to revisit the area around the key more frequently. However, despite these strengths, a notable limitation arises in its inability to effectively handle RGB images. In such cases, the algorithm struggles to accurately count or represent pixels, thereby limiting its applicability in scenarios with high-dimensional state spaces. As a practical recommendation, State Count is a good choice for small, discrete environments, but struggles with complex, high-dimensional ones. Although we did not explore how different representations impact the performance of State Count in this study, incorporating representation learning techniques presents an interesting avenue for future research.

ICM exhibits favorable return performance and effectively explores the observation and position spaces, similarly to State Count, as they both prioritize exploration within the state space. In environments characterized by low-dimensional state spaces (Figure 4.3), ICM showcases consistent stability in solving tasks across diverse scenarios. However, ICM's convergence speed generally lags behind State Count due to the added computational complexity of training both forward and inverse dynamics models. This additional overhead likely introduces inefficiencies that slow down exploration, as shown in heatmap analyses (Appendix A.4), where ICM's slower rate of grid position exploration is evident. These heatmaps illustrate that while ICM achieves thorough state coverage, it does so at a slower rate, potentially limiting its efficiency in tasks requiring rapid convergence. Moreover, similarly to State Count, ICM encounters challenges in effectively processing RGB images (Figure 4.4). The pixel-based inputs add significant complexity, making it difficult for ICM's dynamics models to effectively process and encode meaningful features. This limitation suggests that ICM's performance may be hampered in visually complex environments.

Max Entropy can solve most environments in the case of grid encoding observations (except DoorKey) (Figure 4.3). However, it does not converge faster than State Count and shows a slightly lower average return because it fails for some of the runs (such as on RedBlueDoors). This instability arises from the algorithm's tendency to promote high stochasticity in the policy, even when a more deterministic approach would suffice, ultimately affecting the average performance. By analyzing the heatmaps, we can see that Max Entropy explored unnecessarily or became confined to certain regions of the state space, especially in easy environments such as FourRooms and RedBlueDoors (Figures A.5 and A.6 in Appendix A.4). This unnecessary exploration delays convergence to optimal paths, as the agent is distracted from effectively reaching the goal. The algorithm's inclination to prompt the agent to try all possible actions, including those rarely relevant to task success, can divert focus and hinder progress. Additionally, a drawback of the Max Entropy approach is that states with lower entropy may be visited less frequently or even overlooked. As discussed by [172], the maximum entropy strategy, which optimizes policies to reach high-entropy states, does not always foster effective exploration. Rather, it can create positive feedback loops where the agent becomes overly focused on high-entropy areas, limiting its ability to comprehensively explore the environment. This might reduce the likelihood of reaching less-visited yet potentially critical states.

Nevertheless, in the case of partial RGB observations (Figure 4.4), Max Entropy is less impacted by representation learning challenges. We observe that on DoorKey and FourRooms, it slightly outperforms SimHash in terms of return and shows a decrease in the policy entropy (see columns 1 and 4 of Figure 4.4), as it succeeds in reaching the goal in several runs. Therefore, for grid-based settings with high-dimensional state spaces, where simply counting states becomes impractical, maximizing entropy can be a valuable alternative exploration strategy to State Count. As a practical recommendation, Max Entropy may not be the most effective exploration method in grid-like environments with high-dimensional action spaces, where many actions are unused. However, it performs adequately in environments with large state spaces and small action spaces.

DIAYN generally has the worst average return compared to the other three intrinsic rewards in both grid encodings and RGB scenarios. This is attributed to the tradeoff between the ability to discriminate between different skills and optimality. The need to generate distinguishable skills often leads DIAYN to prioritize visits to easily discriminable states over achieving optimal exploration. In the case of low-dimensional state space (Figure 4.3), it is surprising that DIAYN finetuning has the highest observation and position coverages on most environments (DoorKey, FourRooms and RedBlueDoors). The ease of discriminating observations (due to distinct grid encodings reflecting different object types, colors, or status) drives DIAYN to prioritize visiting them. Unlike environments with distinct features, DIAYN struggles to cover the observation space in Empty 16x16 due to the difficulty of discriminating observations in a near-uniform grid (mostly walls). This further underscores DIAYN's reliance on environments with clear, discriminable features for effective exploration. Moreover, the poor state space coverage by DIAYN (both pretrained and finetuned) in the RGB setting (Figure 4.4) indicates limitations in the discriminator's ability to discriminate between RGB observations. This suggests that the additional challenge of representation learning exacerbates the discriminator's learning difficulties. The presence of high-dimensional visual data introduces an added layer of complexity as the agent must learn both to distinguish visual features and navigate the space effectively. By further analyzing the exploration pattern of DIAYN through the heatmaps, we notice the following: DIAYN demonstrates uneven state coverage, often focusing on corner areas or becoming restricted to specific regions within the grid that contain easily distinguishable states (For example, see Figures A.4, A.6, A.9, and A.10 in Appendix A.4). This suggests potential limitations in its ability to explore diverse regions and acquire transferable skills. Without reaching different target positions (e.g., door/key/goal), the skills lack meaningful variations and adaptability. We hypothesize that this is due to DIAYN's MI objective, which does not explicitly maximize the entropy of the state distribution [173] and does not promote broad state coverage [174]. The agent tends to receive higher rewards for visiting known states rather than exploring novel ones, as fully discriminable states yield a high MI reward [175]. This can hinder novel state exploration and discourage the agent from learning far-reaching skills [173]. Consequently, DIAYN might potentially constrain the diversity of learned skills to those that are easier to distinguish but not necessarily effective for broad exploration or task relevance. In our particular setting, DIAYN also encounters difficulty in learning the abstract skill space effectively. This challenge might be particularly pronounced due to partial observability. As a practical recommendation, learning unsupervised skills with DIAYN does not help exploration in MiniGrid framework, especially in strategic tasks. Nevertheless, pushing for diversity of skills can be useful for skill-chaining, fast adaptation to environment changes, robustness, and generalization to different tasks. We emphasize that our results hold only for our particular setting where skill-learning turns out to be antagonistic to exploration and sample efficiency in MiniGrid. This might not hold in other environments that could benefit from such skills to converge faster. It is also worth noting that exploring factors such as the skill space, the initial skill distribution, and incorporating state abstraction techniques, along with auxiliary exploration mechanisms to enhance state coverage of skills, could significantly improve DIAYN's performance. However, because this constitutes a substantial variation from the original algorithm, we leave these considerations for future work.

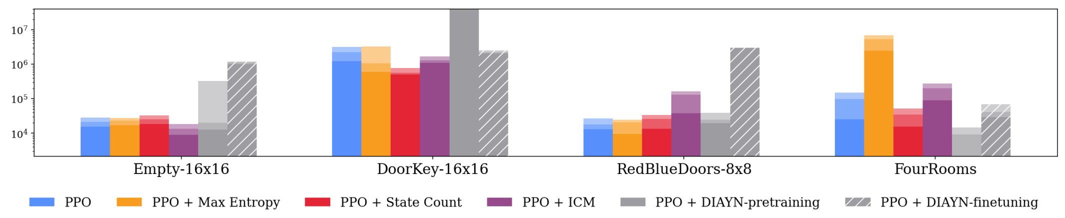


Figure 4.5: Histogram of average frames needed by each exploration method to collect rewards across environments. Observations are grid encodings. Each bar's three fading segments mark the frames at which the first, second, and third rewards are collected; lower values are better. Results are averaged over five runs.

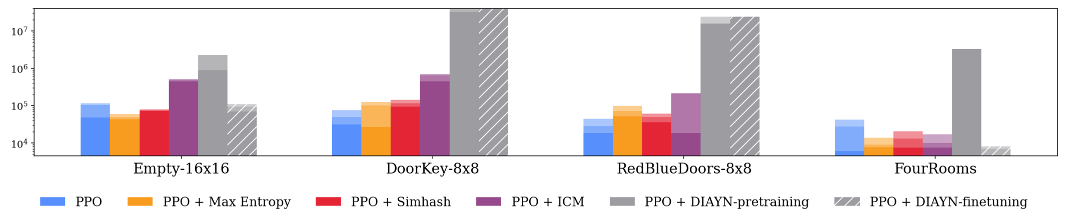


Figure 4.6: Analogous to Figure 4.5 but observations are partial RGB images.

4.4.3 RQ3: First, Second and Third Instances of Discovering the Sparse Reward

We record the time steps at which the sparse reward is found by each of the intrinsic rewards for the first, second, and third times in both grid encodings (Figure 4.5) and RGB (Figure 4.6) scenarios. For more detailed results, including averages and standard deviations, refer to Tables A.3, A.4, A.5, and A.6 in Appendix A.4.1, as well as Tables A.7, A.8, A.9, and A.10 in Appendix A.4.2. We notice that in the case of low-dimensional observation space, State Count (which has the highest return performance) finds the reward soon on most environments, while DIAYN takes time to reach the goal, especially on strategic tasks. For example, in the DoorKey environment, which represents a strategic task, State Count is the first intrinsic reward to find the task reward, while DIAYN finetuning is the last, and DIAYN pretraining does not reach the goal at all within the pretraining time. This shows that DIAYN exhibits limitations in acquiring skills that achieve the goal sequence of visiting the key, the yellow door, and the green goal in this specified order. This limitation is likely due to DIAYN's focus on skill diversity rather than directed exploration, making it less effective in tasks requiring structured sequences. Another interesting observation is that the algorithm that discovers the reward the fastest for the first time, might not be the fastest in visiting the rewarding state a second and third time. This implies that diversity impacts credit assignment. For example, on DoorKey (see Table A.4 in Appendix A.4), Max Entropy finds the first reward before ICM for the first time, but it takes more time to learn that it should go back to the reward for the third time. This is paramount to designing a sample efficient algorithm because visiting rewards more often provides more informative learning signals and allows learning credit faster, more accurately and with less variance [26]. This implies that although Max Entropy promotes policy exploration, it may lack mechanisms for prioritizing or remembering rewarding states that consistently provide useful learning signals. In contrast, the results vary across other environments, highlighting how different algorithms perform under varying conditions. Notably, in the Four-Rooms environment (Figure 4.5), DIAYN pretraining is the first to find the reward, as opposed to the case of strategic tasks. In an environment consisting of several identical compartments, learning skills could lead to quick reward discovery even though it does not directly maximize the task reward. This suggests that DIAYN might be advantageous in environments with structural similarity, where learned skills can be reused across similar compartments. For the case of RGB observations (Figure 4.6), we observe that PPO and Max Entropy are among the fastest methods to find the reward on most environments, surpassing SimHash. This reinforces the hypothesis that, when scaling to high observation spaces, entropy might be a better strategy to push for exploration rather than counting states. DIAYN finetuning also takes a long time to find the reward, especially for strategic tasks such as DoorKey and RedBlueDoors. This suggests that DIAYN's emphasis on diversity may limit its ability to prioritize reaching task-relevant states in complex, sequential tasks. Integrating the MI objective of DIAYN with trajectory-based metrics between states to enhance exploration could be a potential direction for handling strategic tasks.

4.5 Conclusion

In this chapter, we have reinterpreted intrinsic reward techniques in the literature using a diversity perspective (State, State + Dynamics, Policy, and Skill levels of diversity). We conducted empirical studies on MiniGrid, to understand the differences between these diversity levels in a partially observable and procedurally generated framework.

The main outcome of the study is that State Count (representing State level diversity) leads to the best exploration performance in the case of low-dimensional observations. It improves the convergence speed in strategic tasks, covers the state space homogeneously, and results in a rapid decrease in policy entropy. However, State level diversity is fragile and requires good state representations, while entropy maximization seems to be slightly more robust when dealing with image-based observations. Learning good state representations is challenging, so entropy maximization (representing Policy level diversity) is a practical alternative. Lastly, DIAYN (representing Skill level diversity), often associated with improved robustness and generalization, struggles with exploration in MiniGrid due to the difficulty of learning the skill space and exploring within it, in a procedurally generated partially observable setting.

4.5.1 Limitations and Future Work

This study serves as an initial exploration into the relationship between exploration and diversity imposed by intrinsic rewards. While we provide insights into this relationship, several limitations remain to be addressed in future work.

Firstly, we examine only one representative intrinsic reward method for each level of diversity. This choice may not capture the full range of behaviors within each category, potentially limiting the generalizability of our findings. Expanding this work to benchmark a broader selection of intrinsic reward methods would improve the applicability of our results.

Additionally, the effectiveness of intrinsic rewards is closely tied to the environment in which they are applied. Our experiments are restricted to the MiniGrid environment, specifically using grid encodings and RGB observations. Future studies could benefit from exploring more complex and varied environments, such as Mujoco [176], Atari [162], MiniHack [166], and MiniMax (Autocurricula) [177], where the impacts of different diversity levels might yield more distinct behaviors. Some intrinsic reward methods may excel in certain environments but perform poorly in others. Thus, identifying conditions under which each intrinsic reward method performs best across diverse environments would be a valuable contribution.

Moreover, while diversity can enhance exploration, it may also impede performance as discussed in [178] in a phenomenon named *the curse of diversity*. Therefore, pinpointing the conditions under which diversity aids rather than hinders performance—or developing strategies to counterbalance the potential negative effects of diversity—remains an open research question.

For the competence-based category, we employed DIAYN, a method that learns a skill space autonomously. Other goal-conditioned approaches, such as those learning different goal representations [179] or predefining goal abstractions [180] may yield more efficient exploration strategies. Investigating these approaches could offer further insights into competence-based intrinsic rewards.

Finally, representation learning—especially as applied in conjunction with intrinsic reward methods—also significantly impacts exploration efficacy. Analyzing how representation learning interacts with different levels of diversity and affects exploration performance is an important direction for future research.

Chapter 5

Near-Optimal Sample Complexity in Reward-Free Kernel-Based RL

In the previous chapter, we empirically examined the exploration problem in deep RL. Our study showed that different exploration bonuses are effective in different settings, and that exploration behavior can vary substantially across RL scenarios. These findings highlighted both the potential and the limitations of current empirical approaches: while they reveal important patterns, the notion of optimal exploration (i.e., how an agent should explore efficiently to learn an effective policy) remains poorly understood. Moreover, deep RL still lacks a solid theoretical analysis of how many samples are required to explore effectively and converge to a near-optimal policy. This gap between empirical evidence and theoretical guarantees motivates our next step. In this chapter, we build a solid theoretical framework aimed at bridging this gap and providing deeper insight into the foundations of exploration in RL.

While tabular and linear models have been thoroughly explored in RL theory, kernel-based models have recently gained traction for their strong representational capacity and theoretical tractability. They also provide a stepping stone toward understanding more complex nonlinear models, including neural networks, which remain largely theoretical black boxes. Therefore, in this chapter, we examine the question of statistical efficiency in kernel-based RL within the reward-free RL framework. Existing work addresses this question under restrictive assumptions about the class of kernel functions. We begin our investigation under the assumption of a *generative model*, then relax this assumption at the cost of increasing the sample complexity by a factor of H, the length of the episode. Our approach uses a broad class of kernels and a simpler learning algorithm for efficient reward-free exploration compared to prior work, deriving new confidence intervals for kernel ridge regression tailored to our RL setting. We further validate our theoretical findings through simulations.

5.1 Introduction

RL with nonlinear function approximation is a powerful method for learning general Markov Decision Processes (MDPs) through interactions with the environment. Kernel ridge regression for the prediction of the expected value function is perhaps one of the most versatile methods that has gained traction in recent years [31, 181, 182], and lends itself to theoretical analysis. As a burgeoning research area, there are still numerous open problems and challenges in this topic.

We focus our work on statistical aspects of RL within the reward-free RL framework [183, 184, 185], which involves an exploration phase and a planning phase. In the exploration phase, the reward is unknown; the algorithm interacts with the environment to gather information about the underlying MDP, in the form of a dataset of transitions. In the planning phase, the reward is revealed; the algorithm uses the knowledge of the reward and the dataset gathered in the exploration phase to design a near-optimal policy. The planning phase is thus akin to offline RL [186, 187, 188, 189, 190, 191].

From a practical standpoint, this reward-free RL paradigm is particularly well-suited for scenarios involving multiple reward functions of interest, such as in constrained RL [192, 193, 194]. In many applications, it is necessary to modify the reward function to encourage new or more desirable behaviors. For instance, consider the task of training a robot to navigate through an environment while balancing competing objectives such as speed, energy consumption, and safety. Initially, the designer may prioritize efficiency, only to later realize that the policy violates safety constraints or consumes too much energy. Tuning the reward function to encode these preferences typically requires rerunning RL from scratch, which involves additional costly interaction with the environment. To avoid repeatedly invoking the learning algorithm and interacting with the environment, it is desirable for the agent to efficiently explore the environment without access to the reward, collecting a dataset with good coverage over all possible scenarios in the environment. This dataset can then be reused during planning to compute near-optimal policies for a variety of reward functions.

In this chapter, we answer the following fundamental question: *Under some reasonable assumptions on the underlying MDP, what is the minimum number of samples required to enable designing a near-optimal policy?*

We refer to the number of samples as *sample complexity* and measure the optimality of the eventual policy in terms of error in the value function. In particular we refer to a policy as ϵ-optimal if its value function is at most a small $\epsilon > 0$ away from that of the optimal policy for all states.

The reward-free RL framework has been studied in tabular [183] and linear [184, 195, 196] settings. Under the tabular setting, it has been shown that $\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|H^5/\epsilon^2)$ samples are sufficient to achieve an ϵ-optimal policy, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, respectively, and H represents the length of episode. In the linear setting, a sample complexity of $\mathcal{O}(d^3H^6/\epsilon^2)$ has been established that does not scale with the size of the state-action space, but the ambient dimension d of the linear model representing the transition structure of the MDP. With the limitations of the linear model (e.g., as shown in [197]), recent works have considered nonlinear function approximation in RL. The work of [185] considered the reward-free

RL framework with kernel-based function approximation. However, their results only apply to very smooth kernels with exponential eigendecay, such as Squared Exponential (SE), but fail to provide finite sample complexity applicable to a large class of kernels of interest with polynomial eigendecay (see Definition 1), such as Matérn family or Neural Tangent kernels (NTK). This shortcoming arises from the bias in the collected samples. Specifically in the exploration phase of [185], the samples are adaptively collected to achieve a high value with respect to a hypothetical reward—proportional to the uncertainties of the kernel ridge regression—introducing bias to the samples and inflating confidence intervals. Another closely related work on reward-free RL in the kernel setting is [198], which, like [185], uses a hypothetical reward proportional to the uncertainty of kernel ridge regression. However, it improves upon [185] by providing order-optimal sample complexities for kernels with polynomially decaying eigenvalues, where [185]'s results are unbounded. This is achieved via an adaptive domain partitioning procedure inspired by [181]. In this method, the state-action domain is adaptively divided into multiple subdomains as samples are collected, with kernel-based value function estimates constructed based on samples from the same subdomain, while discarding previous observations from other subdomains. Although their approach offers theoretical advantages, it is tedious to implement in practice due to complex domain partitioning structure. Moreover, discarding samples may degrade the empirical performance, a concern that is not addressed in [198]. Additionally, their theoretical results depend on specific assumptions about the relationship between kernel eigenvalues and domain size, which limits generality of their work. A detailed comparison between our work and the two closely related works of [185] and [198] is provided in Section 5.2.2 along with a more comprehensive literature review in Section 5.2.1.

In contrast to the existing work, this chapter establishes near-optimal sample complexities for the reward-free kernel-based RL framework over a general class of kernels, without relying on restrictive assumptions. This is accomplished via a simple algorithm and a novel confidence interval for unbiased samples, broadly applicable to other RL settings (offline RL, model-based, infinite horizon), and supported by empirical evidence. Specifically, we start with a case where a *generative model* [199] is present and it permits the algorithm to sample state-actions of its choice during the exploration phase, not limiting the algorithm to stay on the Markovian trajectory. This setting has been extensively considered in previous work on statistical efficiency of RL (see, e.g., [200, 201, 202, 203, 204, 205]). In the presence of a generative model, we propose a simple algorithm that collects *unbiased* samples by choosing the state-actions with highest kernel-based regression uncertainty at each step. We derive order-optimal sample complexities for this algorithm in terms of $\frac{1}{\epsilon}$, while [198] do not offer any particular advantages in the generative model case. Generative models are applicable in scenarios like games where the algorithm can manipulate the current state, offering insights into the statistical aspects of RL. However, this may not be the case in other scenarios. Inspired by the analysis of the exploration algorithm with a generative model, we propose a second online exploration algorithm that collects samples adhering to the Markovian trajectory. We prove that this relaxing of generative model requirement incurs merely an H factor increase in the sample complexity.

To highlight the significance of our results, we consider kernels with polynomial eigendecay that are of practical and theoretical interest [67, 206, 207]. When the eigenvalues of the kernel decay polynomially as $\mathcal{O}(m^{-p})$—see Definition 1—the results of [185] lead to possibly vacuous (infinite) sample complexities, while we prove an $\tilde{\mathcal{O}}((\frac{H^3}{\epsilon})^{2+\frac{2}{p-1}})$ sample complexity for the generative setting and $\tilde{\mathcal{O}}(H(\frac{H^3}{\epsilon})^{2+\frac{2}{p-1}})$ for the online setting. Our sample complexity results are comparable to those of [198]. In a technical comparison, their approach requires a specific assumption on the dependence between kernel eigenvalues and domain size (see Definition 4.1 in [198]), which we do not. Additionally, they employ a sophisticated domain partitioning algorithm that is more difficult to implement and possibly inefficient in practice, whereas our algorithm is simpler and more straightforward. In the case of Matérn kernel with smoothness parameter ν on a d-dimensional domain, where $p = 1 + \frac{2\nu}{d}$, our results translate to a sample complexity of $\tilde{\mathcal{O}}(H(\frac{H^3}{\epsilon})^{2+\frac{d}{\nu}})$, that matches the $\Omega((\frac{1}{\epsilon})^{2+\frac{d}{\nu}})$ lower bound proven in [65] for the degenerate case of bandits with $H = 1$. Our sample complexities thus are not generally improvable in their scaling with $\frac{1}{\epsilon}$.

To achieve these results, we establish a confidence interval applicable to kernel ridge regression in our RL setting that may be of broader interest. The key technical novelties of this confidence interval involves leveraging the structure of RKHS and the properties of unbiased, independent samples. The main results regarding the confidence interval and sample complexities of the two exploration algorithms, with and without the generative model, are presented in Theorems 1, 2 and 3, respectively, in Section 5.5. We empirically validate our analytical findings through numerical experiments comparing the performance of our proposed exploration algorithms with that of [185], as detailed in Section 5.6. Section 5.2 provides an overview of related work, Section 5.3 introduces episodic MDPs, the reward-free RL framework, and kernel-based models. Section 5.4 presents our algorithms for both the exploration and planning phases. Detailed proofs of theorems, along with the details of experiments and further experimental results, are included in Appendix B.

5.2 Related Work

In this section, we first present a more comprehensive literature review, including related works that were not covered in Section 5.1. We also provide a summary table of sample complexity results in the reward-free RL setting, highlighting our key contributions. Following this, we offer a technical comparison between our work and the two most relevant prior works, [185] and [198].

5.2.1 Literature Review

Numerous studies have addressed the sample complexity problem in the discounted MDP framework with an infinite horizon, where the agent has sam-

Table 5.1: Existing sample complexities in reward-free RL. $\mathcal{S}$, $\mathcal{A}$, H, d and p represent the state space, action space, episode length, state-action space dimension and parameter of the kernel with polynomial eigendecay, respectively. Last two rows correspond to the performance guarantees for the algorithms proposed in this work.

Setting	Sample complexity
Tabular [183]	$\mathcal{O}\left(\frac{\lvert\mathcal{S}\rvert^2\lvert\mathcal{A}\rvert H^5}{\epsilon^2}\right)$
Linear [184]	$\tilde{\mathcal{O}}\left(\frac{d^3H^6}{\epsilon^2}\right)$
Kernel-based (exponential eigendecay) [185]	$\mathcal{O}\left(\frac{H^6\text{polylog}(\frac{1}{\epsilon})}{\epsilon^2}\right)$
Kernel-based (polynomial eigendecay) [198]	$\tilde{\mathcal{O}}\left((\frac{H^3}{\epsilon})^{2+\frac{2}{p-1}}\right)$
Kernel-based (exponential eigendecay) (this work)	$\tilde{\mathcal{O}}\left(\frac{H^7\text{polylog}(\frac{1}{\epsilon})}{\epsilon^2}\right)$
Kernel-based (polynomial eigendecay) (this work)	$\tilde{\mathcal{O}}\left(H(\frac{H^3}{\epsilon})^{2+\frac{2}{p-1}}\right)$

pling access to a generative model, such as [208, 201, 204]. Alternatively, other research has focused on the episodic MDP framework, without reliance on a generative model or an exploratory policy. Both the tabular setting [17, 18, 19] and the linear setting [20, 21, 22, 23, 24] have been thoroughly examined. Recent literature has extended these techniques to the kernel setting [31, 209, 210, 211, 181], although further improvements are needed in achieving better regret bounds. In contrast to these prior works which assume that the reward function is provided, we explore the episodic reward-free setting in this work, both with and without a generative model. This setting is significantly different from standard RL, rendering the existing sample complexity results inapplicable to our context.

In the context of reward-free RL, numerous empirical studies have proposed various exploration methods from a practical perspective, as demonstrated by works such as [131, 135, 212]. Theoretically, researchers have explored the reward-free RL framework across different levels of complexity, ranging from tabular to linear, kernel-based, and deep learning-based models [183, 184, 185] (Table 5.1). Although the existing literature adequately covers the tabular and linear settings, it often provides only partial and incomplete findings when addressing the more intricate kernel-based and deep learning settings. The most relevant work in the kernel setting is [185], which provides a reward-free algorithm whose sample complexity is $\mathcal{O}\left(\frac{H^6\text{polylog}(\frac{1}{\epsilon})}{\epsilon^2}\right)$. Their results however are only applicable to very smooth kernels with exponentially decaying eigenvalues. The recent work of [198] proved a sample complexity of $\tilde{\mathcal{O}}\left((\frac{H^3}{\epsilon})^{2+\frac{2}{p-1}}\right)$ for kernels with polynomial eigendecay. However, they employ a niche domain partitioning technique that, despite its theoretical appeal, is cumbersome to implement and raises practical concerns, as mentioned earlier.

Finally, it's important to mention that the planning phase of our proposed algorithm is similar to the problem of learning a good policy from predefined datasets, typically called batch or offline RL [189]. Many prior works on offline RL make the coverage assumption on the dataset, requiring it to sufficiently include any possible state-action pairs with a minimum probability [186, 187, 191, 188]. These works do not address the exploration needed to achieve such good coverage, which is where our reward-free approach significantly differs. Our goal is to demonstrate how to collect sufficient exploration data without any reward information, enabling the design of a near-optimal policy for any reward function during the planning phase.

5.2.2 Comparison to Existing Works

Here, we discuss the key differences between our approach and the closely related works of [185] and [198]. In [185], they conduct exploration by accumulating standard deviation over an episode, then they apply a planning phase-like algorithm to maximize a reward proportional to $\beta(\delta)\sigma_{h,n}$ at each step of an episode. However, this approach can inflate the confidence interval width multiplier $\beta(\delta)$ by a factor of $\sqrt{\Gamma(n)}$, potentially leading to suboptimal or even trivial sample complexities when $\sqrt{\Gamma(n)}$ is large, as seen in [185]. Specifically, their results are applicable to very smooth kernels like SE, with exponentially decaying Mercer eigenvalues, for which $\Gamma(n) = \mathcal{O}(\text{polylog}(n))$. For kernels with polynomial eigendecay, where $\Gamma(n) = \mathcal{O}(n^{\frac{1}{p+1}})$ grows polynomially with n, this algorithm possibly leads to trivial (infinite) sample complexities. Intuitively, the inflation of $\beta(\delta)$ is due to the adaptive sampling creating statistical dependencies among observations, specifically through next state transitions. When such dependencies exist, the best existing confidence intervals are based on a kernel adaptation of self-normalized vector values martingales [213]. The $\sqrt{\Gamma(n)}$ term cannot be removed in general for adaptive samples that introduce bias, as was discussed in [32] and [214].

The work in [198] utilizes domain partitioning, relying on only a subset of samples to obtain confidence intervals. This approach achieves order-optimal sample complexity for kernels with polynomial eigendecay, offering an H-factor improvement compared to our work in the online setting. However, firstly, their results are limited by specific assumptions regarding the relationship between kernel eigenvalues and domain size, which reduces the generality of their findings. Secondly, their domain partitioning method is cumbersome to implement and lacks practical justification, as it requires dropping samples from other subdomains. In contrast, our algorithm achieves order-optimal results for general kernels with a simpler approach that leverages statistical independence. Moreover, our method is well-suited to the generative setting, where their approach offers no clear advantages.

5.3 Preliminaries and Problem Formulation

In this section, we introduce the episodic MDP setting, describe the reward-free RL framework, provide background on kernel methods, and outline our technical assumptions.

5.3.1 Episodic MDP

An episodic MDP can be described by the tuple $M = (\mathcal{S}, \mathcal{A}, H, P, r)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, and the integer H the length of each episode. Here, $r = \{r_h\}_{h=1}^{H}$ represents the reward functions, and $P =$

$\{P_h\}_{h=1}^H$ the transition probability distributions.[1] The state-action space is denoted by $\mathcal{Z} = \mathcal{S} \times \mathcal{A}$. The notation $z = (s,a)$ is used throughout the chapter for state-action pairs. For each step $h \in [H]$, the reward function $r_h : \mathcal{Z} \to [0,1]$ is supposed to be deterministic for simplicity, and $P_h(\cdot|s,a)$ is the unknown transition probability distribution on $\mathcal{S}$ for the next state given the current state-action pair (s,a). A policy $\pi = \{\pi_h : \mathcal{S} \to \mathcal{A}\}_{h=1}^H$ determines the action $\pi_h(s)$—possibly random—taken by the agent at state s during each step h. At the beginning of each episode, the environment picks an arbitrary initial state s_1. The agent adopts a policy $\pi = \{\pi_h\}_{h=1}^H$. For each step $h \in [H]$, the agent observes the current state $s_h \in \mathcal{S}$, and selects an action $a_h = \pi_h(s_h)$. The subsequent state, s_{h+1}, is then drawn from the transition probability distribution $P_h(\cdot|s_h, a_h)$. The episode ends when the agent receives the final reward $r_H(s_H, a_H)$.

We are interested in maximizing the expected total reward in the episode, starting at step h. This is quantified by the value function, which is defined as follows:

$$V_h^\pi(s) = \mathbb{E}\left[\sum_{h'=h}^{H} r_{h'}(s_{h'}, a_{h'}) \middle| s_h = s\right], \forall s \in \mathcal{S}, h \in [H], \tag{5.1}$$

where the expectation is taken with respect to the randomness in the trajectory $\{(s_h, a_h)\}_{h=1}^H$ obtained by the policy π. It can be shown that under mild assumptions (e.g., continuity of P_h, compactness of $\mathcal{Z}$, and boundedness of r), there exists an optimal policy $\pi^\star$ which attains the maximum possible value of $V_h^\pi(s)$ at every step and at every state (see, e.g., [215]). We use the notation $V_h^\star(s) = \max_\pi V_h^\pi(s)$, $\forall s \in \mathcal{S}, h \in [H]$. By definition $V_h^{\pi^\star} = V_h^\star$. An ϵ-optimal policy is defined as follows.

Definition 2. *(ϵ-optimal policy) For $\epsilon > 0$, a policy π is called ϵ-optimal if it achieves near-optimal values from any initial state as follows: $V_1^\pi(s) \geq V_1^\star(s) - \epsilon, \quad \forall s \in \mathcal{S}$.*

[1]We deliberately do not use the standard term transition kernel for P_h, to avoid confusion with kernel in kernel-based learning.

Policy design often relies on the expected value of a value function with respect to the transition probability distribution, presented using the following notation:

$$[P_h V](s,a) := \mathbb{E}_{s' \sim P_h(\cdot|s,a)}[V(s')]. \tag{5.2}$$

We also define the state-action value function $Q_h^\pi : \mathcal{Z} \to [0, H]$ as follows:

$$Q_h^\pi(s,a) = \mathbb{E}_\pi \left[\sum_{h'=h}^{H} r_{h'}(s_{h'}, a_{h'}) \middle| s_h = s, a_h = a \right], \tag{5.3}$$

where the expectation is taken with respect to the randomness in the trajectory $\{(s_h, a_h)\}_{h=1}^H$ obtained by the policy π. The Bellman equation associated with a policy π is then represented as

$$Q_h^\pi(s,a) = r_h(s,a) + [P_h V_{h+1}^\pi](s,a),$$
$$V_h^\pi(s) = \mathbb{E}[Q_h^\pi(s, \pi_h(s))], \quad V_{H+1}^\pi = \mathbf{0}.$$

The notation $V = \mathbf{0}$ is used for $V(s) = 0$, for all $s \in \mathcal{S}$. We may specify the reward function in $V^\pi, Q^\pi, V^\star, Q^\star$ notations for clarity, for example, $V^\pi(s; r)$ and $Q^\star(z; r)$.

5.3.2 Reward-Free RL Framework

We aim to learn ϵ-optimal policies while minimizing the samples collected during exploration. Specifically, we employ the reward-free RL framework, which consists of two phases: exploration and planning. In the exploration phase, we collect a dataset $\mathcal{D}_N = \{\mathcal{D}_{h,N}\}_{h \in [H]}$, where each $\mathcal{D}_{h,N} = \left\{ \left(s_{h,n}, a_{h,n}, s'_{h+1,n} \sim P_h(\cdot | s_{h,n}, a_{h,n}) \right) \right\}_{n \in [N]}$ consists of N transition samples at step h. Then, in the planning phase, once the reward r is revealed, we design a policy specific to reward r using the data collected during the exploration phase. The number N denotes the sample complexity required to design an ϵ-optimally performing policy. A critical question arises: *How many exploration episodes are necessary to achieve ϵ-optimal policies?* We provide an answer in this chapter.

5.3.3 Kernel Ridge Regression

As introduced in Section 2.6.2, kernel-based methods are effective for estimating the expected value function in RL. In this section, we present the kernel ridge regression formulation used in our setting, focusing on its role in deriving statistical predictions and confidence intervals.

Keeping the Bellman equation in mind, we derive statistical predictions and bounds for the expected value function $[PV]:\mathcal{Z}\to\mathbb{R}$, for some given value function $V:\mathcal{S}\to\mathbb{R}$ and conditional probability distribution $P(\cdot|z)$. Let us use the notation $f=[PV]$. Suppose that we are given n noisy observations of f, represented as $\{(z_i,y_i)\}_{i\in[n]}$, where $y_i=f(z_i)+\varepsilon_i$, and ε_i denotes zero-mean random noise. Provided a positive definite kernel $k:\mathcal{Z}\times\mathcal{Z}\to\mathbb{R}$ and employing kernel ridge regression, we can make the following prediction for f:

$$\hat{f}_n(z)=k_n^\top(z)(K_n+\tau^2I)^{-1}\boldsymbol{y}_n, \tag{5.4}$$

where $k_n(z)=[k(z,z_1),k(z,z_2),\cdots,k(z,z_n)]^\top$ is the pairwise kernel values between z and observation points, $K_n=[k(z_i,z_j)]_{i,j\in[n]}$ is the Gram matrix, $\boldsymbol{y}_n=[y_1,y_2,\cdots,y_n]^\top$ is the vector of observations, $\tau>0$ is a free parameter, and I is the identity matrix, appropriately sized to match the dimensions of K_n. In addition, the following uncertainty estimate can be utilized to bound the prediction error:

$$\sigma_n^2(z)=k(z,z)-k_n^\top(z)(K_n+\tau^2I)^{-1}k_n(z) \tag{5.5}$$

In particular, various $1-\delta$ confidence intervals of the form $|f(z)-\hat{f}_n(z)|\leq\beta(\delta)\sigma_n(z)$, under various assumptions, are proven, where $\beta(\delta)$ is a confidence interval width multiplier that depends on the setting and assumptions [213, 66, 70, 72]. One of our primary contributions is establishing novel confidence intervals for $f=[PV]$, applicable to our RL setting. Equipped with the confidence intervals, we are able to design policies using least squares value iteration or its *optimistic* variant.

Reproducing Kernel Hilbert Spaces (RKHS) and Mercer Representation. We briefly recall the definitions of RKHS and the Mercer theorem from Chapter 3, Section 3.3.2. Given a positive definite kernel k, the associated RKHS $\mathcal{H}_k$ consists of functions f that can be written as $f = \sum_{m=1}^{\infty} w_m \sqrt{\gamma_m} \varphi_m$, where $\{\gamma_m, \varphi_m\}$ are the Mercer eigenpairs of k. The RKHS norm is given by $\|f\|_{\mathcal{H}_k}^2 = \sum_{m=1}^{\infty} w_m^2$, that is, the ℓ^2-norm of the weight vector $\boldsymbol{w} = [w_1, w_2, \cdots]^\top$.

For formal statements and further details, please refer to Appendix B.5. To effectively use the confidence intervals established by the kernel-based models on f, we require the following assumption.

Assumption 1. *We assume $P_h(s|\cdot,\cdot) \in \mathcal{H}_k$, for some positive definite kernel k, and $\|P_h(s|\cdot,\cdot)\|_{\mathcal{H}_k} \leq 1$, for all $s \in \mathcal{S}$ and $h \in [H]$.*

Consequently, for all $V : \mathcal{S} \to [0, H]$, we have $\|[PV]\|_{\mathcal{H}_k} = \mathcal{O}(H)$. See [216], Lemma 3, for a proof.

Information Gain and Eigendecay. As introduced in Chapter 3 (see Section 3.3.5), the maximum information gain $\Gamma(n)$ quantifies the complexity of kernel-based RL and bandit problems [67, 68]:

$$\Gamma(n) = \sup_{\{z_i\}_{i=1}^n \subset \mathcal{Z}} \frac{1}{2} \log\det\left(I + \frac{K_n}{\tau^2}\right). \tag{5.6}$$

Its growth is governed by the decay rate of the eigenvalues of the kernel (Definition 1). In particular, as discussed in Chapter 3, $\Gamma(n) = \mathcal{O}(\text{polylog}(n))$ for kernels with exponential eigendecay (e.g., SE kernels), and $\Gamma(n) = \tilde{\mathcal{O}}(n^{1/p})$ for kernels with polynomial eigendecay (e.g., Matérn kernels), with important implications for regret bounds in RL and bandits [68].

5.4 Algorithm Description

We now present our algorithms for both the exploration and planning phases. We begin by presenting the algorithm for the planning phase, as it remains unchanged across various exploration algorithms.

5.4.1 Planning Phase

In the planning phase, the reward function r is revealed to the learner. In addition, a dataset $\mathcal{D}_N = \{\mathcal{D}_{h,N}\}_{h\in[H]}$ is available, with $\mathcal{D}_{h,N} = \{s_{h,n}, a_{h,n}, s'_{h+1,n} \sim P_h(\cdot|s_{h,n}, a_{h,n})\}_{n\in[N]}$ for each step $h \in [H]$. The objective is to leverage the knowledge of the reward function and utilize the dataset to design a near-optimal policy. As mentioned in the introduction, the planning phase comprises of an offline RL design without further interaction with the environment.

In the planning phase of our algorithm, we derive a policy using least squares value iteration. Specifically, at step h, we compute a prediction, $\hat{g}_h$, for the expected value function in the next step $[P_hV_{h+1}]$. We then define

$$Q_h(\cdot,\cdot) = \Pi_{[0,H]}\left[r_h(\cdot,\cdot) + \hat{g}_h(\cdot,\cdot) + \beta(\delta)\sigma_{h,N}(\cdot,\cdot)\right], \tag{5.7}$$

where $\Pi_{[a,b]}$ denotes projection on $[a,b]$ interval. The policy π is then obtained as a greedy policy with respect to Q. For each $h \in [H]$,

$$\pi_h(\cdot) = \arg\max_{a\in\mathcal{A}} Q_h(\cdot, a).$$

We now detail the computation of $\hat{g}_h$. Keeping the Bellman equation in mind and starting with $V_{H+1} = \mathbf{0}$, $\hat{g}_h$ is the kernel ridge predictor for $[P_hV_{h+1}]$. This prediction uses N observations

$$\boldsymbol{y}_h = [V_{h+1}(s'_{h+1,1}), V_{h+1}(s'_{h+1,2}), \cdots, V_{h+1}(s'_{h+1,N})]^\top$$

at points $\{z_{h,n}\}_{n=1}^N$. Recall that $\mathbb{E}_{s'\sim P(\cdot|z_{h,n})}[V_{h+1}(s')] = [P_hV_{h+1}](z_{h,n})$. The observation noise $V_{h+1}(s'_{h+1,n}) - [P_hV_{h+1}](z_{h,n})$ is due to random transitions and is bounded by $H - h \leq H$. Specifically,

$$\hat{g}_h(z) = k_{h,N}^\top(z)(\tau^2 I + K_{h,N})^{-1}\boldsymbol{y}_h, \tag{5.8}$$

where $k_{h,N}(z) = [k(z, z_{h,1}), k(z, z_{h,2}), \cdots, k(z, z_{h,N})]^\top$ is the pairwise kernel values between z and observation points and $K_{h,N} = [k(z_{h,i}, z_{h,j})]_{i,j\in[N]}$ is the

Gram matrix. Also, $\sigma_{h,N}$ in (5.7) is specified as follows:

$$\sigma^2_{h,N}(z) = k(z,z) - k^\top_{h,N}(z)(\tau^2 I + K_{h,N})^{-1} k_{h,N}(z). \tag{5.9}$$

We then define Q_h according to (5.7) and set

$$V_h(s) = \max_{a\in\mathcal{A}} Q_h(s,a).$$

The values of $\hat{g}_h$, $\sigma_{h,N}$, Q_h and V_h are obtained recursively for $h = H, H-1, \cdots, 1$. For a pseudocode, see Algorithm 7.

Algorithm 7 Planning Phase

```
Input: $\tau$, $\beta$, $\delta$, $k$, $M(\mathcal{S},\mathcal{A},H,P,r)$, and exploration dataset $\mathcal{D}_N$.
for $h = H, H-1, \cdots, 1$, do
    Compute the prediction $\hat{g}_h$ according to (5.8);
    Let $Q_h(\cdot,\cdot) = \Pi_{[0,H]}[\hat{g}_h(\cdot,\cdot) + r_h(\cdot,\cdot) + \beta(\delta)\sigma_{h,N}(.,.)]$;
    $V_h(\cdot) = \max_{a\in\mathcal{A}} Q_h(\cdot,a)$;
    $\pi_h(\cdot) = \arg\max_{a\in\mathcal{A}} Q_h(\cdot,a)$;
end for
Output: $\{\pi_h\}_{h\in[H]}$.
```

5.4.2 Exploration Phase

In the exploration phase, the algorithm collects a dataset $\mathcal{D}_N = \{\mathcal{D}_{h,N}\}_{h\in[H]}$, where $\mathcal{D}_{h,N} = \{s_{h,n}, a_{h,n}, s'_{h+1,n}\}_{h\in[H],n\in[N]}$ for each $h \in [H]$, later used in the planning phase to design a near-optimal policy. The primary goal during this phase is to gather the most informative observations.

Initially, we consider a preliminary case where a *generative model* [199] is present that can produce transitions for the state-actions selected by the algorithm. Under this setting, we demonstrate that a simple rule for data collection leads to improved and desirable sample complexities. Inspired by these results, we introduce a novel algorithm that completely relaxes the requirement for a generative model, at the price of increasing the number of exploration episodes by a factor of H. The key aspect of our algorithms is the *unbiasedness*–statistical independence of the collected samples, which means that the observation points do not depend on previous transitions.

Algorithm 8 Exploration Phase **with** Generative Model

```
Require: $\tau$, $k$, $\mathcal{S}$, $\mathcal{A}$, $H$, $P$, $N$;
1: Initialize $\mathcal{D}_{h,0} = \{\}$, for all $h \in [H]$;
2: for $n = 1, 2, \cdots, N$ do
3:     for $h = 1, 2, \cdots, H$ do
4:         Let $s_{h,n}, a_{h,n} = \arg\max_{s\in\mathcal{S}, a\in\mathcal{A}} \sigma_{h,n-1}(s,a)$;
5:         Observe $s'_{h+1,n} \sim P_h(\cdot|s_{h,n}, a_{h,n})$;
6:         Update $\mathcal{D}_{h,n} = \mathcal{D}_{h,n-1} \bigcup \{s_{h,n}, a_{h,n}, s'_{h+1,n}\}$.
7:     end for
8: end for
9: Output: $\mathcal{D}_N$.
```

Algorithm 9 Exploration Phase **without** Generative Model

```
Require: $\tau$, $k$, $\beta$, $\delta$, $\mathcal{S}$, $\mathcal{A}$, $H$, $P$, $N$;
  Initialize $\mathcal{D}_{h,0} = \{\}$, for all $h \in [H]$;
  for $n = 1, 2, \cdots, N$ do
      for $h_0 = 1, 2, \cdots, H$ do
          Initialize $V_{h_0+1,n} = \mathbf{0}$
          for $h = h_0, h_0 - 1, \cdots, 1$ do
              Obtain $\hat{f}_{h,(n,h_0)}$; $Q_{h,(n,h_0)}$, and $V_{h,(n,h_0)}(\cdot)$ according to (5.12) and
              (5.13), respectively.
          end for
          for $h = 1, 2, \cdots, h_0$ do
              Observe $s_{h,n}$; Take action $a_{h,n} = \arg\max_{a\in\mathcal{A}} Q_{h,n}(s_{h,n}, a)$;
          end for
          Update $\mathcal{D}_{h_0,n} = \mathcal{D}_{h_0,n-1} \bigcup \{s_{h_0,n}, a_{h_0,n}, s_{h_0+1,n}\}$
      end for
  end for
```

5.4.2.1 Exploration with a Generative Model

In this section, we outline the exploration phase when a generative model is present. At each step h of the current exploration episode, uncertainties derived from kernel ridge regression are employed to guide exploration. Specifically, let

$$\sigma^2_{h,n}(z) = k(z,z) - k^\top_{h,n}(z)(\tau^2 I + K_{h,n})^{-1} k_{h,n}(z) \tag{5.10}$$

where $k_{h,n}(z) = [k(z, z_{h,1}), k(z, z_{h,2}), \cdots, k(z, z_{h,n})]^\top$ is the vector of kernel values between the state-action of interest and past observations in $\mathcal{D}_{h,n}$, and

$K_{h,n} = [k(z_{h,i}, z_{h,j})]_{i,j=1}^{n}$ is the Gram matrix of pairwise kernel values between past observations in $\mathcal{D}_{h,n}$. Equipped with $\sigma_{h,n}(z)$, at step h, we select

$$s_{h,n}, a_{h,n} = \underset{s \in \mathcal{S}, a \in \mathcal{A}}{\arg\max}\, \sigma_{h,n-1}(s,a), \tag{5.11}$$

and observe the next state $s'_{h+1,n} \sim P_h(\cdot | s_{h,n}, a_{h,n})$. We then add this data point to the dataset and update $\mathcal{D}_{h,n} = \mathcal{D}_{h,n-1} \cup \{(s_{h,n}, a_{h,n}, s'_{h+1,n})\}$. For a pseudocode, see Algorithm 8.

We highlight that the selection rule (5.11) relies on the generative model that allows the algorithm to deviate from the Markovian trajectory and move to a state of its choice. Since observations $(s_{h,n}, a_{h,n})$ are selected based on maximizing $\sigma_{h,n-1}$, which by definition (5.10) does not depend on previous transitions $\{s'_{h+1,i}\}_{i=1}^{n-1}$, the statistical independence conveniently holds. The generative model setting is feasible in contexts such as games, where the player can manually set the current state. However, this may not always be possible in other scenarios. Next, we introduce our online algorithm, which strictly stays on the Markovian trajectory.

5.4.2.2 Exploration without Generative Models

In this section, we show that a straightforward algorithm, in contrast to existing approaches, achieves near-optimal performance in an online setting without requiring a generative model. Compared to the scenario with a generative model, the sample complexity of this algorithm increases by a factor of H. For a detailed and technical comparison with existing work, please refer to Section 5.2.2.

Our online algorithm operates as follows: in each exploration episode, only one data point specific to a step h is collected—this accounts for the H scaling in sample complexity. This observation however is collected in an unbiased way, which eventually leads to tighter performance guarantees. Specifically, at episode $nH + h_0$, where $n \in [N]$ and $h_0 \in [H]$, the algorithm collects an informative sample for the transition at step h_0. This results in a total of N

samples at each step over NH episodes. The algorithm initializes $V_{h_0+1,(n,h_0)} = \mathbf{0}$. Let $\hat{f}_{h,(n,h_0)}$ and $\sigma_{h,n}$ represent the predictor and uncertainty estimator for $[P_h V_{h+1,(n,h_0)}]$, respectively. These are derived from the historical data $\mathcal{D}_{h,n-1}$ of observations at step h. Specifically,

$$\hat{f}_{h,(n,h_0)}(z) = k_{h,n}^{\top}(z)(K_{h,n}+\tau^2 I)^{-1}\boldsymbol{y}_{h,n},$$
$$\sigma_{h,n}^2(z) = k(z,z) - k_{h,n}^{\top}(z)(K_{h,n}+\tau^2 I)^{-1}k_{h,n}(z), \tag{5.12}$$

where $k_{h,n}(z) = [k(z,z_{h,1}), k(z,z_{h,2}), \cdots, k(z,z_{h,n})]^{\top}$ is the vector of kernel values between the state-action of interest and past observations in $\mathcal{D}_{h,n}$, $K_{h,n} = [k(z_{h,i}, z_{h,j})]_{i,j=1}^{n}$ is the Gram matrix of pairwise kernel values between past observations in $\mathcal{D}_{h,n}$, and

$$\boldsymbol{y}_{h,(n,h_0)} = [V_{h+1,(n,h_0)}(s_{h+1,1}), V_{h+1,(n,h_0)}(s_{h+1,2}), \quad \cdots, V_{h+1,(n,h_0)}(s_{h+1,n})]^{\top}$$

is the vector of observations. We then have

$$Q_{h,(n,h_0)} = \Pi_{0,H}\left[\hat{f}_{h,(n,h_0)} + \beta(\delta)\sigma_{h,n}\right],$$
$$V_{h,(n,h_0)}(\cdot) = \max_{a\in\mathcal{A}} Q_{h,(n,h_0)}(\cdot,a). \tag{5.13}$$

The values of $Q_{h,(n,h_0)}$ and $V_{h,(n,h_0)}$ are obtained recursively for all $h \in [h_0]$. The exploration policy at episode $nH + h_0$ is then the greedy policy with respect to $Q_{h,(n-1,h_0)}$. The dataset is updated by adding the new observation to the dataset for step h_0, such that $\mathcal{D}_{h_0,n} = \mathcal{D}_{h_0,n-1} \cup \{(s_{h_0,n}, a_{h_0,n}, s_{h_0+1,n})\}$, while datasets for all other steps remain unchanged: $\mathcal{D}_{h,n} = \mathcal{D}_{h,n-1}$ for all $h \neq h_0$. This specific update ensures that the collected samples are unbiased. More specifically, the sample collected at h_0 solely relies on the uncertainty $\sigma_{h_0,n}$, due to the initialization $V_{h_0+1,(n,h_0)} = \mathbf{0}$ which implies $\hat{f}_{h_0,(n,h_0)} = \mathbf{0}$. Since $\sigma_{h_0,n}$ does not depend on previous transitions $s_{(h_0+1,i)}$ for any $i \leq n$, the samples at $h = h_0$ are unbiased. However, for $h < h_0$, the samples depend on both the uncertainty $\sigma_{h,n}$ and the prediction $\hat{f}_{h,(n,h_0)}$ (5.12). Since the prediction depends on the transitions $s_{(h+1,i)}$ for $i \leq n$, these samples are biased. As a result, we discard them and only retain the unbiased samples at $h = h_0$. This approach improves the rates in our analysis, albeit at the cost of a factor of H. For a pseudocode, see Algorithm 9, and for a diagram sketch, see Figure 5.1.

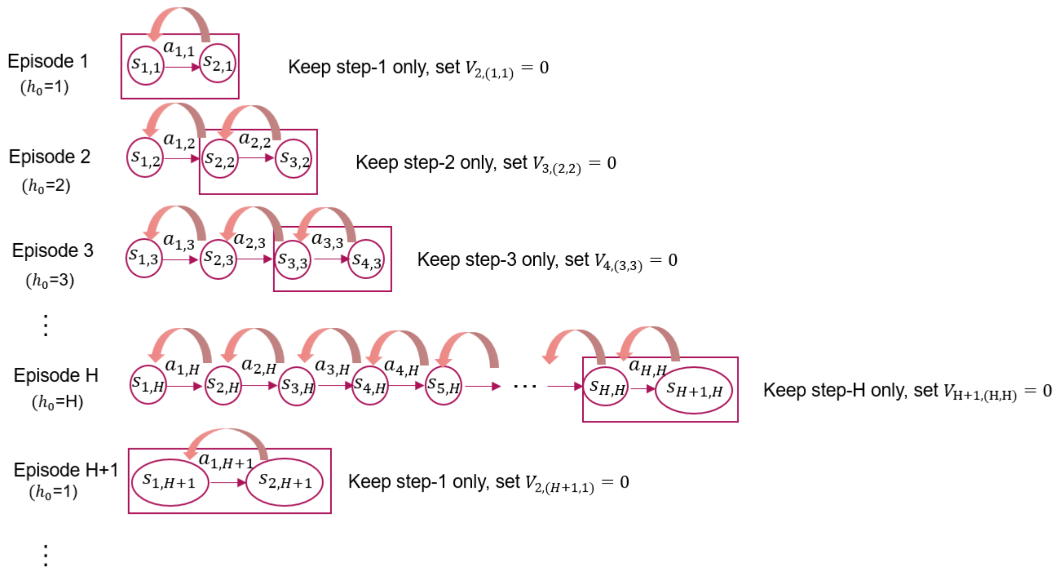

Episode 1 (h_0=1)
Keep step-1 only, set $V_{2,(1,1)} = 0$
Episode 2 (h_0=2)
Keep step-2 only, set $V_{3,(2,2)} = 0$
Episode 3 (h_0=3)
Keep step-3 only, set $V_{4,(3,3)} = 0$
Episode H (h_0=H)
Keep step-H only, set $V_{H+1,(H,H)} = 0$
Episode H+1 (h_0=1)
Keep step-1 only, set $V_{2,(H+1,1)} = 0$

Figure 5.1: High-level illustration of sample collection in the exploration algorithm without a generative model (Algorithm 9). At each episode, only one unbiased sample corresponding to step h_0 (shown inside the rectangle) is collected. The backward arrows indicate the recursive computation of the value functions.

5.4.3 Computational Complexity

The main computational bottleneck is the matrix inversion in kernel ridge regression, which incurs a cost of $\mathcal{O}(n^3)$, leading to a total complexity of $\mathcal{O}(N^4)$ for our algorithms. This is comparable to the complexities in related work, such as [181] and [185]. Notably, the $\mathcal{O}(n^3)$ cost of matrix inversion is not unique to RL but is common across kernel-based supervised learning and bandit literature. Sparse approximation methods, such as Sparse Variational Gaussian Processes (SVGP) and the Nyström method, can significantly reduce this complexity (in some cases, to linear time) while preserving kernel-based confidence intervals and corresponding rates (e.g., [217]). However, since these methods are broadly applicable rather than specific to our setting, we chose to maintain a clear, notation-light presentation focused on our main contributions.

5.5 Analysis of the Sample Complexity

In this section, we present our main results on the sample complexity of the algorithms. We first establish a novel confidence interval that is applicable to the unbiased samples collected by our exploration algorithms. We then provide theorems detailing the performance of these algorithms.

5.5.1 Confidence Intervals

We introduce a novel confidence interval that is tighter than existing ones in our RL setting and can also be applied to other RL problems such as offline RL and infinite-horizon settings.

Theorem 1 (Confidence Bounds). *Consider compact sets $\mathcal{S} \subset \mathbb{R}^{d_s}, \mathcal{A} \subset \mathbb{R}^{d_a}$, and define $\mathcal{Z} = \mathcal{S} \times \mathcal{A}$, $d = d_a + d_s$. Consider two Mercer kernels $k_\varphi : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ and $k_\psi : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$. Assume that functions $f : \mathcal{Z} \to \mathbb{R}$ and $V : \mathcal{S} \to \mathbb{R}$, and for each $z \in \mathcal{Z}$, a conditional probability distribution $P(\cdot|z)$ over $\mathcal{S}$, are given such that $f(z) = \mathbb{E}_{s\sim P(\cdot|z)}[V(s)]$, $\|f\|_{\mathcal{H}_{k_\varphi}} \le B_1$, $\|V\|_{\mathcal{H}_{k_\psi}} \le B_2$, and $\max_{s\in\mathcal{S}} V(s) \le v_{\max}$, for some $B_1, B_2, v_{\max} > 0$. Assume a dataset of $\{z_i, s'_i\}_{i=1}^n$ is provided, where each z_i is independent of the set $\{s'_j\}_{j=1}^n$, and $s'_i \sim P(\cdot|z_i)$. Let $\hat{f}^n$ and σ^n be the kernel ridge predictor and uncertainty estimator of f using the observations:*

$$\hat{f}_n(z) = k_{\varphi_n}^\top(z)(\tau^2 I + K_{\varphi_n})^{-1}\boldsymbol{y}_n,$$
$$\sigma_n^2(z) = k_\varphi(z,z) - k_{\varphi_n}^\top(z)(\tau^2 I + K_{\varphi_n})^{-1}k_{\varphi_n}(z), \tag{5.14}$$

where $\boldsymbol{y}_n = [V(s'_1), V(s'_2), \cdots, V(s'_n))]^\top$. In addition, let λ_m, $m = 1, 2, \cdots$ represent the Mercer eigenvalues of k_ψ in a decreasing order, and ψ_m the corresponding Mercer eigenfunctions. Assume $\psi_m \le \psi_{\max}$ for some $\psi_{\max} > 0$. Fix $M \in \mathbb{N}$, and let C be a constant such that $C \ge \sum_{m=1}^M \lambda_m$.

Then, for a fixed $z \in \mathcal{Z}$, and for all V, with $\|V\|_{\mathcal{H}_{k_\psi}} \le B_2$, each of the following *holds with probability at least* $1-\delta$*:*

$$|f(z)-\hat{f}_n(z)| \leq \beta(\delta)\sigma_n(z)$$

with $\beta(\delta) =$

$$B_1 + \frac{CB_2\psi_{\max}}{\tau}\sqrt{2\log(\frac{M}{\delta})} + \frac{2B_2\psi_{\max}}{\tau}\sqrt{n\sum_{m=M+1}^{\infty}\lambda_m}\ .$$

Theorem 1 provides a confidence bound for kernel ridge regression that is applicable to our RL setting, and is a key result in deriving our sample complexities.

Proof sketch. To derive our confidence bounds, we use the Mercer representation of V and decompose the prediction error $f(z)-\hat{f}_n(z)$ into error terms corresponding to each Mercer eigenfunction ψ_m. We then divide these terms into two groups: the first M elements, corresponding to eigenfunctions with the largest eigenvalues, and the remainder. For the top M eigenfunctions, we establish high-probability bounds using standard kernel-based confidence intervals from [70]. The remaining terms are bounded based on eigendecay, and we sum over all m to obtain $\beta(\delta)$.

Remark 1. *Under some mild conditions, for example, the polynomial eigendecay given in Definition 1, the following expression can be derived for* β*:*

$$\beta(\delta) = \mathcal{O}\left(B_1 + \frac{B_2\psi_{\max}}{\tau}\sqrt{\log\left(\frac{n}{\delta}\right)}\right). \tag{5.15}$$

With polynomial eigendecay, the remark follows from setting M to $\lceil n^{\frac{1}{p-1}}\rceil$ in the expression of β in Theorem 1.

The confidence interval presented in Theorem 1 is applicable to a fixed $z \in \mathcal{Z}$. Over a discrete domain this can be easily extended to all $z \in \mathcal{Z}$ using a probability union bound and replacing δ with $\frac{\delta}{|\mathcal{Z}|}$ in the expression of $\beta(\delta)$. Using standard discretization techniques, we can also prove a variation of the confidence interval that holds true uniformly over continuous domains. In particular, under the following assumption, we present a variation of the theorem over continuous domains.

Assumption 2. *For each $n \in \mathbb{N}$, there exists a discretization $\mathbb{Z}$ of $\mathcal{Z}$ such that, for any $f \in \mathcal{H}_k$ with $\|f\|_{\mathcal{H}_k} \leq B_1$, we have $f(z) - f([z]) \leq \frac{1}{n}$, where $[z] = \text{argmin}_{z' \in \mathbb{Z}} ||z' - z||_{l^2}$ is the closest point in $\mathbb{Z}$ to z, and $|\mathbb{Z}| \leq cB_1^d n^d$, where c is a constant independent of n and B_1.*

Assumption 2 is a mild technical assumption that holds for typical kernels [67, 66, 70].

Corollary 1. *Under the setting of Theorem 1, and under Assumption 2, the following inequalities each hold uniformly in $z \in \mathcal{Z}$ and $V : \|V\|_{\mathcal{H}_{k_\psi}} \leq B_2$, with probability at least $1 - \delta$*

$$
\begin{aligned}
f(z) &\leq \hat{f}_n(z) + \frac{2}{n} + \tilde{\beta}(\delta)(\sigma_n(z) + \frac{2}{\sqrt{n}}), \\
f(z) &\geq \hat{f}_n(z) - \frac{2}{n} - \tilde{\beta}(\delta)(\sigma_n(z) + \frac{2}{\sqrt{n}}),
\end{aligned}
$$

with $\tilde{\beta}(\delta) = \beta(\frac{\delta}{2c_n})$, $c_n = c(u_n(\frac{\delta}{2}))^d n^d$, and $u_n(\delta) = \mathcal{O}(\sqrt{n + \log(\frac{1}{\delta})})$.

Remark 2. *Under some mild conditions, for example, the polynomial eigen-decay given in Definition 1, the following expression can be derived for $\tilde{\beta}$:*

$$
\tilde{\beta}(\delta) = \mathcal{O}\left(B_1 + \frac{CB_2\psi_{\max}}{\tau}\sqrt{d\log\left(\frac{n}{\delta}\right)}\right). \tag{5.16}
$$

5.5.2 Sample Complexities

We have the following theorem on the performance of Algorithm 8. The weakest assumption one can pose on the value functions is realizability, which asserts that the optimal value functions $V_h^\star$ for $h \in [H]$ belong to the RKHS H_{k_ψ} for some kernel $k_\psi : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$, or at least can be well-approximated by H_{k_ψ}. For stateless MDPs or multi-armed bandits (MAB) where $H = 1$, realizability alone is enough to guarantee provably efficient algorithms [67, 66, 70]. However, when $H > 1$, this assumption appears insufficient [218, 219], and stronger assumptions are typically made in these settings [20, 220, 182]. Following these works, our main assumption is a closure property for all value functions in the following class:

$$\mathcal{V} = \Big\{ s \mapsto \min \Big\{ H, \max_{a\in\mathcal{A}} \Big\{ r(s,a) + \varphi^\top(s,a)\boldsymbol{w} + \beta\sqrt{\varphi^\top(s,a)\Sigma^{-1}\varphi(s,a)} \Big\}\Big\}\Big\}, \tag{5.17}$$

where $0 < \beta < \infty$, $\|\boldsymbol{w}\| \leq \infty$, and Σ is an $\infty \times \infty$ matrix with $\Sigma \succ \tau^2 I$.

Assumption 3 (Optimistic Closure). *For any $V \in \mathcal{V}$, for some positive constant c_v, we have $\|V\|_{H_{k_\psi}} \leqslant c_v$.*

This is the same assumption as Assumption 1 in [182] and can be relaxed to value functions ϵ away from this class as described in Section 4.3 of [182]. The assumption ensures that the proxy value functions $(V_{h,n})$ lie within the RKHS of a suitable kernel k_ψ. Notably, the RKHS of widely used kernels, such as Matérn and NT kernels, can uniformly approximate any continuous function over compact subsets of $\mathbb{R}^d$ [67]. We have the following theorem on the sample complexity of the exploration algorithm with a generative model.

Theorem 2. *Consider the reward-free RL framework described in Section 5.3. Assume the existence of a generative model in the exploration phase that allows the algorithm to select state-action pairs of its choice at each step. Let N_0 be the smallest integer satisfying*

$$2H\beta(\delta)\sqrt{\frac{2\Gamma(N_0)}{N_0\log(1+1/\tau^2)}} + \frac{4\beta(\delta)H}{\sqrt{N_0}} + \frac{4H}{N_0} \leq \epsilon,$$

with $\beta(\delta) = \mathcal{O}(\frac{H}{\tau}\sqrt{d\log(\frac{NH}{\delta})})$ with a sufficiently large constant. Run Algorithm 8 for $N \geq N_0$ episodes to obtain the dataset $\mathcal{D}_N$. Then, use the obtained samples to design a policy π using Algorithm 7 with $\beta(\delta) = \mathcal{O}(\frac{H}{\tau}\sqrt{d\log(\frac{NH}{\delta})})$ with a sufficiently large constant. Then, under Assumptions 1, 2 and 3, with probability at least $1-\delta$, π is guaranteed to be an ϵ-optimal policy.

The following theorem presents the sample complexity for exploration without generative models.

Theorem 3. *Consider the reward free RL framework described in Section 5.3. Let N_0 be the smallest integer satisfying*

$$3H(H+1)\beta(\delta)\sqrt{\frac{2\Gamma(N_0)}{N_0\log(1+1/\tau^2)}}+\frac{8\beta(\delta)H(H+1)}{\sqrt{N_0}}$$
$$+\frac{4H(H+1)(\log(N_0)+1)}{N_0}+2H\sqrt{N_0(H+1)\log\left(\frac{2}{\delta}\right)}\leq\epsilon \tag{5.18}$$

with $\beta(\delta)=\mathcal{O}(\frac{H}{\tau}\sqrt{d\log(\frac{NH}{\delta})})$ with a sufficiently large constant. Run Algorithm 9 for $NH\geq N_0H$ episodes to obtain the dataset $\mathcal{D}_N$. Then, use the obtained samples to design a policy π using Algorithm 7. Then, under Assumptions 1, 2 and 3, with probability at least $1-\delta$, π is guaranteed to be an ϵ-optimal policy.

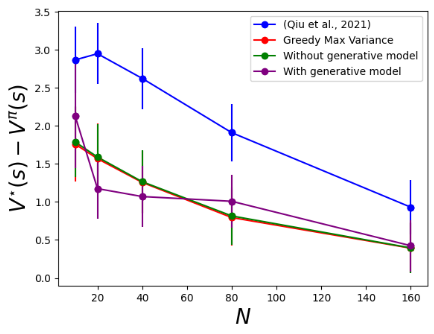
The proof of theorems are provided in Appendices B.2 and B.3.

The expression of suboptimality gap after N samples, given in (5.18), can be simplified as

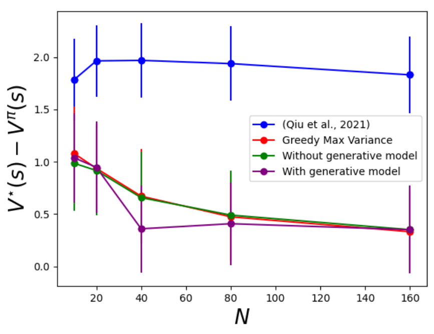
$$\mathcal{O}\left(H^3\sqrt{\frac{\Gamma(N)\log(NH/\delta)}{N}}\right).$$

Remark 3. *Replacing $\Gamma(N)=\tilde{\mathcal{O}}(N^{\frac{1}{p}})$ in the case of kernels with polynomial eigendecay, we obtain a sample complexity of $N=\tilde{\mathcal{O}}((\frac{H^3}{\epsilon})^{2+\frac{2}{p-1}})$. We also recall that without a generative model, we interact with H times more episodes to collect these samples. Specifically, the number of episodes in the exploration phase is $NH=\tilde{\mathcal{O}}\left(H(\frac{H^3}{\epsilon})^{2+\frac{2}{p-1}}\right)$.*

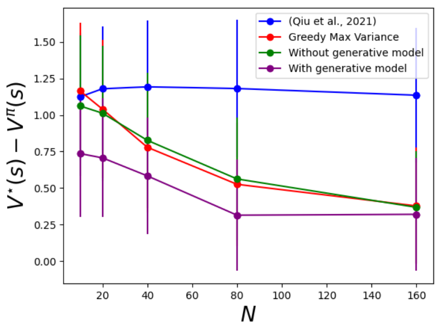
When specialized for the case of Matérn kernels with $p=1+\frac{2\nu}{d}$, we obtain $NH=\tilde{\mathcal{O}}(H(\frac{H^3}{\epsilon})^{2+\frac{d}{\nu}})$ that matches the lower bound for the degenerate case of bandits with $H=1$ proven in [65]. Our sample complexity is thus order optimal in terms of ϵ dependency. We also recall that the existing results lead to possibly vacuous (infinite) sample complexities for these kernels.



(a) Squared Exponential Kernel



(b) Matérn Kernel with $\nu = 2.5$



(c) Matérn kernel with $\nu = 1.5$

Figure 5.2: Average suboptimality gap against N. The error bars indicate standard deviation.

5.6 Experiments

We numerically validate our proposed algorithms and compare with the baseline algorithms. From the literature, we implement [185], in which the exploration aims at maximizing a hypothetical reward of $\beta\sigma_n/H$ over each episode n. The planning phase is similar to Algorithm 7. We also implement our exploration algorithms with and without a generative model: Algorithms 8 and 9 respectively. Additionally, we implement a heuristic variation of Algorithm 9, which collects the exploration samples in a greedy manner $a_{h,n} = \arg\max_{a\in\mathcal{A}} \sigma_{h,n}(s_{h,n}, a)$ while remaining on the Markovian trajectory by sampling $s_{h+1} \sim P_h(\cdot|s_h, a_h)$. We refer to this heuristic as *Greedy Max Variance*. For all these algorithms, we use Algorithm 7 to obtain a planning policy. In the experimental setting, we choose $H = 10$ and $\mathcal{S} = \mathcal{A} = [0,1]$ consisting of 100 evenly spaced points. We choose r and P from the RKHS of a fixed kernel. For the detailed framework and hyperparameters, please refer to Appendix B.4. We run the experiment for three different kernels across all 4 algorithms for 80 independent runs, and plot the average suboptimality gap $V_1^\star(s) - V_1^\pi(s)$ for $N = 10, 20, 40, 80, 160$, as shown in Figure 5.2. Our proposed Algorithm 9, without generative model, demonstrates better performance compared to prior work [185] across all three kernels, validating the improved sample efficiency. Notably, [185] performs poorly with nonsmooth kernels. Greedy Max Variance is a heuristic that in many of our experiments performs close to Algorithm 9. Furthermore, with access to a generative model, Algorithm 8 performs the best. This is anticipated, as the generative model provides the flexibility to select the most informative state-action pairs, unconstrained by Markovian transitions.

5.7 Conclusion

In this chapter, we proposed novel algorithms for the kernel-based reward-free RL problem, both with and without generative models, designed to efficiently gather informative data that facilitates near-optimal policy planning once the reward function becomes available. We demonstrated that, with a generative model, a simple algorithm can achieve near-optimal sample complexities. Without the generative model requirement, we showed that an online algorithm requires a sample complexity greater by a factor of H, implying that online sampling is H times more costly. Our results apply to a general class of kernels, including those with polynomial eigendecay, where existing methods may either lead to vacuous sample complexities [185] or require additional assumptions and a sophisticated, difficult-to-implement domain partitioning method [198]. Our experimental results support these analytical findings.

When compared to the lower bounds established in the degenerate case of bandits with $H = 1$ for the Matérn kernel, the order optimality of our results with respect to ϵ becomes evident. Nonetheless, an important limitation of our analysis is the additional scaling with H that arises as the price of online samples, in contrast to the case where a generative model is present.

For our experiments, we considered very general environments by arbitrarily selecting the reward and the transition probability distribution from the RKHS of a kernel. As a limitation, we did not provide experiments on RL benchmarks. This is a deliberate choice that allowed us to focus on the theoretical framework and, for example, experiment with various kernels with different levels of smoothness. This approach facilitates a finer comparison among algorithms. Investigating the performance of our methods on widely used RL benchmarks constitutes an interesting and valuable direction for future research.

Chapter 6

Bayesian Optimization from Human Feedback

In the previous chapter, we studied exploration in RL from a theoretical perspective, contributing to the development of sample-efficient exploration algorithms with convergence guarantees. Continuing with exploration as the central theme, we now shift to settings where scalar reward signals are absent and the agent instead receives feedback in the form of preferences between outcomes. This perspective extends the study of exploration to scenarios where learning must be driven by relative and limited feedback. Such preference-based exploration is increasingly relevant in practical applications, most notably in the alignment of LLMs, where specifying a reward function is difficult, but preferences can provide powerful learning signals.

To study this problem formally, we adopt the framework of Bayesian Optimization with preference-based feedback—referred to as Bayesian Optimization from Human Feedback (BOHF). Unlike conventional BO, where the learner observes scalar-valued outcomes, BOHF relies solely on the preference between two candidate actions. The objective is to identify the best action using a limited number of preference queries, which are often costly. Existing work, which adopts the Bradley-Terry-Luce (BTL) feedback model, provides regret bounds for the performance of several algorithms. In this chapter, within the same framework, we develop tighter performance guarantees.

Specifically, we derive regret bounds of $\tilde{\mathcal{O}}(\sqrt{\Gamma(T)T})$, where $\Gamma(T)$ represents the maximum information gain and T is the number of queries. Our results significantly improve upon existing bounds and, importantly, recover the order-optimal sample complexities of conventional BO with scalar feedback. In other words, preference-based learning can achieve the same order-optimal sample complexity as reward-based learning, despite operating with a more restrictive feedback model.

6.1 Introduction

Optimizing a black-box function using only preference-based feedback between pairs of candidate solutions has recently emerged as an interesting problem. This approach finds application, for instance, in prompt optimization [221], which aims to efficiently identify the best prompt for black-box LLMs, thereby significantly enhancing their performance [222, 221, 223]. Obtaining a numeric score to evaluate each prompt's performance is often unrealistic, but human users are generally much more reliable at providing preference feedback between pairs of prompts [221]. Since human feedback is costly, it becomes essential to develop efficient methods that can sequentially select favorable pairs of actions while minimizing the number of feedback instances required.

The theoretical framework for learning from preference-based feedback (see, e.g., [116, 117]) can be modeled as Bayesian Optimization from Human Feedback (BOHF). Similarly to conventional BO [224, 225, 67], the learner leverages previously collected samples through kernel-based regression to learn an unknown black-box function. However, unlike conventional BO methods that rely on direct evaluations of the target function, this approach collects pairwise comparisons instead of direct evaluation feedback, adding further complexities to the problem.

In the BOHF framework, at each time step $t = 1, 2, \cdots, T$, the learner selects a pair of actions (x_t, x'_t) and receives binary feedback $y_t \in \{0, 1\}$ representing the preference between the two actions. This binary feedback is modeled as a Bernoulli random variable, where the parameter is determined by applying a link function (here, sigmoid) to the difference in the unobserved utilities corresponding to each action, quantifying the preference between them. Performance is measured in terms of regret, defined as the cumulative loss in the selected pairs of actions compared to the optimal action (details are provided in Section 6.2). Kernel-based models employed within the BOHF framework allow for powerful and versatile modeling of preferences among actions, leveraging structures, and handling continuous domains or very large action spaces.

Existing work establishes a regret bound of $\tilde{\mathcal{O}}\left(\Gamma(T)\kappa^2\sqrt{T}\right)$ for the BOHF problem [116]. In this expression, κ is the maximum of the derivative of the inverse link function (see Equation (6.2)) and $\Gamma(T)$ is the maximum information gain, a kernel-specific and algorithm-independent complexity term (see Equation (6.11) for a slightly different notation compared to previous chapters).

It is insightful to compare the existing BOHF regret bound with the order-optimal regret bounds of $\tilde{\mathcal{O}}\left(\sqrt{\Gamma(T)T}\right)$ in conventional BO. In comparison, an additional κ^2 factor arises due to the feedback model. While this constant is independent of T, it can be very large. There is also an extra $\sqrt{\Gamma(T)}$ factor, which introduces potential challenges. As discussed in Section 3.3.5, the information gain $\Gamma(T)$ is polylogarithmic in T for smooth kernels like the Squared Exponential (SE), but grows polynomially for more general kernels such as the Matérn family [69] and Neural Tangent kernels (NTK) [226]. In such cases, the regret can grow linearly with T, making the bound potentially vacuous.

Our contribution is that we establish regret bounds of $\tilde{\mathcal{O}}\left(\sqrt{\Gamma(T)T}\right)$ for the BOHF problem (Theorem 4), achieving a $\sqrt{\Gamma(T)}$ improvement and eliminating the dependency on κ, resolving both issues and matching the regret bounds of conventional BO. From our regret bounds, we derive the sample complexities—the number of preference query samples required to identify near-optimal actions. Our sample complexities match the lower bounds obtained in [65] for conventional BO, which benefits from a richer feedback model with a different noise distribution. We will provide a technical discussion on this in Section 6.4.

In summary, we establish the intriguing result that the number of preferential feedback samples required to identify near-optimal actions is of the same order as the number of scalar-valued feedback samples. This is in sharp contrast and a significant improvement over the existing work [116, 117]. To obtain the improved regret bounds, we propose an algorithm referred to as Multi-Round Learning from Preference-based Feedback (MR-LPF). The proposed algorithm proceeds in rounds. In each round, pairs of actions are sequentially selected based on the highest uncertainty in their preference. This method effectively reduces uncertainties about the preferences between actions by the end of each round. The uncertainties are represented by kernel-based standard deviations. At the end of each round, the kernel-based confidence intervals are used to eliminate actions unlikely to be the best. Our multi-round structure is inspired by the BPE algorithm of [84], though the details and analysis differ significantly due to the preference-based feedback model. Details are provided in Section 6.3. We show that this structure allows for a more efficient use of kernel-based confidence intervals, contributing to improvements in both $\Gamma(T)$ and κ.

We present experimental results on the performance of MR-LPF on synthetic functions that closely align with the analytical assumptions, as well as on a dataset of Yelp reviews, demonstrating the utility of the proposed algorithm in real-world applications (Section 6.5).

6.1.1 Related Work

Two works closely related to ours are [116] and [117], which consider the exact same BOHF framework. The work by [116] proposed the MaxMinLCB algorithm, which takes a game-theoretic approach to selecting the pair of actions (x_t, x'_t) at each time step t. Specifically, x_t and x'_t are selected according to a game, with the objective function defined as a lower confidence bound (LCB) on the probability of favoring x_t over x'_t. Hence, the name: x_t is chosen to

*Max*imize and x'_t to *Min*imize the *LCB* (see [116], Algorithm 1). Their regret bound scales as $\tilde{\mathcal{O}}\left(\Gamma(T)\kappa^2\sqrt{T}\right)$, which may be vacuous for some commonly used kernels and scales with κ^2, which can be a large constant.

Another closely related work is [117], which develops Principled Optimistic Preferential Bayesian Optimization (POP-BO), an algorithm based on the optimism principle. Specifically, at each time step t, x'_t is set to x_{t-1}, one of the actions from the previous time step, and x_t is set to the maximizer of an upper confidence bound (UCB) on the preference between the two actions (see [117], Algorithm 1). They establish a regret bound of $\tilde{\mathcal{O}}\left((\Gamma(T)T)^{3/4}\right)$, which is larger than the one in [116] by a factor of $(T/\Gamma(T))^{1/4}$ and similarly may be vacuous for many cases of interest.[1] Their definition of regret is based directly on the utility function and slightly differs from ours. However, it remains equivalent to our regret definition up to a constant factor, as discussed in [116].

Table 6.1: Comparison of regret bounds in BOHF.

[116]	[117]	**This work**
$\tilde{\mathcal{O}}\left(\Gamma(T)\kappa^2\sqrt{T}\right)$	$\tilde{\mathcal{O}}\left((\Gamma(T)T)^{3/4}\right)$	$\tilde{\mathcal{O}}\left(\sqrt{\Gamma(T)T}\right)$

Some other preferential BO methods mainly propose heuristics without formal theoretical guarantees on regret or convergence proofs [227, 228, 229].

6.1.1.1 Conventional BO

Theoretical aspects of classical BO algorithms have been reviewed in Section 3.3.7. Notably, methods such as GP-UCB [67] and GP-TS [66] achieve cumulative regret bounds of $\mathcal{O}(\Gamma(T)\sqrt{T})$.

To improve upon this rate, several algorithmic refinements have been proposed. For example, SupKernelUCB [80], GP-ThreDS [83], and Batched Pure Exploration (BPE) [84] attain tighter bounds of $\mathcal{O}(\sqrt{\Gamma(T)T})$. Among these, BPE is especially relevant to our work, as it introduces a multi-round exploration structure that has inspired the design of our MR-LPF algorithm.

[1] [117] does not explicitly report the scaling of the regret bound with κ.

However, there are differences in the inference procedure and analysis, due to the use of a reduced preference-based feedback model, which introduces additional complexities in both algorithm design and theoretical analysis.

6.1.1.2 Dueling Bandits

The BOHF framework can be viewed as an extension of bandits with preference-based feedback, also known as dueling bandits [87, 94]. For a detailed review of this framework—see Section 3.4. In summary, earlier work in this area largely focused on finite-action settings and aimed to learn a pairwise preference matrix using tournament-based strategies or noisy sorting procedures [98, 230, 231, 99]. These methods do not scale well to infinite or large action spaces. To address this, linear contextual dueling bandits [93, 103, 232, 104, 105] introduced parametric utility models that generalize across actions, albeit under the restrictive assumption of linearity. More recent work has extended the dueling bandit problem to kernel-based settings, though these still differ from our BOHF framework. For instance, [113, 114, 115] reduce the problem to conventional BO by making strong assumptions on the Borda function. In contrast, our analytical requirements are significantly different from these approaches. A recent extension by [118] considers neural dueling bandits with a wide neural network for preference prediction. Their approach differs from ours in both modeling and action selection, with regret bounds that depend on the model's effective dimension and the curvature parameter κ.

6.1.1.3 Reinforcement Learning from Human Feedback (RLHF)

Another related line of work is RLHF [233, 234, 235, 236, 108, 237], which has gained popularity due to its success in finetuning LLMs [238]. In this context, preference-based feedback is provided for MDP trajectories or policies rather than pairs of actions. However, these results are primarily limited to tabular (finite state-action) or linear settings and are not directly related to our kernel-based setting.

6.2 Preliminaries and Problem Formulation

In this section, we provide details of the BOHF framework. We also outline the methods used to predict preference functions and estimate uncertainty, which form the foundation of our algorithm's design and analysis.[2]

6.2.1 BOHF Framework

At each step $t = 1, 2, \cdots, T$, the agent selects a pair of actions x_t and x'_t, from the set $\mathcal{X}$, which can either be a continuous space or a (possibly very large) discrete set. We consider the following feedback model: Let $y_t \in \{0, 1\}$ be a binary random variable indicating the preference between x_t and x'_t, defined as $y_t = \mathbb{1}\{x_t \succ x'_t\}$. The notation $x_t \succ x'_t$ denotes that action x_t is preferred over action x'_t and $\mathbb{1}$ is the indicator function. Specifically, following the existing work, for each pair $(x, x') \in \mathcal{X} \times \mathcal{X}$, the random variable $y = \mathbb{1}\{x \succ x'\}$ is modelled as a Bernoulli random variable satisfying $\mathbb{P}(y = 1 | x, x') = \mu(f(x) - f(x'))$. $\mathbb{P}(y_t = 1 | x_t, x'_t) = \mu(f(x_t) - f(x'_t)) = \mu(h(x_t, x'_t))$. Here, $\mu : \mathbb{R} \to [0, 1]$ is a known monotonically increasing link function satisfying $\mu(0) = \frac{1}{2}$ that is assumed to be the sigmoid function $\mu(\cdot) = (1 + e^{-\cdot})^{-1}$, and $f : \mathcal{X} \to \mathbb{R}$ is an unknown latent utility function that quantifies the value of each action. This preference feedback model is referred to as the Bradeley-Terry-Luce (BTL) model [102] and is widely utilized in bandit and RL problems with preference feedback [116, 117, 109, 236].

We note that when $f(x) > f(x')$, we have $\mathbb{P}(x \succ x') = \mathbb{P}(y = 1 | x, x') = \mu(f(x) - f(x')) > \frac{1}{2}$, and vice versa. We also emphasize that this feedback model is weaker than the standard BO where the per-step utility signal (the quantitative value of f) is revealed, typically as a scalar value.

The goal is to sequentially select favorable action pairs over a horizon of

[2]Several symbols used in this chapter take on meanings that differ from earlier chapters: (i) $z = (x, x')$ denotes a pair of actions in BOHF (cf. Chapter 4, where z is a latent skill, and Chapter 5, where $z = (s, a)$ is a state-action pair); (ii) h denotes a preference function $h : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, in contrast to its use as the episode step in Chapter 5; (iii) r is used to denote the round index in the BOHF setting, not to be confused with the reward variable used in the previous chapters; (iv) the regularization parameter τ^2 in kernel ridge regression is denoted as λ in this chapter. All notations are defined locally when first introduced.

T steps, and converge to the globally preferred action $x^\star$, defined as $x^\star = \arg\max_{x\in\mathcal{X}} f(x)$. A common objective adopted in the literature is to design an algorithm with sublinear cumulative regret over the horizon T, defined as the sum of the average sub-optimality gap between the selected pair and the globally optimal action:

$$R(T) = \sum_{t=1}^{T} \frac{\mathbb{P}(x^\star \succ x_t) + \mathbb{P}(x^\star \succ x_t') - 1}{2}. \tag{6.1}$$

It can be shown that the value of regret above is equivalent to a variation of regret defined on the utility function: $\sum_{t=1}^{T} (f(x^\star) - (f(x_t) + f(x_t'))/2)$—used in [117]—up to constants that depend on the link function [107].

The notion of regret accounts for the entire sequence of query points throughout steps $t = 1, 2, \ldots, T$. Alternatively, one may be interested solely in the final performance. In this case, the algorithm outputs $\hat{x}_T$ at the end of T samples, and the performance is measured in terms of $\mathbb{P}(x^\star \succ \hat{x}_T) - \frac{1}{2}$. We refer to the number of samples T required to ensure $\mathbb{P}(x^\star \succ \hat{x}_T) - \frac{1}{2} \leq \epsilon$, for some $0 < \epsilon < 1/2$, as the sample complexity and also remark on the sample complexity of different algorithms.

An important quantity that appears in the analysis is

$$\kappa = \sup_{x,x'\in\mathcal{X}} \frac{1}{\dot{\mu}(f(x) - f(x'))}, \tag{6.2}$$

where $\dot{\mu}$ denotes the derivative of the link function μ and κ captures its curvature. The dependence on κ has been extensively studied in linear logistic bandits, with recent works successfully removing the regret dependency on κ [239]. To emphasize the significance of this quantity, consider the case where f is bounded within the interval $[-5, 5]$. In this scenario, κ can become extremely large (> 22028). When the algorithm selects an action pair (x, x') that are nearly equally favorable, $f(x) - f(x')$ will be close to 0, in which case the inverse derivative of the sigmoid function is almost a constant 4. However, when one action is clearly preferred over the other, $|f(x) - f(x')|$ becomes large, making the inverse derivative of the sigmoid function very large. Therefore, a crucial aspect of algorithm design is to remove the dependency on κ defined in (6.2) by ensuring that the algorithm gradually queries only closely preferred actions.

6.2.2 Preliminaries and Assumptions

Similar to [116, 117], we assume that the utility function f belongs to a known Reproducing Kernel Hilbert Space (RKHS). This is a very general assumption, considering that the RKHS of common kernels can approximate almost all continuous functions on the compact subsets of $\mathbb{R}^d$ [67] . Consider a positive definite kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Let $\mathcal{H}_k$ be the RKHS induced by k, where $\mathcal{H}_k$ contains a family of functions defined on $\mathcal{X}$. Let $\langle \cdot, \cdot \rangle_{\mathcal{H}_k} : \mathcal{H}_k \times \mathcal{H}_k \to \mathbb{R}$ and $\|\cdot\|_{\mathcal{H}_k} : \mathcal{H}_k \to \mathbb{R}$ denote the inner product and the norm of $\mathcal{H}_k$, respectively. For a formal statement on RKHS and Mercer Theorem, please see Appendix B.5. Let us use the notation $z = (x, x')$ and $h(z) = f(x) - f(x')$, for $(x, x') \in \mathcal{X} \times \mathcal{X}$. As shown in [116], we can define a *dueling* kernel

$$\Bbbk(z_1, z_2) = k(x_1, x_2) + k(x_1', x_2') - k(x_1, x_2') - k(x_1', x_2), \tag{6.3}$$

where, we have: $\|f\|_{\mathcal{H}_k} = \|h\|_{\mathcal{H}_{\Bbbk}}$ (see [116], Proposition 4).
Below is a formal statement of our assumptions on f.

Assumption 4. *We assume that the utility function is in the RKHS of a known kernel k satisfying $\|f\|_{\mathcal{H}_k} \leq B$ for some constant $B > 0$. Without loss of generality, we assume that the kernel function is normalized $k(.,.) \leq 1$ everywhere in the domain.*

6.2.3 Preference Function Prediction and Uncertainty Estimation

The preference-based feedback model in BOHF is weaker than the standard BO, where quantitative observations of utility are available at each step. Before discussing the case with preference feedback, we briefly recall kernel ridge regression in the standard BO setting from Chapter 3, Section 3.3.3.

Hypothetically, assume a dataset $\{(x_i, o_i)\}_{i=1}^t$ of observations of f is available, where $o_i = f(x_i) + \varepsilon_i$, with observation noise ε_i. As described in Section 3.3.3, kernel ridge regression provides a predictor $\hat{f}_t$ and an uncertainty estimate $\sigma_t^2(x)$ by minimizing a regularized least-squares error optimization, leading to closed-form expressions for both quantities:

$$\begin{aligned}\hat{f}_t(x) &= k_t^\top(x)(K_t + \lambda I)^{-1}\boldsymbol{o}_t \\ \sigma_t^2(x) &= k(x,x) - k_t^\top(x)(K_t + \lambda I)^{-1}k_t(x),\end{aligned} \tag{6.4}$$

where $k_t(x) = [k(x, x_i)]_{i=1}^t$ represents the pairwise kernel values between the prediction point x and the observation points, $K_t = [k(x_i, x_j)]_{i,j=1}^t$ is the kernel (or covariance) matrix, $\lambda > 0$ is a free parameter, and $\boldsymbol{o}_t = [o_i]_{i=1}^t$ is the vector of observation values.

Confidence intervals of the form $|f(z) - \hat{f}_t(z)| \leq \beta(\delta)\ \sigma_t(z)$, where $\beta(\delta)$ is a confidence interval width multiplier for a $1-\delta$ confidence level, have been shown in several works [213, 66, 70, 72] under various assumptions, and serve as key building blocks in the analysis and algorithm design of standard BO.

In the absence of straightforward observations $\boldsymbol{o}_t$ and with preference-based feedback, a closed-form prediction is no longer available. Intuitively, this case resembles a classification-like problem with binary outputs, where we can employ a logistic negative log-likelihood loss. Specifically, for a history of preference feedback $\mathbb{H}_t = (x_1, x_1', y_1), \ldots, (x_t, x_t', y_t)$ in the BOHF framework, we define the following loss:

$$\begin{aligned}\mathcal{L}_{\Bbbk}(h, \mathbb{H}_t) = \sum_{i=1}^t & -y_i \log \mu(h(x_i, x_i')) \\ & - (1-y_i)\log(1 - \mu(h(x_i, x_i')) + \frac{\lambda}{2}||h||_{\mathcal{H}_{\Bbbk}}^2\end{aligned}$$

A prediction h_t of the preference function h (difference in the utilities) can be obtained as:

$$h_t = \arg\min_{h \in \mathcal{H}_{\Bbbk}} \mathcal{L}_{\Bbbk}(h, \mathbb{H}_t), \tag{6.5}$$

which represents the minimizer of the regularized negative log-likelihood loss.

To solve this minimization problem, we apply the Representer Theorem, similar to [116], which provides a parametric representation of h_t:

$$h_t(\cdot) = \sum_{i=1}^{t} \theta_i \Bbbk\left(\cdot, (x_i, x_i')\right), \tag{6.6}$$

in terms of $\boldsymbol{\theta}_t = [\theta_1, \theta_2, \cdots, \theta_t]^\top \in \mathbb{R}^t$. With a slight abuse of notation, replacing h with $\boldsymbol{\theta}$ in $\mathcal{L}_{\Bbbk}$, the regularized negative log-likelihood loss can then be rewritten in terms of the parameter vector $\boldsymbol{\theta}$ as follows:

$$\begin{aligned}\mathcal{L}_{\Bbbk}(\boldsymbol{\theta}, \mathbb{H}_t) = \sum_{i=1}^{t} & -y_i \log \mu(\boldsymbol{\theta}^\top \Bbbk_t(x_i, x_i')) \\ & - (1 - y_i) \log(1 - \mu(\boldsymbol{\theta}^\top \Bbbk_t(x_i, x_i')) + \frac{\lambda}{2} ||\boldsymbol{\theta}||_2^2, \end{aligned} \tag{6.7}$$

where $\Bbbk_t(z) = [\Bbbk(z, (x_j, x_j'))]_{j=1}^t$ is the kernel values between the pair z and observation pairs.

Similar to (6.4), we have an uncertainty estimation for each $z \in \mathcal{X} \times \mathcal{X}$ as follows

$$\sigma_t^2(z) = \Bbbk(z, z) - \Bbbk_t^\top(z)(\mathbb{K}_t + \lambda\kappa I)^{-1}\Bbbk_t(z), \tag{6.8}$$

where the notation $\mathbb{K}_t = [\Bbbk\left((x_i, x_i'), (x_j, x_j')\right)]_{i,j=1}^t$ represents the (dueling) kernel matrix on the space of pair observations $\mathcal{X} \times \mathcal{X}$. Note the subtle difference in the definition of σ_t^2 above for the preference-based feedback case compared to the conventional kernel-based regression case, where the free parameter λ is multiplied by κ, reflecting the effect of the sigmoid nonlinearity on the quality of prediction.

Centered around the prediction $\mu(h_t(\cdot))$ and incorporating the uncertainty estimate from kernel ridge regression, as defined in Equation (6.8), we can construct $1-\delta$ confidence intervals of the form:

$$|\mu(h_t(z)) - \mu(h(z))| \leq \beta_t(\delta)\sigma_t(z),$$

for a pair of interest $z = (x, x')$. In Theorem 5, we prove a novel confidence interval of this form applicable to the analysis of our algorithm.

6.3 Algorithm Description

In this section, we present the MR-LPF algorithm, inspired by [84], designed to achieve low regret within the BOHF framework described in Section 6.2.1.

The algorithm partitions the time horizon T into R rounds, indexed by $r = 1, 2, \dots, R$. During each round r, a total of N_r samples are collected, ensuring that the cumulative number of samples across all rounds equals T, i.e., $\sum_{r=1}^{R} N_r = T$. We define $t_r = \sum_{j=1}^{r} N_j$ as the time step at the end of round r. The size of each round is determined as follows: $N_1 = \lceil\sqrt{T}\rceil$, $N_r = \lceil\sqrt{N_{r-1}T}\rceil$ for $1 < r < R$, and $N_R = \min\{\lceil\sqrt{N_{R-1}T}\rceil, T - t_{R-1}\}$.

We introduce the notations $\sigma_{(n,r)}(x, x')$ and $h_{(n,r)}(x, x')$ to represent the kernel-based uncertainty estimate and prediction, respectively, from the first n samples in round r according to Section 6.2.3.

MR-LPF maintains a set $\mathcal{M}_r$ of actions in each round that are likely to be the most preferable. Initially, $\mathcal{M}_1$ is set to $\mathcal{X}$ and is updated at the end of each round while satisfying a nested structure, $\mathcal{M}_r \subseteq \mathcal{M}_{r-1}$, as subsequently described.

Within each round r, the n-th sample is chosen as the pair of actions within $\mathcal{M}_r$ that maximizes uncertainty :

$$(x_{(n,r)}, x'_{(n,r)}) = \arg\max_{x,x' \in \mathcal{M}_r} \sigma_{(n-1,r)}(x, x'). \tag{6.9}$$

The preference feedback for this pair $y_{(n,r)} = \mathbb{1}\{x_{(n,r)} \succ x'_{(n,r)}\}$ is then revealed to the algorithm. The tuple $(x_{(n,r)}, x'_{(n,r)}, y_{(n,r)})$ is added to the observations specific to round r: $\mathbb{H}_{n,r} = \mathbb{H}_{n-1,r} \cup \{(x_{(n,r)}, x'_{(n,r)}, y_{(n,r)})\}$, which is initialized as an empty set at the beginning of the round: $\mathbb{H}_{0,r} = \emptyset$.

At the end of round r, we compute the prediction function $h_{(N_r,r)}$ based on observations $\mathbb{H}_{N_r,r}$, following the method of minimizing the regularized negative log-likelihood loss described in Section 6.2.3. Subsequently, we update $\mathcal{M}_r$ according to the following rule:

$$\begin{aligned}\mathcal{M}_{r+1} = \{x \in \mathcal{M}_r | \forall x' \in \mathcal{M}_r :& \\ \mu(h_{(N_r,r)}(x,x')) + \beta_{(r)}\sigma_{(N_r,r)}(x,x') \geq 0.5\}.& \qquad (6.10)\end{aligned}$$

The round specific parameters $\beta_{(r)}$ are designed in a way that the left hand side of the inequality is a UCB on the probability of favoring x over x' (the values are given in Theorem 4). The rationale here is that when a UCB on the probability of preferring x to any x' is greater than 0.5, x is plausible to be the most preferred action. Therefore, we keep it in the update of $\mathcal{M}_{r+1}$. All other actions are removed as they are unlikely to be the most preferred. More precisely, as we will show in the analysis, with high probability, the removed actions are not the most preferred, while the most preferred actions remain within the sets $\mathcal{M}_r$ and are not removed. A pseudocode is provided in Algorithm 10.

When the confidence intervals shrink at a sufficiently fast rate, only near-optimal actions remain in $\mathcal{M}_r$ as the rounds progress. This is a key aspect of our algorithm's design, which eliminates the dependency of regret scaling on κ by ensuring that the algorithm gradually queries only closely preferred actions. Recall the discussion following Equation (6.2). In the next section, we provide an analysis of the performance guarantees of the algorithm.

6.4 Analysis of MR-LPF

In this section, we present our main results on the performance of MR-LPF (Algorithm 10). The performance is given in terms of the maximum informa-

Algorithm 10 MR-LPF

```
Require: $\forall r, \beta_{(r)}$; time horizon $T$
  $\mathcal{M}_1 \leftarrow \mathcal{X}$, $t \leftarrow 1$
  for $r = 1, 2, \cdots, R$ do
    Initialize $\mathbb{H}_{0,r} = \{\}$
    for $n = 1, 2, \cdots, N_r$ do
      Select the pair of actions $(x_{(n,r)}, x'_{(n,r)})$ that maximizes the variance, with ties broken arbitrarily:
      $(x_{(n,r)}, x'_{(n,r)}) = \arg\max_{x,x' \in \mathcal{M}_r} \sigma_{(n-1,r)}(x, x')$
      $t \leftarrow t + 1$
      if $t \geq T$ then
        Terminate
      end if
      Observe $y_{(n,r)} = \mathbb{1}\{x_{(n,r)} \succ x'_{(n,r)}\}$
      $\mathbb{H}_{n,r} = \mathbb{H}_{n-1,r} \cup \{(x_{(n,r)}, x'_{(n,r)}, y_{(n,r)})\}$
    end for
    Update $h_{(N_r,r)}$ based on observations in $\mathbb{H}_{N_r,r}$
    Update the set of maximizers $\mathcal{M}_{r+1}$ by removing actions unlikely to be optimal:
    $\mathcal{M}_{r+1} = \{x \in \mathcal{M}_r | \forall x' \in \mathcal{M}_r : \mu(h_{(N_r,r)}(x, x')) + \beta_{(r)}\sigma_{(N_r,r)}(x, x') \geq 0.5\}$
  end for
```

tion gain defined as

$$\Gamma_\lambda(T) = \max_{(x_1,x'_1),\ldots(x_T,x'_T)} \frac{1}{2} \log\det\left(I + \lambda^{-1}\mathbb{K}_T\right), \tag{6.11}$$

where $\mathbb{K}_T$ is the kernel matrix of T observations.[3]

Theorem 4 (Regret bound for MR-LPF)**.** *Consider the BOHF framework described in Section 6.2.1 and the MR-LPF algorithm presented in Algorithm 10. For $\delta \in (0,1)$, in MR-LPF, let*

$$\beta_{(r)}(\delta) = L\left(B + \sqrt{\frac{\kappa_r}{\lambda}\log(\frac{2R|\mathcal{X}|}{\delta})}\right), \tag{6.12}$$

where, B is the upper bound on the RKHS norm of f given in Assumption 4, $L = \max_{x,x' \in \mathcal{X}} \dot{\mu}(h(x,x'))$, $\kappa_1 = \kappa$ defined in Equation (6.2), $\forall r > 1, \kappa_r = 6$, λ

[3]This is the same maximum information gain formula introduced in Section 3.3.5, Equation (3.4), except that we use λ instead of τ^2. The two definitions are mathematically equivalent. Unlike in Section 6.1, where λ was omitted from the expression of Γ, we include it here for clarity.

is the regularization parameter of the kernel-based regression. Then, for some constant $T_0 > 0$*, independent of* T *(specified in Appendix C.1), and for all* $T \geq T_0$*, with probability at least* $1-\delta$*:*

$$R(T) \leq 2CR\beta_{(R)}(\delta)\sqrt{\Gamma_{(4\lambda)}(T)}\left(T^{1/2}+1\right),$$

where $R \leq \lceil \log_2 \log_2(T) \rceil + 1$ *is the maximum number of rounds and* $C = 2\sqrt{\frac{2}{\log(1+4(6\lambda)^{-1})}}$ *is a constant.*

Remark 4. *The value of* $\Gamma_\lambda(T)$ *is kernel-specific and algorithm-independent. This term is a common complexity measure that appears in the analysis of both BO and BOHF in the existing literature (see e.g., [67, 116, 117]). Bounds on* $\Gamma_\lambda(T)$ *have been established for various kernels, as discussed in Chapter 3 (Section 3.3.5). In particular, for linear kernels,* $\Gamma_\lambda(T) = \mathcal{O}(d\log(T))$*. For kernels with exponentially decaying Mercer eigenvalues, such as the SE kernel,* $\Gamma_\lambda(T) = \mathcal{O}(\mathit{poly}\log(T))$*. For kernels with polynomially decaying eigenvalues,* $\Gamma_\lambda(T)$ *grows polynomially (though sublinearly) with* T*. For example, in the case of the Matérn family of kernels,* $\Gamma_\lambda(T) = \tilde{\mathcal{O}}(T^{\frac{d}{2\nu+d}})$*, where* d *is the input dimension and* $\nu > 0.5$ *is the smoothness parameter (see, e.g., [68]). In Proposition 4 of [116], it is shown that the eigenvalues of the dueling kernel* $\mathbb{k}$ *are exactly twice those of the original kernel* k *(see their Appendix C.1). Since the maximum information gain* $\Gamma_\lambda(T)$ *scales with the decay rate of the kernel eigenvalues [68], both kernels exhibit the same scaling of the information gain with* T*.*

Remark 5. *By substituting the value of* $\beta_{(R)}(\delta)$*, the expression of the regret bound can be simplified to*

$$R(T) = \tilde{\mathcal{O}}\left(\sqrt{\Gamma_\lambda(T)T\log\left(\frac{|\mathcal{X}|}{\delta}\right)}\right), \tag{6.13}$$

as T *becomes large. This represents a sublinear regret growth rate for a broad class of commonly used kernels where* $\Gamma_\lambda(T)$ *grows sublinearly with* T*.*

Our regret bounds eliminate the dependency on κ. MR-LPF gradually queries only closely preferred actions, reducing the effective impact of the curvature of the link function. Our regret bounds also show an $\mathcal{O}\left(\sqrt{\Gamma(T)}\right)$ improvement compared to [116] and an $\mathcal{O}\left((\Gamma(T)T)^{1/4}\right)$ improvement over [117]. This becomes particularly crucial for kernels with polynomially decaying eigenvalues, where existing results may become vacuous, failing to guarantee sublinear regret in T.

6.4.1 Sample Complexity and Simple Regret

In certain applications, the learner may be primarily concerned with eventual performance, specifically the simple regret after T observations. Accordingly, we can pose the dual question: *How many samples are required to achieve ϵ simple regret?* This aspect of our algorithm's performance is formalized in the following corollary.

Corollary 2. *Under the setting of Theorem 4, assume $T = t_R$, the time step at the end of round R. For any action $\hat{x}_T \in \mathcal{M}_{R+1}$, we have, with probability at least $1-\delta$,*

$$\mathbb{P}(x^\star \succ \hat{x}_T) - \frac{1}{2} \leq 2\beta_{(R)}(\delta) C \sqrt{\frac{R\Gamma_{(4\lambda)}(T)}{T}}. \tag{6.14}$$

The proof is given in Appendix C.1, that follows from Theorem 4.

Corollary 3. *As a consequence of Corollary 2, assume we run MR-LPF for $T = t_R$ rounds and select $\hat{x}_T \in \mathcal{M}_{R+1}$ arbitrarily. In the case of a linear kernel with some $T = \tilde{\mathcal{O}}\left(\frac{d\log(\frac{1}{\delta})}{\epsilon^2}\right)$, an SE kernel with some $T = \tilde{\mathcal{O}}\left(\frac{\log(\frac{1}{\delta})}{\epsilon^2}\right)$, and a Matérn kernel with some $T = \tilde{\mathcal{O}}\left(\frac{\log(\frac{1}{\delta})}{\epsilon^{2+\frac{d}{\nu}}}\right)$, with probability at least $1-\delta$, at most ϵ error is guaranteed: $\mathbb{P}(x^\star \succ \hat{x}_T) - \frac{1}{2} \leq \epsilon$.*

Remark 6. *Our sample complexities match the $\Omega\left(\frac{1}{\epsilon^{2+\frac{d}{\nu}}}\right)$ lower bounds for conventional BO with Matérn kernels, as established in [65] (up to logarithmic terms). These bounds apply to scalar-valued feedback, which is richer than the binary preference feedback used in BOHF.*

For technical details, consider a standard BO setting with scalar observations $o_i = f(x_i) + \varepsilon_i$, where ε_i are i.i.d., zero-mean noise terms (following *the notation in Section 6.2.3). Suppose that at each step t, instead of observing $o_t = f(x_t) + \varepsilon_t$ and $o'_t = f(x'_t) + \varepsilon'_t$, we receive binary preference feedback $y_t = \mathbb{1}\{o_t > o'_t\}$. Under the BTL model, this corresponds to the case where the noise difference $\varepsilon'_t - \varepsilon_t$ follows a logistic distribution, which can arise if the individual noise terms ε_t are Gumbel-distributed. Thus, the lower bound on sample complexity in the BOHF setting should be at least half of that of conventional BO under Gumbel noise for achieving at most ϵ loss in the value of the target function.*

Since the lower bound construction in [65] assumes Gaussian noise, a formal comparison is not strictly valid (as the BTL model corresponds to Gumbel noise). We therefore present this connection as an informal justification of the tightness of our bounds, rather than a formal optimality proof.

6.4.2 Confidence Intervals and Proofs

An important building block in analyzing the performance of MR-LPF is the confidence intervals applied to the samples collected in each round. We now present a formal statement of this result.

Theorem 5 (Confidence Bounds)**.** *Consider the kernel-based prediction h_t and uncertainty estimate σ_t for a dataset $\mathbb{H}_t$ and a known kernel $\mathbb{k}$, as given in Equations* (6.5) *and* (6.8) *satisfying Assumption 4. Assume the observation points $\{(x_i, x'_i)\}_{i=1}^t$ are independent of the observation values $\{y_i\}_{i=1}^t$. For a fixed $(x, x') \in \mathcal{X} \times \mathcal{X}$ and for any $\delta \in (0,1)$, we have, with probability at least $1-\delta$,*

$$|\mu(h_t(x,x')) - \mu(h(x,x'))| \leq \beta(\delta)\sigma_t(x,x'), \tag{6.15}$$

where $\beta(\delta) = L\left(B + \frac{1}{2}\sqrt{\frac{2\kappa}{\lambda}\log(2/\delta)}\right)$, $L = \sup_{x,x' \in \mathcal{X}} \dot{\mu}(h(x,x'))$ as defined in Theorem 4, B is the RKHS norm bound specified in Assumption 4, λ is the parameter in kernel-based regression, and κ is defined in Equation (6.2).

A key distinction in our results is that our confidence interval is tighter than the one presented in [116] by a factor of $\mathcal{O}(\sqrt{\Gamma(T)})$. This improvement comes from the multi-round structure and action selection rule within each round of the algorithm, which ensures that the observation points used for confidence intervals at the end of rounds are independent of the observation values within that round. This removes certain intricate dependencies in deriving the confidence interval. Recall that the observation points in each round, $(x_{(n,r)}, x'_{(n,r)})$, are collected solely based on the variance, which is independent of the observation values by definition. In contrast, both the MaxMinLCB algorithm in [116] and the POP-BO algorithm in [117] select observation point at step t based on statistics that depend on $\{y_i\}_{i=1}^{t-1}$. We emphasize that our algorithm is by no means a pure exploration algorithm; it effectively balances exploration and exploitation by learning and updating $\mathcal{M}_r$ at the end of each round.

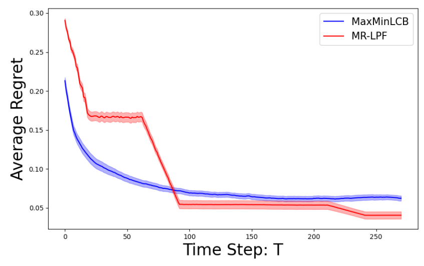
Given the confidence intervals in Theorem 5, the update rule of $\mathcal{M}_r$ in MR-LPF ensures that the best action is not eliminated (Lemma 10). Additionally, we can use the confidence intervals to bound the regret for each action in $\mathcal{M}_r$, based on the maximum variance in predictions from previous rounds. By summing up the regret over all rounds, we achieve the overall regret bound, with details provided in Appendix C.1. For proof of Theorem 5, see Appendix C.2.

6.5 Experiments

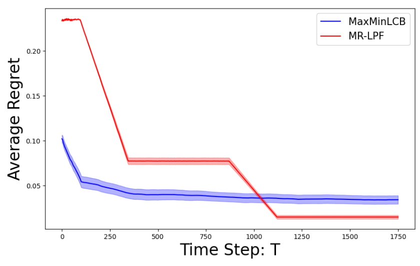
We run numerical experiments to evaluate the performance of MR-LPF and compare it to MaxMinLCB (see [116], Algorithm 1) on various test functions, including both synthetic and real-world cases. Our implementation is publicly available.[4]

We first select the test function f as an arbitrary function in the RKHS of a known kernel. To do this, we choose 10 points in the $[0, 1]$ interval and assign them random values. We then fit a standard kernel ridge regression to these samples using a kernel k and use its mean as f. The kernel k is set to the SE kernel and Matérn kernels with smoothness parameters $\nu = 2.5$ and

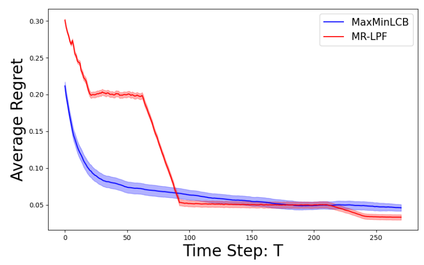
[4]`https://github.com/ayakayal/BOHF_code_submission`



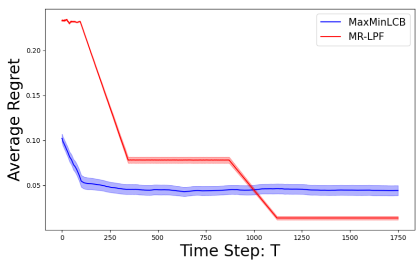
(a) SE kernel (RKHS)



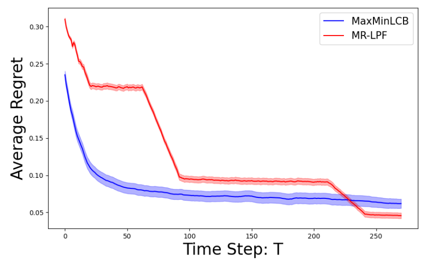
(b) SE kernel (Ackley)



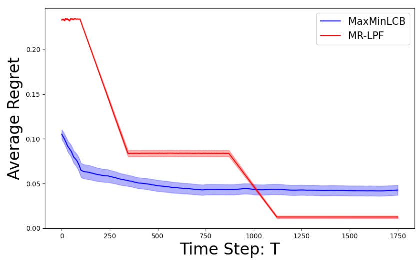
(c) Matérn kernel with $\nu = 2.5$ (RKHS)



(d) Matérn kernel with $\nu = 2.5$ (Ackley)



(e) Matérn kernel with $\nu = 1.5$ (RKHS)



(f) Matérn kernel with $\nu = 1.5$ (Ackley)

Figure 6.1: Average Regret against T with RKHS test functions (left column) and Ackley test function (right column). The shaded area represents the standard error.

$\nu = 1.5$. This is a common approach to constructing functions in an RKHS (see, e.g., [66]). We also test the algorithms on the Ackley function, similar to [116]. The Ackley function has a diverse optimization landscape, featuring multiple local minima, flat plateaus, and valleys, making it a popular choice in non-convex optimization literature [240].

To showcase the utility of our approach in real-world applications, we experimented using the Yelp Open Dataset[5] of restaurant reviews. This serves as a proof of concept, demonstrating both the potential integration of BOHF with LLM-generated vector embeddings and the scalability of the method to higher-

[5]Yelp Open Dataset

dimensional domains. The objective is to learn user preferences from comparative feedback and recommend restaurants tailored to each user's choices. After data filtering and pre-processing, the dataset consists of 275 restaurants, 20 users, and 2563 reviews. Each restaurant is represented by a 32-dimensional vector embedding of its text-based reviews, generated using OpenAI's text-embedding-3-large model[6]. Users rate restaurants on a scale from 1 to 5. We adopt the experimental setup and Yelp data preprocessing from [116] to ensure a fair evaluation. While we implemented our own version instead of using their code[7] directly, we acknowledge their contribution in establishing this benchmark, which inspired our experiment. We frame this problem within the BOHF framework, where the action set $\mathcal{X}$ consists of 275 restaurants, each represented as a 32-dimensional vector, and the utility values f correspond to user ratings. SE kernel is used for these experiments. For details on the experimental setup, see Appendix C.3.

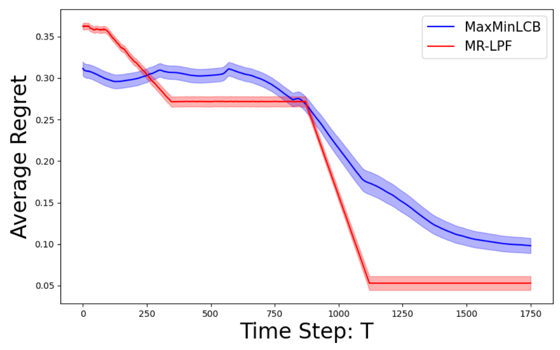


Figure 6.2: Average regret against T for the experiment with Yelp Open Dataset. The shaded area represents the standard error.

We plot the average regret at each time step, averaged over 60 independent runs. Figure 6.1 shows the results on the RKHS and Ackley test functions, while Figure 6.2 presents the results on the Yelp Open Dataset. MR-LPF consistently achieves lower regret than MaxMinLCB across all test functions. The initial regret of MR-LPF reflects highly exploratory behavior during the early rounds. At the end of each round r, suboptimal actions are removed

[6]OpenAI Vector Embeddings
[7]`https://github.com/lasgroup/MaxMinLCB`.

from $\mathcal{M}_r$, leading to the sharp drops that eventually result in near-optimal actions in later rounds. Relatively constant behavior within rounds represents exploration, while sharp drops indicate exploitation.

6.6 Conclusion

In this chapter, we proposed an algorithm, referred to as MR-LPF (Algorithm 10), to address the BOHF problem. We provided a comprehensive performance analysis under relatively general assumptions, demonstrating that MR-LPF achieves a regret bound of $\tilde{O}\left(\sqrt{\Gamma_\lambda(T)T}\right)$ for general kernels. This result represents a significant improvement over existing approaches, effectively tightening the regret by a factor of $\sqrt{\Gamma(T)}$ and eliminating the dependence on κ. Our results recover the order-optimal sample complexities achieved by conventional BO, showing they are not improvable in general. In particular, this implies that the number of preferential feedback samples required to identify near-optimal actions is of the same order as the number of scalar-valued feedback samples. To validate these findings, we conducted numerical experiments on both synthetic benchmarks and real-world examples. The results consistently corroborate our theoretical guarantees and demonstrate that MR-LPF outperforms the most competitive existing algorithm.

While our experiments were designed to align closely with the theoretical analysis, there are exciting opportunities to further explore the practical impact of MR-LPF. A natural next step is to demonstrate its utility in larger-scale applications. For example, applying our method to scenarios where preference feedback is elicited from interactions with LLMs—such as prompt optimization or text summarization comparison—would provide a compelling extension. Although such experiments require extensive setup and careful implementation, we view them as a promising direction for future work.

Chapter 7

Conclusions

Efficiency is a cornerstone on the path toward AGI, as progress will ultimately depend not only on raw computational power but also on the ability to learn and reason effectively from limited data and resources. In this thesis, we focused on advancing data-efficient strategies for sequential decision-making, with a particular emphasis on improving exploration. In this final chapter, we summarize the key contributions of the thesis and discuss potential directions for future research arising from our findings.

7.1 Summary of Contributions

This thesis first addressed fundamental questions in deep RL that the community has yet to answer definitively. In particular: What constitutes good exploration in practice? Can different exploration methods be directly compared? How can an RL agent efficiently explore its environment? Since no consensus exists on the optimal exploration strategy, in Chapter 4 we presented a re-interpretation of exploration bonuses (intrinsic rewards) based on the level of diversity they promote—namely, state, policy, and skill diversity. We conducted an empirical study to examine how these different levels of diversity affect exploration. Our results showed that state-level diversity led to the best exploration performance in environments with low-dimensional observations. In contrast, policy-level diversity was more effective in high-dimensional settings, possibly owing to its robustness to challenges in representation learning. To the best of our knowledge, Skill-level diversity, often linked to robustness, did not contribute positively to exploration in MiniGrid environments. These findings deepened our understanding of how exploration strategies can be adapted to different environments. They also motivated our shift toward a theoretical analysis of exploration, as the empirical definition of optimal exploration remains elusive. In Chapter 5, we strengthened the theoretical foundations of exploration in RL. We extended the analytical study of RL beyond the well-studied tabular and linear settings to the more flexible kernel-based setting, which supports nonlinear function approximation and provides a stepping stone toward understanding RL in neural network-based settings. We studied reward-free RL with kernel-based models and proposed two algorithms for data collection and optimal policy derivation, under both the presence and absence of a generative model. Each algorithm was supported by a sample complexity analysis and empirical validation. Our contributions included relaxed assumptions compared to prior work and the derivation of novel confidence intervals for kernel ridge regression, which have broader applications beyond RL. Finally, based on the insights gained from previous chapters and the growing importance of learning from preference feedback—particularly for finetuning and aligning LLMs—we investigated how to efficiently explore in order to learn from human preferences with minimal feedback. In Chapter 6, we framed this problem as Bayesian Optimization from Human Feedback (BOHF). We proposed a novel algorithm, Multi-Round Learning from Preference-based Feedback (MR-LPF), that efficiently solves this problem. We proved an improved regret bound for MR-LPF compared to existing algorithms and showed that it matched the lower bound of conventional BO, despite relying on a weaker preferential feedback model. We validated our approach through experiments on both synthetic and real-world datasets.

7.2 Future Work

Building on the findings of this thesis, several directions for future research emerge, which we outline in the following section.

7.2.1 Theoretical Framework Unifying Exploration Strategies

While Chapter 4 presents an empirical study of various exploration bonuses across different types of environments—highlighting the strengths and limitations of each—there remains a lack of a unified theoretical framework that connects these diverse approaches. To deepen our understanding of how these methods function, it is necessary to formalize them within a common theoretical structure by deriving a general objective function that encapsulates multiple exploration strategies as special cases.

Such a framework would enable a principled comparison of intrinsic reward mechanisms in terms of sample complexity, computational efficiency, and robustness to environmental perturbations. Ultimately, this would not only clarify existing strategies but also guide the development of novel and more efficient exploration methods.

7.2.2 Exploration as Key to Generalization in RL

Effective exploration in RL is not just about finding optimal policies for the training environments—it also helps in acquiring behaviors that generalize to unseen tasks, highlighting the deep connection between exploration and generalization [241]. One concrete approach to studying this connection is through the exploration of task-agnostic skills—general-purpose, reusable behaviors that can serve as building blocks for solving diverse downstream tasks. This perspective aligns with the discussion on skill learning in Chapter 4 and parallels the rationale behind foundation models: just as large models are pre-trained to capture broadly useful knowledge, RL agents discover a repertoire of skills (goal-conditioned policies) through intrinsically motivated exploration. These skill repertoires can act as behavioral priors, allowing agents to adapt faster to novel tasks by composing and refining pre-existing skills.

A central challenge in skill learning is balancing diversity, efficiency, and generalization. Skills must be diverse, covering a broad range of behaviors to maximize transfer potential. Exploration should also be efficient, prioritizing informative behaviors, avoiding redundancy, and not wasting effort on unachievable goals. Ultimately, the acquired skill set should generalize effectively, enabling agents to rapidly solve new tasks with minimal additional learning.

Open questions include: How can we quantify a skill set's diversity, efficiency, and generalization? How can we design richer goal representations and mechanisms for long-horizon goals? What strategies can support efficient skill adaptation and composition? Equally important are robust benchmarks and principled metrics to evaluate whether skill repertoires truly facilitate exploration and zero-shot generalization. Progress will require a deeper understanding of how exploration shapes transferable behaviors and how such behaviors can be composed, refined, and reused across tasks.

7.2.3 Active Exploration for LLM Alignment

LLMs have revolutionized AI, yet aligning them with human values remains a critical challenge. LLM alignment ensures that these models generate responses that are not only coherent but also align with human intent, ethical considerations, and safety constraints [238, 242]. The first method introduced for alignment has been Reinforcement Learning from Human Feedback (RLHF), which involves training a reward model (RM) based on human preference comparisons and then optimizing a policy against this RM using RL [243, 244]. More recently, direct preference-based alignment methods have gained traction as simpler and more efficient alternatives to RLHF. One prominent example is Direct Preference Optimization (DPO), which directly updates the language model (policy) using pairwise preference data without requiring a separate RM [245]. These methods offer improvements in stability and sample efficiency. However, DPO relies on a fixed, pre-collected dataset, and often struggles with generalization, especially when confronted with out-of-distribution data [246]. To address this, recent research has explored online alignment techniques, which incorporate interactive feedback from humans or AI systems during training [247, 248, 114]. Nevertheless, acquiring human feedback at scale remains expensive and time-consuming. This challenge naturally leads to the question of active preference learning, where the goal is to make alignment more sample-efficient by strategically selecting which comparisons to query. Our recent work on BOHF contributes to this direction by proposing a framework that seeks to learn efficiently with minimal preference data. Building on this, we identify several open questions and promising avenues for future research:

1. How can active preference learning methods be adapted to operate in the vastly larger and more complex action spaces encountered in LLM alignment, while maintaining computational tractability?

2. Which acquisition strategies yield the greatest improvements in sample efficiency when selecting preference comparisons?

3. What sample complexity guarantees can be established for preference-based alignment in realistic LLM settings?

4. How robust are active preference learning strategies to noisy, inconsistent, or adversarial human feedback and what mechanisms can improve their resilience?

These and related questions highlight the need for continued research at the intersection of preference learning, alignment, and sample-efficient optimization. As LLMs continue to evolve, developing principled, scalable, and cost-effective alignment methods remains a central challenge for the field.

Appendix A

Appendix of Chapter 4

A.1 Diversity Levels Categorization

We divide intrinsic rewards into two categories: "Where to explore" and "How to explore?", as described in the following and shown in Figure A.1.

A.1.1 *"Where to Explore?"*

State level diversity:

In this subcategory, we collect all the works that encourage the exploration of unseen states. The most common method is "State Count", which stores the visitation count of each state, and gives high intrinsic rewards to encourage revisiting states with low counts. Algorithms that implement this approach include UCB, Model-based Interval Estimation (MBIE-EB), and Bayesian Exploration Bonus (BEB) [17, 159, 249]. While counting works well in tabular cases, it becomes difficult in vast state spaces. Several methods were proposed to extend State Count to large or continuous state spaces, such as pseudo-counts [131] and hashing [132].

Besides count-based methods, feature prediction error can be used as a measure of state novelty. For example, in [133], the authors assessed state novelty by distilling a fixed, randomly initialized neural network (target network) into another neural network (predictor network) trained on the data collected by the agent. This technique is called Random Network Distillation (RND), and the main motivation behind it is that the prediction error should be small

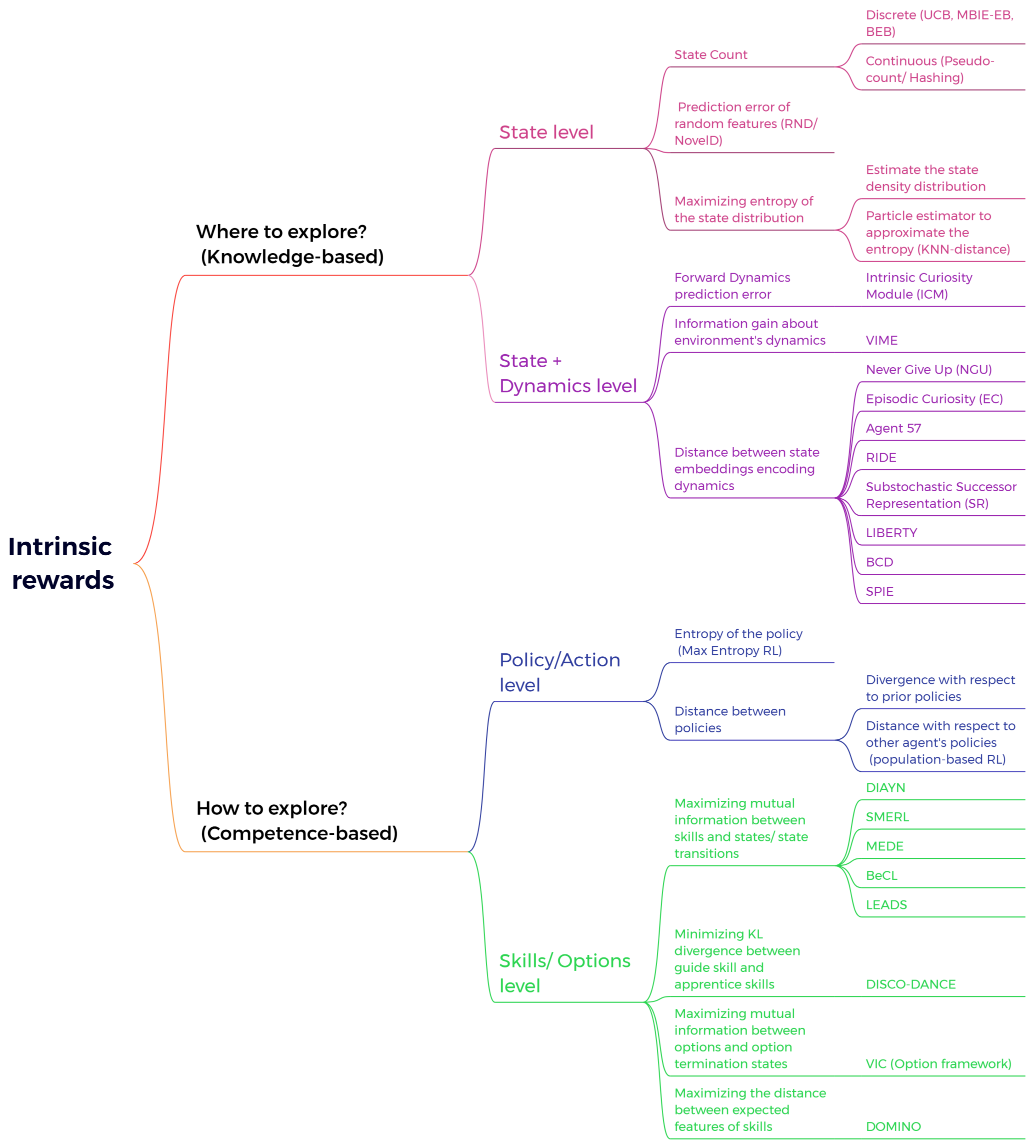

Intrinsic rewards
Where to explore? (Knowledge-based)
How to explore? (Competence-based)
State level
State + Dynamics level
Policy/Action level
Skills/ Options level
State Count
Prediction error of random features (RND/ NovelD)
Maximizing entropy of the state distribution
Discrete (UCB, MBIE-EB, BEB)
Continuous (Pseudo-count/ Hashing)
Estimate the state density distribution
Particle estimator to approximate the entropy (KNN-distance)
Forward Dynamics prediction error
Intrinsic Curiosity Module (ICM)
Information gain about environment's dynamics
VIME
Distance between state embeddings encoding dynamics
Never Give Up (NGU)
Episodic Curiosity (EC)
Agent 57
RIDE
Substochastic Successor Representation (SR)
LIBERTY
BCD
SPIE
Entropy of the policy (Max Entropy RL)
Distance between policies
Divergence with respect to prior policies
Distance with respect to other agent's policies (population-based RL)
Maximizing mutual information between skills and states/ state transitions
DIAYN
SMERL
MEDE
BeCL
LEADS
Minimizing KL divergence between guide skill and apprentice skills
DISCO-DANCE
Maximizing mutual information between options and option termination states
VIC (Option framework)
Maximizing the distance between expected features of skills
DOMINO

Figure A.1: Categorization of the different levels of diversity incurred by intrinsic rewards for exploration in RL.

for frequently visited states. Similarly, the NovelD algorithm [250] uses RND as a measure of state novelty but defines the intrinsic reward as the difference in RND prediction errors at two consecutive states, s_t and s_{t+1}, in a trajectory.

Finally, this level of diversity includes methods that aim to maximize the entropy of the state distribution induced by the policy over a finite or infinite horizon by estimating the state density distribution [212, 251] or by relying on the K-Nearest Neighbors (KNN) distance as an approximation of state entropy [252, 253, 164, 254].

State + Dynamics level diversity:
This class also aims to visit diverse states, but the difference with respect to State level is that the agent considers the novelty of the dynamics as well (not only states) to drive exploration. The agent either tries to build an accurate dynamical model of the environment or learns a dynamics-relevant state representation for exploration.

This subcategory mainly includes curiosity-driven methods that use the forward dynamics prediction error as an intrinsic reward, such as [135] and [255]. The key intuition is to encourage the agent to revisit unfamiliar state transitions where the prediction error is high (high mismatch between the agent's expectation and true experience). Another curiosity-driven technique is Variational Information Maximizing Exploration (VIME) [136], which pushes the agent to explore states that lead to a larger change in the dynamics model (higher information gain).

Moreover, this subcategory includes techniques that estimate the state novelty within a feature space designed to capture the temporal or dynamical aspects of states. For instance, Exploration via Elliptical Episodic Bonuses (E3B) [165] and RIDE [160] both utilize an inverse dynamics model to learn state embeddings that represent the controllable dynamics of the environment. While RIDE encourages the agent to select actions that produce substantial changes in the state embedding, E3B applies an elliptical episodic bonus to guide exploration. Additional examples include Never Give Up (NGU) [134], Agent 57 [256], and Episodic Curiosity (EC) [257], all of which employ memory-based methods using distance-based metrics in a dynamics-aware feature space to approximate State + Dynamics novelty. Similarly, [258] propose the LIBERTY approach, which utilizes an inverse dynamic bisimulation metric to measure distances between states in a latent space, ensuring effective exploration and policy invariance. The work of [259] also presents a novel behavioral metric with Cyclic Dynamics (BCD), leveraging successor features and vector quantization to evaluate behavioral similarity between states and capture interrelations among environmental dynamics. Finally, [260] propose using the inverse of the norm of the successor representation (SR) as an intrinsic reward to account for transition dynamics. More recently, [261] developed the SPIE approach (Successor-Predecessor Intrinsic Exploration), which constructs an intrinsic reward by integrating both prospective and retrospective information from previous trajectories, also based on SR.

A.1.2 *"How to Explore?"*

Policy/Action level diversity:

Algorithms in this subcategory aim to explore diverse actions from the same state. What makes it different from the State + Dynamics algorithms introduced in A.1.1 is that the previous category uses knowledge about the states and dynamics of the environment and pushes for exploring the areas where the agent knows the least (high uncertainty). In contrast, this level of diversity considers the previous exploration behavior represented by the policy (how the agent has explored) and pushes it to explore differently, inducing diversity in the policy learned. For example, in Maximum Entropy RL (Max Entropy), the aim is to learn the optimal behavior while acting as randomly as possible. The objective function becomes the sum of expected rewards and conditional action entropy [262]. Soft Actor-Critic (SAC) [167] is a popular RL algorithm implementing the Max Entropy RL framework. Diversity-driven exploration strategy [263] is another exploration technique that encourages the agent to behave differently in similar states. It maximizes the divergence between the current policy and prior policies. Similarly, Adversarially Guided Actor-Critic (AGAC) [264] maximizes the divergence between the prediction of the policy and an adversary policy trained to mimic the behavioral policy. The main motivation is to encourage the policy to explore different behaviors by remaining different from the adversary. Another branch that belongs to this diversity level is population-based exploration, which combines evolutionary strategies with Reinforcement Learning. These approaches train a population of agents to learn diverse behaviors that are high scoring at the same time, in order to effectively explore the environment [265, 266]. For more details on the connection between evolutionary approaches and RL, please refer to the comprehensive survey by [267].

Skill level diversity:

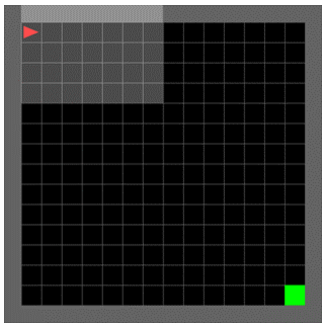
Skill level diversity disentangles diverse behaviors into different latent-conditioned policies (also called skills). The policy π is conditioned on a latent variable $z \sim p(z)$, and each z defines a different policy denoted by $\pi(a|s,z)$ [139]. This category aims to discover diverse skills, and the intrinsic reward is a function of the skill. Most methods falling into this category come from the domain of unsupervised skill discovery and use a discriminator-based architecture, such as Diversity is All You Need (DIAYN) [138]. DIAYN replaces the task reward with a learned discriminator term $q_\alpha(z|s)$ that infers the behavior from the current state in order to generate diverse policies visiting different sets of states. It also uses the Max Entropy RL framework to learn skills that are as random as possible [138]. Maximum Entropy Diverse Exploration (MEDE) [139] is very similar to "DIAYN + extrinsic reward", with the small difference of conditioning the discriminator on the state-action pair $q_\alpha(z|s,a)$ instead of the state only. Moreover, MEDE uses the discriminator term as a prior in the objective function instead of adding it as an intrinsic reward. Structured Max Entropy RL (SMERL) is another algorithm with the same approach as DIAYN, but it adds the intrinsic reward to the task reward only when the policies have achieved at least near-optimal returns [143]. DOMINO (Diversity Optimization Maintaining Near Optimally) also learns diverse policies while remaining near-optimal; it uses an intrinsic reward that maximizes the diversity of policies by measuring the distance between the expected features of the policies' state-action occupancies [144]. It is important to mention that skills in the literature can be called options or goals. For example, Variational Intrinsic Control (VIC) is an algorithm that provides the agent with an intrinsic reward that relies on modeling options and learning policies conditioned on these options [140]. Instead of sampling options from a fixed prior distribution as in DIAYN, VIC learns the prior distribution of options and updates it to choose options that yield higher rewards [140]. Both DIAYN and VIC are part of goal-conditioned RL methods, where goals are internally generated by agents and achieved via self-generated rewards [129]. More recent unsupervised skill-learning methods have emerged, such as [173] which proposed Behavior Contrastive Learning (BeCL), a novel competence-based method that uses contrastive learning to encourage similar behaviors within the same skill and diverse behaviors across different skills. This is done by maximizing the mutual information (MI) between different states generated by the same skill as an intrinsic reward. Another recent work by [175] proposed skill discovery with guidance (DISCO-DANCE) which identifies the guide skill most likely to reach unexplored states, directs other skills to follow it, and disperses them to maximize distinctiveness. Moreover, [174] proposed Learning Diverse Skills through Successor States (LEADS), which maximizes a variant of the MI between skills and states, by leveraging the successor state measure to tailor skills toward less-visited states while also maximizing the disparity between skills.

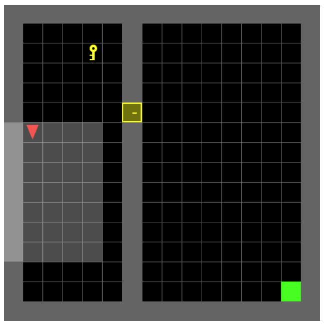
A.2 MiniGrid Environments

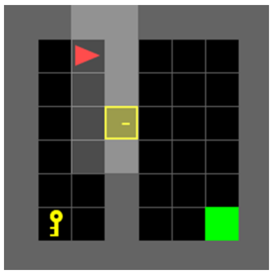
We use the following MiniGrid environments shown in Figure A.2:

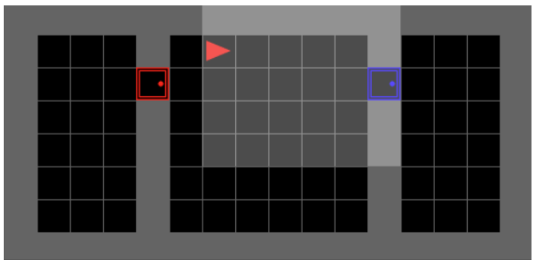
1. Empty: This is an empty grid where the agent is always placed in the corner opposite to the goal. The task is to get to the green goal square. We use the regular variant “MiniGrid-Empty-16x16-v0”.

2. DoorKey: This is a sparse reward environment that requires a specific order of visiting the states to solve the task; the agent needs to pick up the key, open the door, then get to the green goal square. It does not receive any reward after picking up the key or unlocking the door; it is rewarded only at the end of the task. We use “MiniGrid-DoorKey-16x16-v0” in the case of grid encodings and “MiniGrid-DoorKey-8x8-v0” in the case of RGB observations.


(a) Empty 16x16


(b) DoorKey 16x16


(c) DoorKey 8x8


(d) RedBlueDoors 8x8

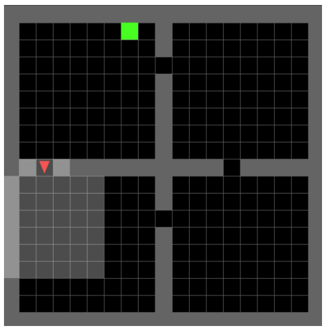

(e) FourRooms

Figure A.2: MiniGrid environments.

3. FourRooms: In this environment, the agent must navigate a maze consisting of four rooms, with both its initial position and goal position being randomized. We use "MiniGrid-FourRooms-v0" where each of the four rooms consists of a grid of size 8×8.

4. RedBlueDoors: The agent is randomly placed in a room where there are one red and one blue door facing opposite directions. The task consists of opening the red door before opening the blue door. The agent must rely on its memory of whether it has previously opened the other door to successfully complete the task, as it cannot see the door behind it. We use "MiniGrid-RedBlueDoors-8x8-v0".

For all tasks, a maximum number of steps t_{max} is assigned, to encourage the agent to solve the task as quickly as possible. When the agent succeeds after t steps, it receives a reward $r_t = 1 - 0.9t/t_{max}$ in all three environments. The episode ends when the agent collects the final reward or when the maximum number of steps is exceeded.

Observation and Action spaces. The observations are egocentric and partially observable. We first considered the grid encoding observations of size $7\times7\times3$. The first two dimensions (7×7) compose the tile set, and the last dimension encodes the object type (wall, door, $\cdots$), the object color (red, green, $\cdots$) and the object status (door open, door closed, door locked). Specifically, object type $\in \{0,1,2,3,4,5,6,7,8,9,10\}$, object color $\in \{0,1,2,3,4,5\}$, and object status $\in \{0,1,2\}$. Then, we used partial RGB visual observations of size $56\times56\times3$ (7 tiles of 8×8 pixels each) to increase the complexity of the task, as agents must extract features directly from the images. There are 7 actions available to the agent: turn left, turn right, move forward, pick up an object, drop an object, toggle and done. Some of these actions are unused in certain tasks.

A.3 Hyperparameters

For State Count and ICM, we use the hyperparameters from the previous study [150]. Since Max Entropy + PPO and DIAYN were not tested before on MiniGrid, we run a grid search over $\beta^{int} \in [0.1, 0.01, 0.001, 0.0005]$ and pick the best values of β^{int} that result in the highest return during training. The chosen values of β^{int} are summarized in Table A.1. For DIAYN, we choose to train 10 skills, which is the number used in the study by [268], and we use a discriminator learning rate of $3\times e^{-4}$ following the implementation of the DIAYN paper [138] (Table A.2). Note that we reused the same hyperparameters for the second part, where we tested on RGB observations.

Table A.1: Best intrinsic reward coefficients β^{int}.

	Empty	DoorKey	RedBlueDoors	FourRooms
State Count	0.005	0.005	0.005	0.005
Max Entropy	0.0005	0.0005	0.0005	0.0005
ICM	0.05	0.05	0.05	0.05
DIAYN	0.01	0.01	0.01	0.01

Table A.2: List of hyperparameters.

Number of parallel actors	16
Number of frames per rollout	128
Number of epochs	4
Batch size	256
Discount γ	0.99
Learning rate	0.0001
λ_{gae}	0.95
Entropy regularization coefficient	0.0005
Value loss coefficient	0.5
Clipping factor PPO	0.2
Gradient clipping	0.5
Forward dynamics loss coefficient	10
Inverse dynamics loss coefficient	0.1
Learning rates (state embedding, forward, and inverse dynamics)	0.0001
Number of skills	10
Discriminator learning rate	0.0003
SimHash key size	16

A.4 Additional Experimental Results

In this section, we present supplementary analyses to complement the main results. Specifically, we provide **reward discovery statistics** and **state visitation heatmaps** across multiple environments and two observation spaces (grid encoding and RGB). The tables report the frame number at which each exploration method discovers the reward for the first, second, and third time. Results are averaged across five independent runs, with mean and standard deviation reported as $\mu \pm \sigma$. If the reward is never found, the frame number is set to the training budget (40M). The *heatmaps* visualize state visitation counts accumulated during 10M frames of training on singleton environments. For each intrinsic reward method, snapshots are taken at three representative points in training: **T1** = 100K frames, **T2** = 500K frames, and **T3** = 10M frames. Color intensity indicates the proportion of frames spent in each state, with high values capped to enhance visibility. Results are organized by observation space: Section A.4.1 presents experiments with **grid encoding observations**, while Section A.4.2 reports results with **RGB observations**.

A.4.1 Grid Encoding Observation Space

Table A.3: Frame number at which the reward is discovered for the first, second and third time in Empty 16x16 environment with grid encodings.

Empty 16x16	First reward	Second Reward	Third reward
PPO	15 452 ± 7112	21 273 ± 11539	28 304 ± 14785
PPO + State Count	18 428 ± 9119	25 340 ± 11537	32 483 ± 12888
PPO + Max Entropy	16 841 ± 6916	22 768 ± 6736	27 318 ± 10355
PPO + ICM	**8 918 ± 3565**	**13 436 ± 6830**	**18 281 ± 9467**
PPO + DIAYN pretraining	12 668 ± 7030	20 076 ± 13898	326 963 ± 662300
PPO + DIAYN finetuning	1 001 862 ± 1187338	1 130 208 ± 1082785	1 207 600 ± 1038924

Table A.4: Frame number at which the reward is discovered for the first, second and third time in DoorKey 16x16 environment with grid encodings.

DoorKey 16x16	First reward	Second Reward	Third reward
PPO	1 242 342 ± 529863	2 276 508 ± 917486	3 186 537 ± 1616226
PPO + State Count	**496 486 ± 550012**	**558 204 ± 548684**	**783 075 ± 615917**
PPO + Max Entropy	594 649 ± 696956	1 067 401 ± 743704	3 300 668 ± 3108002
PPO + ICM	1 089 286 ± 734419	1 287 632 ± 674758	1 683 612 ± 539173
PPO + DIAYN Pretraining	40 000 000 ± 0	40 000 000 ± 0	40 000 000 ± 0
PPO + DIAYN finetuning	2 087 398 ± 449537	2 221 756 ± 447746	2 516 739 ± 689340

Table A.5: Frame number at which the reward is discovered for the first, second and third time in RedBlueDoors environment with grid encodings.

RedBlueDoors	First reward	Second Reward	Third reward
PPO	13 136 ± 5647	**17 568 ± 8303**	26 553 ± 6733
PPO + State Count	13 180 ± 8236	25 923 ± 11537	33 545 ± 19115
PPO + Max Entropy	**9 417 ± 2678**	20 464 ± 10420	**24 432 ± 10339**
PPO + ICM	37 721 ± 68636	129 043 ± 175 507	162 060 ± 193005
PPO + DIAYN pretraining	19 244 ± 10004	24 611 ± 12661	39 280 ± 25334
PPO + DIAYN finetuning	2 992 614 ± 2118551	3 006 659 ± 2114575	3 033 043 ± 2096962

Table A.6: Frame number at which the reward is discovered for the first, second and third time in FourRooms environment with grid encodings.

FourRooms	First reward	Second Reward	Third reward
PPO	25 222 ± 32606	97 033 ± 41446	150 188 ± 104821
PPO + State Count	15 465 ± 9712	34 649 ± 11090	51 820 ± 23054
PPO + Max Entropy	2 479 424 ± 5498212	5 327 913 ± 5306632	6 874 905 ± 5056693
PPO + ICM	89 433 ± 111832	197 312 ± 171435	274 883 ± 171782
PPO + DIAYN pretraining	2 089 ± 1178	9 049 ± 5350	14 531 ± 9099
PPO + DIAYN finetuning	29 238 ± 27103	41 376 ± 35912	69 737 ± 53656

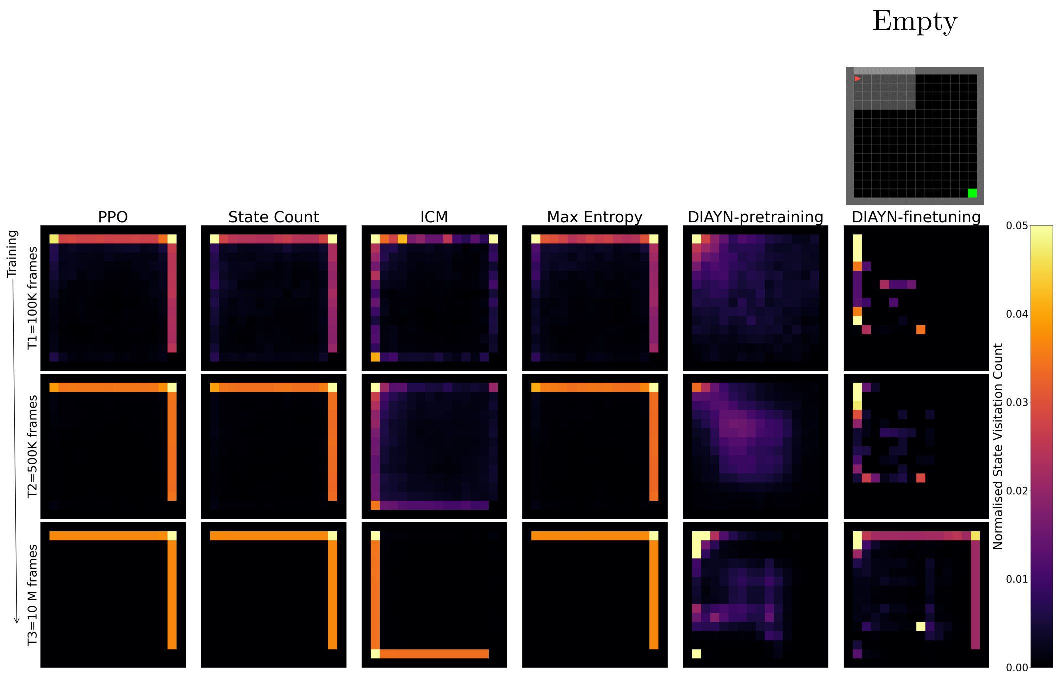


Figure A.3: State visitation heatmap for singleton Empty 16x16 environment with grid encoding observations.

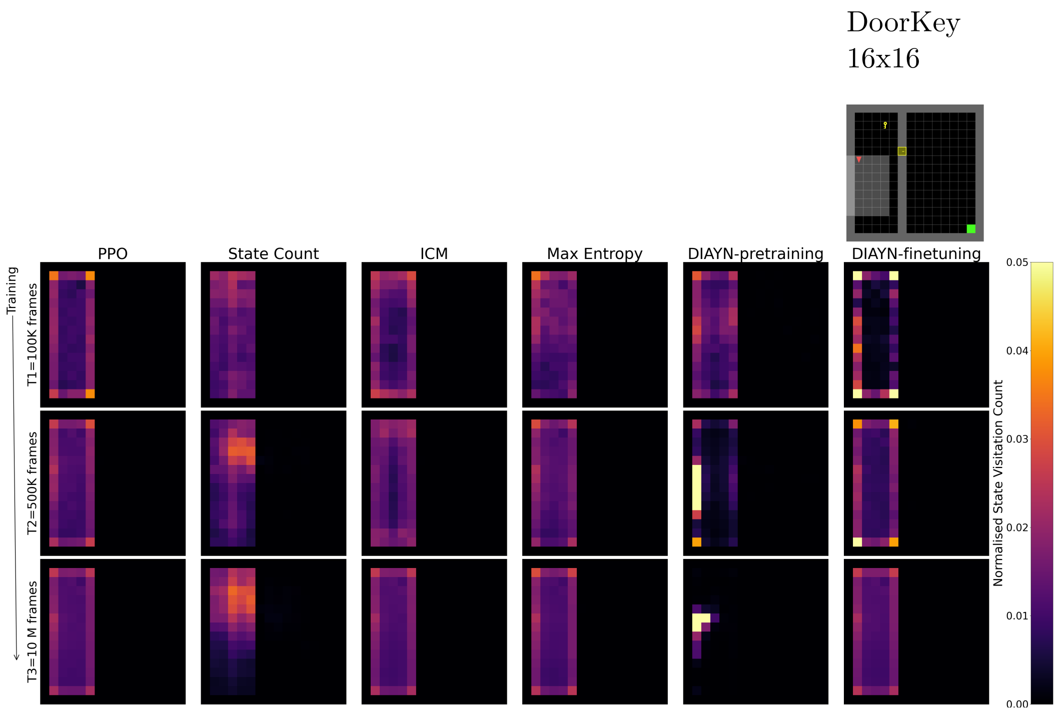


Figure A.4: State visitation heatmap for singleton DoorKey 16x16 environment with grid encoding observations.

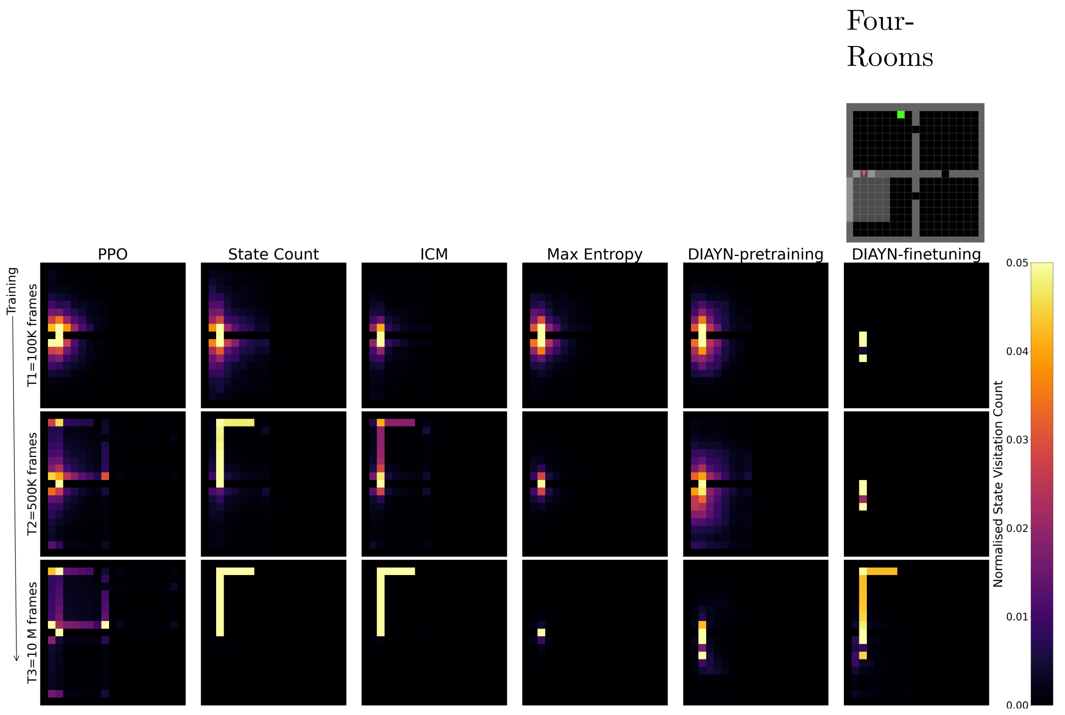


Figure A.5: State visitation heatmap for singleton FourRooms environment with grid encoding observations.

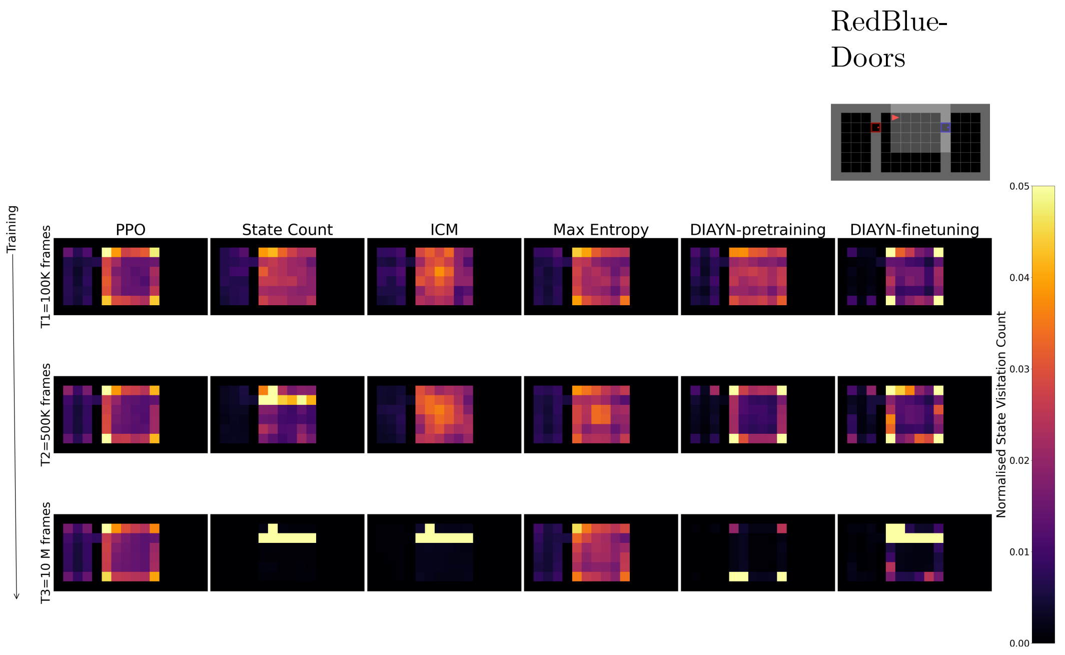


Figure A.6: State visitation heatmap for singleton RedBlueDoors environment with grid encoding observations.

A.4.2 RGB Observation Space

Table A.7: Frame number at which the reward is discovered for the first, second, and third time in Empty 16x16 environment with RGB observations.

Empty 16x16 RGB	First reward	Second Reward	Third reward
PPO	48 256 ± 85183	103 401 ± 112583	115 148 ± 117261
PPO + SimHash	71 132 ± 9119	75 686 ± 95328	79 699 ± 96431
PPO + Max Entropy	**43 072 ± 76270**	**49 033 ± 79868**	**59 494 ± 81361**
PPO + ICM	448 121 ± 500641	493 401 ± 522557	509 750 ± 509223
PPO + DIAYN pretraining	896 963 ± 1257009	2 262 649 ± 3635427	2 267 164 ± 3639831
PPO + DIAYN finetuning	69 712 ± 102945	94 240 ± 138306	110 710 ± 135551

Table A.8: Frame number at which the reward is discovered for the first, second, and third time in DoorKey 8x8 environment with RGB observations.

DoorKey 8x8 RGB	First reward	Second Reward	Third reward
PPO	31 430 ± 39987	**49 257 ± 44984**	**75 587 ± 32508**
PPO + SimHash	93 494 ± 75207	115 529 ± 80404	143 840 ± 71717
PPO + Max Entropy	**26 870 ± 34213**	100 931 ± 96891	124 828 ± 114469
PPO + ICM	445 222 ± 433991	655 200 ± 384520	713 795 ± 381101
PPO + DIAYN pretraining	32 003 513 ± 17880687	40 000 000 ± 0	40 000 000 ± 0
PPO + DIAYN finetuning	40 000 000 ± 0	40 000 000 ± 0	40 000 000 ± 0

Table A.9: Frame number at which the reward is discovered for the first, second, and third time in RedBlueDoors environment with RGB observations.

RedBlueDoors RGB	First reward	Second Reward	Third reward
PPO	18 504 ± 12321	**28 179 ± 19650**	**44 342 ± 36719**
PPO + SimHash	35 516 ± 38700	49 548 ± 48897	60 643 ± 58630
PPO + Max Entropy	51 871 ± 51587	71 776 ± 59120	97 907 ± 88967
PPO + ICM	**18 355 ± 26236**	206 547 ± 420774	219 718 ± 419780
PPO + DIAYN pretraining	16 012 892 ± 21897134	16 015 049 ± 21895167	24 011 241 ± 21893513
PPO + DIAYN finetuning	24 212 716 ± 21618947	24 212 716 ± 21618947	24 219 180 ± 21610106

Table A.10: Frame number at which the reward is discovered for the first, second, and third time in FourRooms environment with RGB observations.

FourRooms RGB	First reward	Second Reward	Third reward
PPO	6 057 ± 7369	27 203 ± 34679	41 766 ± 47238
PPO + SimHash	7 561 ± 13820	13 171 ± 11599	20 406 ± 17137
PPO + Max Entropy	7 654 ± 13591	9 014 ± 13184	13 907 ± 12435
PPO + ICM	7 491 ± 10928	10 060 ± 12737	16 912 ± 16744
PPO + DIAYN pretraining	3 276 505 ± 7322741	3 290 252 ± 7342741	3 296 742 ± 7343739
PPO + DIAYN finetuning	**4 470 ± 2384**	**6 854 ± 4956**	**8 166 ± 4735**

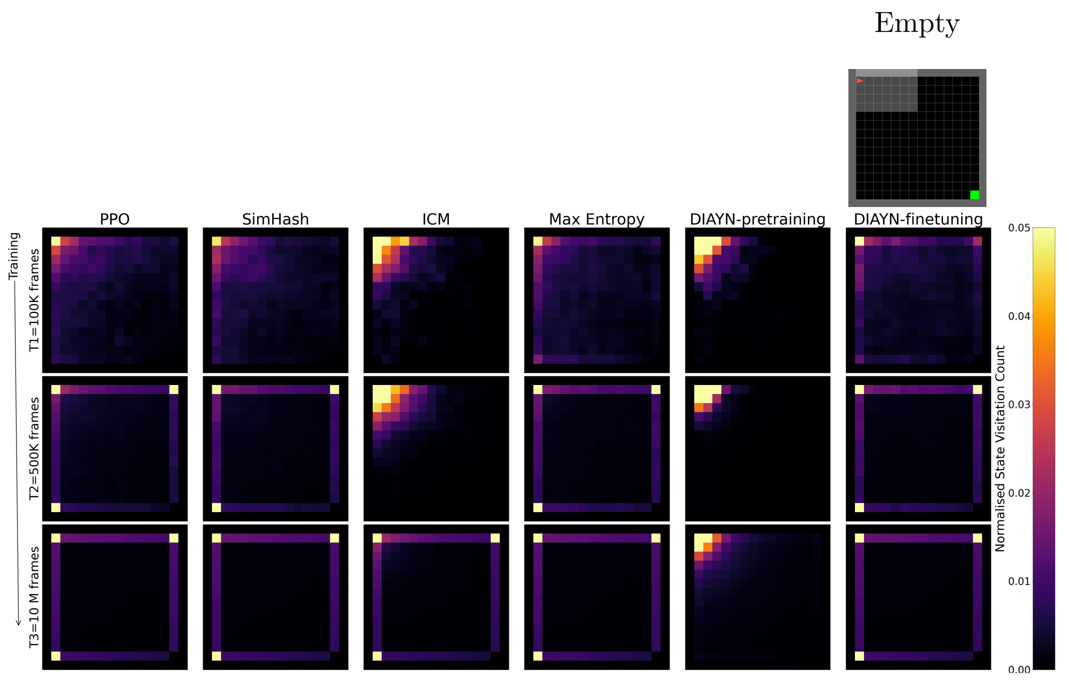


Figure A.7: State visitation heatmap for singleton Empty 16x16 environment with RGB observations.

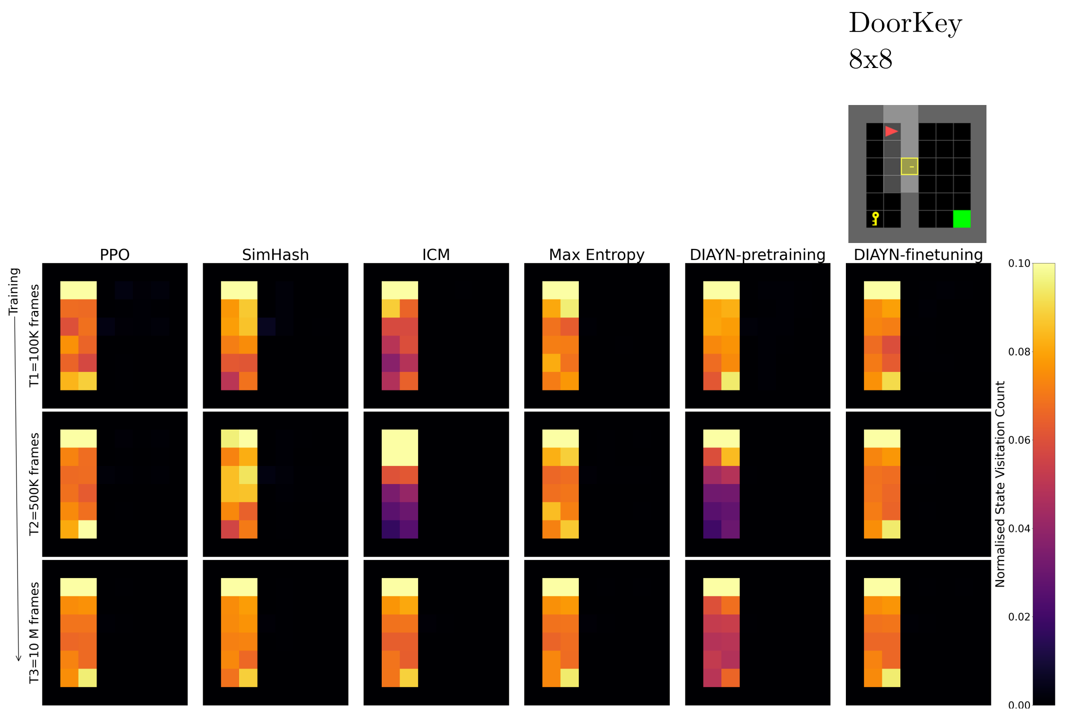


Figure A.8: State visitation heatmap for singleton DoorKey 8x8 environment with RGB observations.

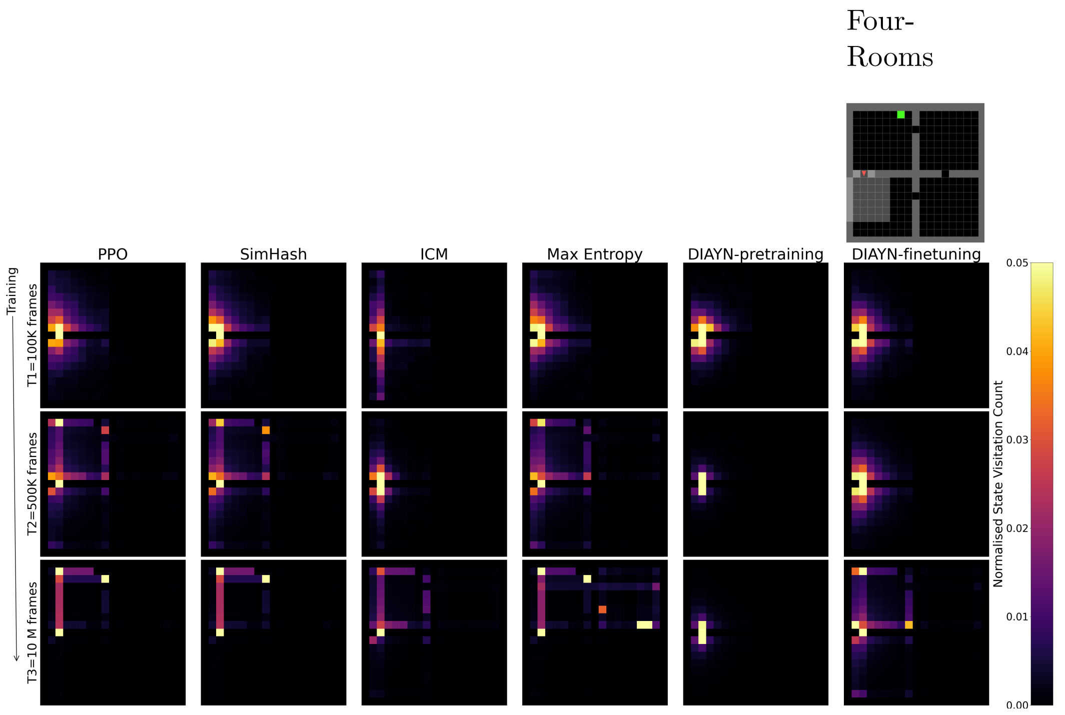


Figure A.9: State visitation heatmap for singleton FourRooms environment with RGB observations.

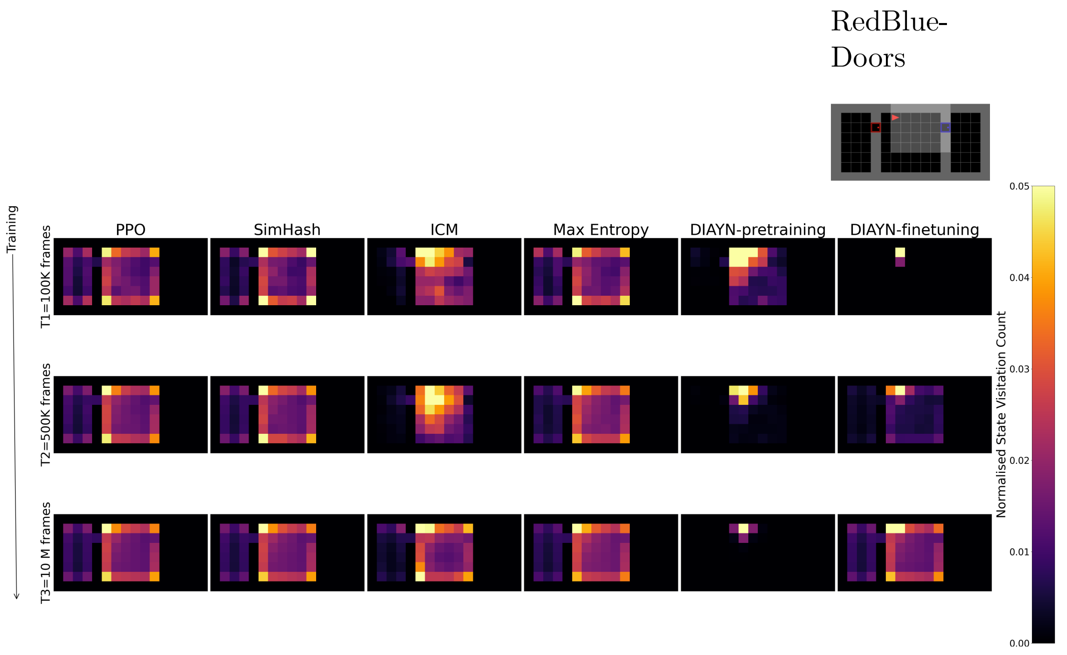


Figure A.10: State visitation heatmap for singleton RedBlueDoors environment with RGB observations.

A.5 DIAYN Extrinsic

Initially, we evaluated DIAYN combined with extrinsic rewards, but it did not perform well because of the imbalance between discriminability and reward maximization (see Figure A.11). Recognizing that DIAYN is primarily intended for unsupervised pretraining of skills rather than simultaneous use with return maximization, we decided to split the training budget between pretraining and finetuning.

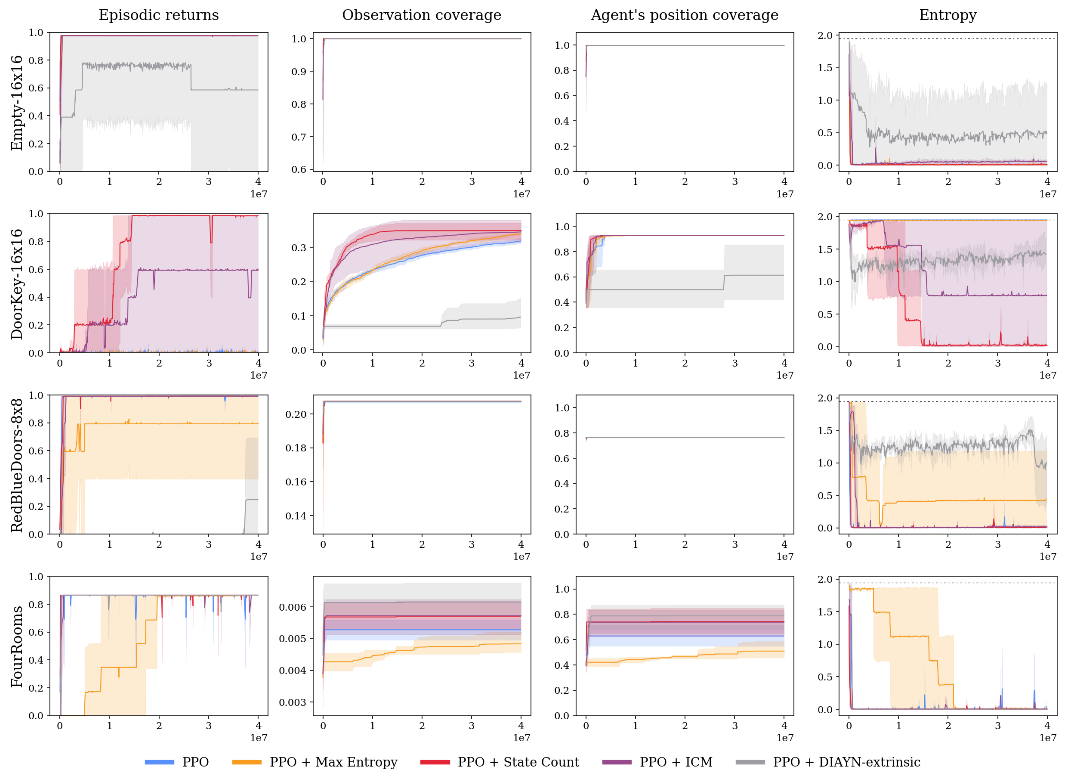


Figure A.11: Analogous to Figure. 4.3, but showing DIAYN combined with extrinsic rewards on grid encoding observations. Metrics are plotted against the number frames and averaged over five seeds.

Appendix B

Appendix of Chapter 5

B.1 Proof of Theorem 1 and Corollary 1

For the proof of Theorem 1, we leverage the fact that V belongs to an RKHS. Specifically, we use the Mercer representation of V

$$V(s) = \sum_{m=1}^{\infty} w_m \lambda_m^{\frac{1}{2}} \psi_m(s). \tag{B.1}$$

We can also rewrite the observations in the observation vector $\boldsymbol{y}_n$ as the sum of a noise term and the expected value of the observation (noise free part).

$$V(s_i') = \underbrace{(V(s_i') - f(z_i))}_{\text{Observation noise}} + \underbrace{f(z_i)}_{\text{Noise-free observation}} \tag{B.2}$$

Using the notation $\overline{\psi}_m(z) = \mathbb{E}_{s' \sim P(\cdot|z)} \psi_m(s')$, we can rewrite $f(z_i)$ as follows

$$\begin{aligned} f(z_i) &= \mathbb{E}_{s \sim P(\cdot|z_i)}[V(s)] \\ &= \mathbb{E}_{s \sim P(\cdot|z_i)} \left[\sum_{m=1}^{\infty} w_m \lambda_m^{\frac{1}{2}} \psi_m(s) \right] \\ &= \sum_{m=1}^{\infty} w_m \lambda_m^{\frac{1}{2}} \mathbb{E}_{s \sim P(\cdot|z_i)}[\psi_m(s)] \\ &= \sum_{m=1}^{\infty} w_m \lambda_m^{\frac{1}{2}} \overline{\psi}_m(z_i) \end{aligned} \tag{B.3}$$

We then use the following notations, $\varepsilon_i = V(s'_i) - f(z_i)$, $\boldsymbol{\varepsilon}_n = [\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_n]^\top$, $\boldsymbol{f}_n = [f(z_1), f(z_2), \cdots, f(z_n)]^\top$, to rewrite the prediction error

$$\begin{aligned} f(z) - \hat{f}_n(z) &= f(z) - k_n^\top(z)(\tau^2 I + K_n)^{-1}\boldsymbol{y}_n \\ &= f(z) - k_n^\top(z)(\tau^2 I + K_n)^{-1}(\boldsymbol{\varepsilon}_n + \boldsymbol{f}_n) \\ &= \underbrace{f(z) - k_n^\top(z)(\tau^2 I + K_n)^{-1}\boldsymbol{f}_n}_{\text{Prediction error from noise-free observations}} \underbrace{- k_n^\top(z)(\tau^2 I + K_n)^{-1}\boldsymbol{\varepsilon}_n}_{\text{The error due to noise}} \end{aligned}$$

The first term is deterministic (not random) and can be bounded following the standard approaches in kernel-based models, for example using the following result from [70]. Let us use the notations $\boldsymbol{\zeta}_n(z) = k_n^\top(z)(\tau^2 I + K_n)^{-1}$ and $\zeta_i(z) = [\boldsymbol{\zeta}_n(z)]_i$.

Lemma 1 (Proposition 1 in [70])**.** *We have*

$$\sigma_n^2(z) = \sup_{f:\|f\|_{\mathcal{H}} \le 1} (f(z) - \boldsymbol{\zeta}_n^\top(z)\boldsymbol{f}_n)^2 + \tau^2 \|\boldsymbol{\zeta}_n(z)\|_{\ell^2}^2.$$

Based on this lemma, the first term can be deterministically bounded by $B_1\sigma_n(z)$:

$$|f(z) - k_n^\top(z)(\tau^2 I + K_n)^{-1}\boldsymbol{f}_n| \le B_1 \sigma_n(z) \tag{B.4}$$

We next bound the second term, the error due to noise.

$$\begin{aligned} k_n^\top(z)(\tau^2 I + K_n)^{-1}\boldsymbol{\varepsilon}_n &= \sum_{i=1}^n \zeta_i(z)\varepsilon_i \\ &= \sum_{i=1}^n \zeta_i(z)\Big(\sum_{m=1}^\infty w_m \lambda_m^{\frac{1}{2}} \psi_m(s'_i) - \sum_{m=1}^\infty w_m \lambda_m^{\frac{1}{2}} \overline{\psi}_m(z_i)\Big) \\ &= \sum_{m=1}^\infty w_m \lambda_m^{\frac{1}{2}} \sum_{i=1}^n \zeta_i(z)(\psi_m(s'_i) - \overline{\psi}_m(z_i)) \\ &= \sum_{m=1}^M w_m \lambda_m^{\frac{1}{2}} \sum_{i=1}^n \zeta_i(z)(\psi_m(s'_i) - \overline{\psi}_m(z_i)) \\ &\quad + \sum_{m=M+1}^\infty w_m \lambda_m^{\frac{1}{2}} \sum_{i=1}^n \zeta_i(z)(\psi_m(s'_i) - \overline{\psi}_m(z_i)) \end{aligned}$$

We note that $\psi_m(s_i') - \overline{\psi}_m(z_i)$ are bounded random variables with a range of $2\psi_{\max}$. Using Chernoff-Hoeffding inequality and the bound on the norm of $\boldsymbol{\zeta}_n$ provided in Lemma 1, we have that with probability at least $1-\delta/M$

$$\sum_{i=1}^{n}\zeta_i(z)(\psi_m(s_i') - \overline{\psi}_m(z_i)) \le \frac{\psi_{\max}\sigma_n(z)}{\tau}\sqrt{2\log\left(\frac{M}{\delta}\right)}.$$

Using a probability union bound, with probability $1-\delta$

$$\begin{aligned}
&\sum_{m=1}^{M} w_m\lambda_m^{\frac{1}{2}}\sum_{i=1}^{n}\zeta_i(z)(\psi_m(s_i') - \overline{\psi}_m(z_i))\\
&\le \sum_{m=1}^{M} w_m\lambda_m^{\frac{1}{2}}\frac{\psi_{\max}\sigma_n(z)}{\tau}\sqrt{2\log\left(\frac{M}{\delta}\right)}\\
&\le \left(\sum_{m=1}^{M}\lambda_m\right)^{\frac{1}{2}}\left(\sum_{m=1}^{M}w_m^2\right)^{\frac{1}{2}}\frac{\psi_{\max}\sigma_n(z)}{\tau}\sqrt{2\log\left(\frac{M}{\delta}\right)}\\
&\le \left(\sum_{m=1}^{M}\lambda_m\right)^{\frac{1}{2}}B_2\frac{\psi_{\max}\sigma_n(z)}{\tau}\sqrt{2\log\left(\frac{M}{\delta}\right)}\\
&\le CB_2\frac{\psi_{\max}\sigma_n(z)}{\tau}\sqrt{2\log\left(\frac{M}{\delta}\right)}
\end{aligned}$$

The second inequality is based on the Cauchy-Schwarz inequality. In the third inequality, we used that B_2 is the upper bound on the RKHS norm of V. In the last inequality, we used the observation that under both polynomial eigenvalue decay with $p>1$ and exponential eigendecay, the sum of the eigenvalues is bounded by an absolute constant C.

Also, for the second term, we have

$$\begin{aligned}
&\sum_{m=M+1}^{\infty} w_m\lambda_m^{\frac{1}{2}}\sum_{i=1}^{n}\zeta_i(z)\left(\psi_m(s_i') - \overline{\psi}_m(z_i)\right)\\
&\le 2\psi_{\max}\sum_{m=M+1}^{\infty} w_m\lambda_m^{\frac{1}{2}}\sum_{i=1}^{n}\zeta_i(z)\\
&\le 2\psi_{\max}\sum_{m=M+1}^{\infty} w_m\lambda_m^{\frac{1}{2}}\left(n\sum_{i=1}^{n}\zeta_i^2(z)\right)^{\frac{1}{2}}\\
&\le \frac{2\sigma_n(z)\psi_{\max}\sqrt{n}}{\tau}\sum_{m=M+1}^{\infty} w_m\lambda_m^{\frac{1}{2}}
\end{aligned}$$

$$\leq \frac{2\sigma_n(z)\psi_{\max}\sqrt{n}}{\tau}\left(\left(\sum_{m=M+1}^{\infty} w_m^2\right)\left(\sum_{m=M+1}^{\infty}\lambda_m\right)\right)^{\frac{1}{2}}$$
$$\leq \frac{2B_2\sigma_n(z)\psi_{\max}}{\tau}\left(n\sum_{m=M+1}^{\infty}\lambda_m\right)^{\frac{1}{2}}.$$

The first inequality holds by definition of $\psi_{\max}$. The second inequality is based on the Cauchy-Schwarz inequality. The third inequality uses Lemma 1. The fourth inequality utilizes the Cauchy-Schwarz inequality again, and the last inequality results from the upper bound on the RKHS norm of V.

Putting together, with probability $1-\delta$,

$$k_n^\top(z)(\tau^2 I+K_n)^{-1}\varepsilon_n \leq \frac{CB_2\psi_{\max}\sigma_n(z)}{\tau}\sqrt{2\log\left(\frac{M}{\delta}\right)} + \frac{2B_2\sigma_n(z)\psi_{\max}}{\tau}\sqrt{n\sum_{m=M+1}^{\infty}\lambda_m}. \tag{B.5}$$

Proof of Corollary 1 To extend the confidence interval given in Theorem 1 to hold uniformly on $\mathcal{Z}$, we use a discretization argument. For this purpose, we apply Assumption 2 to f and $\hat{f}_n$, and also use Assumption 2 to bound the discrimination error in σ_n. The following lemma provides a high probability bound on $\|\hat{f}_n\|_{k_\varphi}$.

Lemma 2. *For function f defined in Theorem 1, the RKHS norm of $\hat{f}_n$ satisfies the following with probability at least $1-\delta$:*

$$\|\hat{f}_n\|_{\mathcal{H}_{k_\varphi}} \leq B_1 + \frac{v_{\max}}{\tau}\sqrt{2\left(\Gamma_{k_\varphi}(n)+1+\log\left(\frac{1}{\delta}\right)\right)}. \tag{B.6}$$

For a proof see Lemma 5 in [181].

Let $B_3(\delta) = B_1 + \frac{v_{\max}}{\tau}\sqrt{2(\Gamma_{k_\varphi}(n)+1+\log(\frac{1}{\delta}))}$ denote the $1-\delta$ upper confidence bound on $\|\hat{f}_n\|_{H_{k_\varphi}}$. Let $\mathbb{Z}$ be the discretization of $\mathcal{Z}$ specified in Assumption 2 with RKHS norm bound $B_3(\frac{\delta}{2})$. That is for any $g\in\mathcal{H}_{k_\varphi}$ with $\|g\|_{\mathcal{H}_{k_\varphi}} \leq B_3(\frac{\delta}{2})$, we have $g(z)-g([z]) \leq \frac{1}{n}$, where $[z] = \arg\min_{z'\in\mathbb{Z}}\|z'-z\|$ is the closest point in $\mathbb{Z}$ to z, and $|\mathbb{Z}| \leq c_n$, where $c_n = c(B_3(\frac{\delta}{2}))^d n^d$. Applying

Assumption 2 to f and $\hat{f}_n$ with this discretization, it holds for all $z \in \mathcal{Z}$ that

$$|f(z) - f([z])| \leq \frac{1}{n}. \tag{B.7}$$

In addition, by Lemma 2, with probability eat least $1-\delta$

$$|\hat{f}_n(z) - \hat{f}_n([z]) \leq \frac{1}{n}. \tag{B.8}$$

Furthermore, we have the following lemma, which can roughly be viewed as a Lipschitz continuity property for σ_n.

Lemma 3. *Under Assumption 2, with the discrimination $\mathbb{Z}$ described above, it holds for all $z \in \mathcal{Z}$ that*

$$\sigma_n(z) - \sigma_n([z]) \leq \frac{2}{\sqrt{n}}.$$

Proof of Lemma 3. Using the reproducing property of RKHS, we have

$$\|k_n^\top(z)(K_n + \tau^2 I)^{-1} k_n(\cdot)\|_{\mathcal{H}_k} \leq \frac{k_{\max}\sqrt{n}}{\tau}, \tag{B.9}$$

where $k_{\max}$ is the maximum value of the kernel. Let us define $q(\cdot, \cdot') = k_n^\top(\cdot)(K_n + \tau^2 I)^{-1} k_n(\cdot')$. We can write

$$\begin{aligned}
&|\sigma_n^2(z) - \sigma_n^2([z])| \\
&= \Big|(k(z,z) - q(z,z)) - (k([z],[z]) - q([z],[z]))\Big| \\
&= \Big|(k(z,z) - q(z,z)) - (k(z,[z]) - q(z,[z])) + (k(z,[z]) - q(z,[z])) - (k([z],[z]) - q([z],[z]))\Big| \\
&\leq |k(z,z) - k(z,[z])| + |k(z,[z]) - k([z],[z])| + |q(z,z) - q(z,[z])| + |q(z,[z]) - q([z],[z])| \\
&\leq \frac{4}{n}.
\end{aligned}$$

To obtain a discretization error bound for the standard deviation from that of the variance, we write

$$\begin{aligned}(\sigma_n(z)-\sigma([z]))^2 &\leq |\sigma_n(z)-\sigma([z])|\,(\sigma_n(z)+\sigma([z]))\\ &= |\sigma_n^2(z)-\sigma^2([z])|\\ &\leq \frac{4}{n}.\end{aligned}$$

Therefore,

$$|\sigma_n(z)-\sigma([z])| \leq \frac{2}{\sqrt{n}}.$$

□

Applying a probability union bound on the discretization $\mathbb{Z}$ to Theorem 1, and considering the error bounds in (B.7), (B.8) and Lemma 3, we arrive at Corollary 1.

B.2 Proof of Theorem 2

First, we define the following high-probability event :

$$\mathcal{E} = \left\{\forall h \in [H], |\hat{g}_h(z) - [P_h V_{h+1}](z)| \leq \beta(\delta)\left(\sigma_{h,N}(z) + \frac{2}{\sqrt{n}}\right) + \frac{2}{n}\right\}, \quad \text{(B.10)}$$

where $\beta(\delta) = \mathcal{O}\left(\frac{H}{\tau}\sqrt{d\log(\frac{NH}{\delta})}\right)$ as specified in Corollary 1 with $B_1 = \mathcal{O}(H)$ and $B_2 = c_v$. Using Corollary 1, we have $\mathbb{P}[\mathcal{E}] \geq 1-\delta$.

We divide the rest of the analysis into several steps, as outlined next.

Step 1: Under $\mathcal{E}$, with reward r, we bound $V_1^\star(s) - V_1^\pi(s)$ using $V_1(s) - V_1^\pi(s)$, based on the following lemma. Recall that V_h^π and $V_h^\star$ are the value functions of policy π and the optimal policy, respectively, and V_h is the proxy value functions used in Algorithm 7.

Lemma 4. *Under $\mathcal{E}$, we have*

$$V_h^\star(s) - V_h(s) \leq (H+1-h)(\frac{2\beta(\delta)}{\sqrt{N}} + \frac{2}{N}). \quad \text{(B.11)}$$

Proof of Lemma 4. The lemma is proven by induction over h, starting from

$V^{\star}_{H+1} = V_{H+1} = \mathbf{0}$. We have

$$\begin{aligned}
Q^{\star}_h(s,a) - Q_h(s,a) &= r_h(s,a) + [P_h V^{\star}_{h+1}](s,a) - r_h(s,a) - \hat{g}_h(s,a) - \beta(\delta)\sigma_{h,N}(s,a) \\
&\le [P_h V^{\star}_{h+1}](s,a) - [P_h V_{h+1}](s,a) + \frac{2\beta(\delta)}{\sqrt{N}} + \frac{2}{N} \\
&= [P_h(V^{\star}_{h+1} - V_{h+1})](s,a) + \frac{2\beta(\delta)}{\sqrt{N}} + \frac{2}{N} \\
&\le (H+1-h)(\frac{2\beta(\delta)}{\sqrt{N}} + \frac{2}{N})
\end{aligned}$$

The first inequality holds by $\mathcal{E}$, and the second inequality by induction assumption. Then, we have

$$\begin{aligned}
V^{\star}_h(s_h) - V_h(s_h) &= \max_{a\in\mathcal{A}} Q^{\star}_h(s,a) - \max_{a\in\mathcal{A}} Q_h(s,a) \\
&\le \max_{a\in\mathcal{A}} \{Q^{\star}_h(s,a) - Q_h(s,a)\} \\
&\le (H+1-h)(\frac{2\beta(\delta)}{\sqrt{N}} + \frac{2}{N}).
\end{aligned}$$

That proves the lemma.

□

Step 2: We also bound $V_1(s) - V^{\pi}_1(s)$ using the sum of standard deviations for the trajectory generated by the policy.

Lemma 5. *Under $\mathcal{E}$, we have*

$$V_1(s_1) - V^{\pi}_1(s_1) \le \mathbb{E}\left[\sum_{h=1}^{H} 2\beta(\delta)\sigma_{h,N}(s_h,a_h)\right] + \frac{2H\beta(\delta)}{\sqrt{N}} + \frac{2H}{N},$$

where the expectation is taken with respect to the trajectory generated by the policy.

Proof of Lemma 5. Note that $V_{H+1} = V^{\pi}_{H+1} = \mathbf{0}$. We next obtain a recursive relationship for the difference $V_h(s) - V^{\pi}_h(s)$.

$$\begin{aligned}
V_h(s_h) - V_h^{\pi}(s_h) &= Q_h\left(s_h, \pi(s_h)\right) - Q_h^{\pi}\left(s_h, \pi(s_h)\right) \\
&= r\left(s_h, \pi(s_h)\right) + \hat{g}_h\left(s_h, \pi(s_h)\right) + \beta(\delta)\sigma_{h,N}\left(s_h, \pi(s_h)\right) \\
&\quad - r\left(s_h, \pi(s_h)\right) - \left[P_h V_{h+1}^{\pi}\right]\left(s_h, \pi(s_h)\right) \\
&\leq \left[P_h V_{h+1}\right]\left(s_h, \pi(s_h)\right) + 2\beta(\delta)\sigma_{h,N}\left(s_h, \pi(s_h)\right) \\
&\quad + \frac{2\beta(\delta)}{\sqrt{N}} + \frac{2}{N} - \left[P_h V_{h+1}^{\pi}\right]\left(s_h, \pi(s_h)\right),
\end{aligned}$$

where the inequality is due to $\mathcal{E}$. Recursive application of the above inequality over $h = H, H-1, \cdots, 1$, we obtain

$$V_1(s_1) - V_1^{\pi}(s_1) \leq \mathbb{E}_{s_{h+1} \sim P(\cdot|s_h, \pi(s_h)), h<H}\left[\sum_{h=1}^{H} 2\beta(\delta)\sigma_{h,N}\left(s_h, \pi(s_h)\right)\right] + \frac{2H\beta(\delta)}{\sqrt{N}} + \frac{2H}{N}.$$

$\square$

Step 3: By definition, we have $V_1^{\pi}(s_1; \beta(\delta)\sigma_N) \leq V_1^{\star}(s_1; \beta(\delta)\sigma_N)$. Note that $V_1^{\pi}(s_1; \beta(\delta)\sigma_N) = \beta(\delta)\sum_{h=1}^{H}\sigma_{h,N}(s_h, \pi(s_h))$.

Step 4: We have $V_1^{\star}(s; \beta(\delta)\sigma_N) \leq V_1^{\star}(s; \beta(\delta)\sigma_n)$. This is due to the observation that $\sigma_{h,n}$ is decreasing in the number n of observations. We note that conditioning on observations only reduces the variance. That is seen from the positive definiteness of the Gram matrix and the formula for kernel ridge uncertainty estimator given in (5.5).

Step 5: Recall the selection rule in Algorithm 8: $s_{h,n}, a_{h,n} = \arg\max_{s,a}\sigma_{h,n-1}(s,a)$. When exploring with generative model, with this rule of selection, we have $V_1^{\star}(s_1; \beta(\delta)\sigma_{n-1}) \leq \beta(\delta)\sum_{h=1}^{H}\sigma_{h,n-1}(s_{h,n}, a_{h,n})$.

Step 6: Combining all previous steps, we conclude that, under $\mathcal{E}$,

$$V_1^{\star}(s) - V_1^{\pi}(s) \leq \frac{2\beta(\delta)}{N}\sum_{n=1}^{N}\sum_{h=1}^{H}\sigma_{h,n-1}(s_{h,n}, a_{h,n}) + \frac{4\beta(\delta)H}{\sqrt{N}} + \frac{4H}{N}. \tag{B.12}$$

Step 7: We bound the sum of standard deviations according to the following lemma that is a kernel based version of elliptical potential lemma [213].

Lemma 6. *For each h, we have*

$$\sum_{n=1}^{N} \sigma_{h,n-1}^2(s_{h,n}, a_{h,n}) \leq \frac{2\Gamma(N)}{\log(1+1/\tau^2)}. \tag{B.13}$$

See, e.g., [67] for a proof. Using Cauchy–Schwarz inequality, we obtain

$$\sum_{n=1}^{N} \sigma_{h,n-1}(s_{h,n}, a_{h,n}) \leq \sqrt{\frac{2N\Gamma(N)}{\log(1+1/\tau^2)}}.$$

Step 8: From Steps 6 and 7, we conclude that, π is an ϵ-optimal policy with ϵ no larger than

$$V_1^\star(s) - V_1^\pi(s) \leq 2H\beta(\delta)\sqrt{\frac{2\Gamma(N)}{N\log(1+1/\tau^2)}} + \frac{4\beta(\delta)H}{\sqrt{N}} + \frac{4H}{N}.$$

A simpler expression can be given as

$$V_1^\star(s) - V_1^\pi(s) = \mathcal{O}\left(H^2\sqrt{\frac{\Gamma(N)\log(NH/\delta)}{N}}\right).$$

Now, let N_0 be the smallest integer such that the right hand side less than ϵ. For any $N \geq N_0$ the suboptimality gap of the policy is at most ϵ. This completes the proof of Theorem 2.

B.3 Proof of Theorem 3

We define the event $\mathcal{E}$ similar to the proof of Theorem 2. The first 4 steps related to the planning phase are exactly the same as in the proof of Theorem 2. The rest of the proof is different and we will present it here.

In addition to $\mathcal{E}$, we define another high-probability event $\mathcal{E}'$ where all the confidence intervals utilized in the exploration hold true. Specifically, we define the following:

$$\mathcal{E}' = \{\forall n \in [N], \forall h \in [H], |\hat{f}_{h,n}(z) - f_{h,n}(z)| \le \beta(\delta)(\sigma_{h,n}(z) + \frac{2}{\sqrt{n}}) + \frac{2}{n}\}, \quad \text{(B.14)}$$

where $\beta(\delta) = \mathcal{O}\left(\frac{H}{\tau}\sqrt{d\log(\frac{NH}{\delta})}\right)$. Using Corollary 1, we have $\mathbb{P}[\mathcal{E}'] \ge 1-\delta$. The following steps are specific to the exploration without the generative model. We define a reward sequence using $\tilde{\sigma}_{h,n}^{h_0}$ such that $\tilde{\sigma}_{h,n}^{h_0} = \sigma_{h,n}$ when $h = h_0$ and $\tilde{\sigma}_{h,n}^{h_0} = 0$ for all $h \neq h_0$.

Step 5b: We have $V_1^\star(s;\beta(\delta)\sigma_n) \le \sum_{h_0=1}^{H} V_1^\star(s;\beta(\delta)\tilde{\sigma}_n^{h_0})$. Note that the optimal policy with rewards $\tilde{\sigma}_{h,n}^{h_0}$ optimizes $\sigma_{h,n}$ at step $h = h_0$, while the optimal policy with rewards σ_n optimizes the sum of $\sigma_{h,n}$ over all steps.

Step 6b: We bound $V_1^\star(s;\beta(\delta)\tilde{\sigma}_n^{h_0})$ using $V_{1,(n,h_0)}(s)$ where $V_{h,(n,h_0)}$ is the proxy for the value function used in Algorithm 9.

Lemma 7. *Under event $\mathcal{E}'$, for all $s \in \mathcal{S}$,*

$$V_h^\star(s;\beta(\delta)\tilde{\sigma}_n^{h_0}) \le V_{h,(n,h_0)}(s) + (h_0+1-h)(\frac{2\beta(\delta)}{\sqrt{n}} + \frac{2}{n}).$$

Proof of Lemma 7. The lemma is proven by induction, starting from $V_{h_0+1}^\star(\cdot;\beta(\delta)\tilde{\sigma}_n^{h_0}) = V_{h_0+1,(n,h_0)} = \mathbf{0}$. We have, for $h \le h_0$

$$\begin{aligned}
V_h^\star(s;\beta(\delta)\tilde{\sigma}_n^{h_0}) - V_{h,(n,h_0)}(s) &= \max_{a\in\mathcal{A}} Q_h^\star(s,a;\beta(\delta)\tilde{\sigma}_n^{h_0}) - \max_{a\in\mathcal{A}} Q_{h,(n,h_0)}(s,a) \\
&\le \max_{a\in\mathcal{A}} \left\{Q_h^\star(s,a;\beta(\delta)\tilde{\sigma}_n^{h_0}) - Q_{h,(n,h_0)}(s,a)\right\} \\
&\le \max_{a\in\mathcal{A}} \Big\{[P_h V_{h+1}^\star](s,a;\beta(\delta)\tilde{\sigma}_n^{h_0}) \\
&\qquad - [P_h V_{h+1,n}](s,a) + \frac{2\beta(\delta)}{\sqrt{n}} + \frac{2}{n}\Big\} \\
&\le (h_0+1-h)\Big(\frac{2\beta(\delta)}{\sqrt{n}} + \frac{2}{n}\Big).
\end{aligned}$$

The first inequality is due to rearrangement of max, the second inequality holds under $\mathcal{E}'$, and the third inequality is by the base of induction. We thus prove the lemma. □

Step 7b: Fix n and h_0. Let π_{n,h_0} denote the exploration policy in episode $nH + h_0$. We bound $V_{1,(n,h_0)}(s_1) - V_1^{\pi_{n,h_0}}(s_1; \beta(\delta)\tilde{\sigma}_{n-1}^{h_0})$ using $2\beta(\delta)\sum_{h=1}^{h_0}\sigma_{h,n-1}(s_h, a_h)$. Here for simplicity of notation, we use s_h and a_h for the state and action at step h of episode corresponding to n and h_0. In a richer notation, n and h_0 should be specified.

Lemma 8. *Under event $\mathcal{E}'$, we have*

$$\begin{aligned}
V_{1,(n,h_0)}(s_1) - V_1^{\pi_{n,h_0}}(s_1; \beta(\delta)\tilde{\sigma}_{n-1}^{h_0}) \le \sum_{h=1}^{h_0}&\left(2\beta(\delta)\sigma_{h,n-1}(s_h,a_h) + \frac{2\beta(\delta)}{\sqrt{n}} + \frac{2}{n}\right)\\
&+ \sum_{h=1}^{h_0}\Big([P_h V_{h+1,n}](s_h,a_h) - V_{h+1,n}(s_{h+1})\Big)\\
&+ \sum_{h=1}^{h_0}\Big(V_{h+1}^{\pi_{n,h_0}}(s_{h+1}; \beta(\delta)\tilde{\sigma}_{n-1}^{h_0})\\
&\qquad - [P_h V_{h+1}^{\pi_{n,h_0}}](s_h,a_h; \beta(\delta)\tilde{\sigma}_{n-1}^{h_0})\Big).
\end{aligned}$$

The second and third terms are martingale sums which can be bounded using Azuma-Hoeffding inequality, we refer to them as

$$\zeta_{h,(n,h_0)} = [P_h V_{h+1,n}](s_h,a_h) - V_{h+1,n}(s_{h+1})$$

$$\xi_{h,(n,h_0)} = V_{h+1}^{\pi_{n,h_0}}(s_{h+1}; \beta(\delta)\tilde{\sigma}_{n-1}^{h_0}) - [P_h V_{h+1}^{\pi_{n,h_0}}](s_h,a_h; \beta(\delta)\tilde{\sigma}_{n-1}^{h_0})$$

Proof of Lemma 8. We obtain a iterative relation over h. In particular

$$\begin{aligned}
V_{h,(n,h_0)}(s_h) - V_h^{\pi_{n,h_0}}(s_h; \beta(\delta)\tilde{\sigma}_{n-1}^{h_0}) &= Q_{h,(n,h_0)}(s_h,a_h) - Q_h^{\pi_{n,h_0}}(s_h,a_h; \beta(\delta)\tilde{\sigma}_{n-1}^{h_0})\\
&\le [P_h V_{h+1,n}](s_h,a_h) - [P_h V_{h+1}^{\pi_{n,h_0}}](s_h,a_h; \beta(\delta)\tilde{\sigma}_{n-1}^{h_0})\\
&\quad + 2\beta(\delta)\sigma_{h,n-1}(s_h,a_h) + \frac{2\beta(\delta)}{\sqrt{n}} + \frac{2}{n}\\
&= V_{h+1,(n,h_0)}(s_h) - V_{h+1}^{\pi_{n,h_0}}(s_h; \beta(\delta)\tilde{\sigma}_{n-1}^{h_0})\\
&\quad + 2\beta(\delta)\sigma_{h,n-1}(s_h,a_h) + \frac{2\beta(\delta)}{\sqrt{n}} + \frac{2}{n}\\
&\quad + \zeta_{h,(n,h_0)} + \xi_{h,(n,h_0)}.
\end{aligned}$$

Iterating over h and noticing $V_{h_0+1,(n,h_0)} - V^{\pi_{n,h_0}}_{h_0+1}(\cdot;\beta(\delta)\tilde{\sigma}^{h_0}_{n-1}) = \mathbf{0}$ the lemma is proven. $\square$

Step 8b: We note that $V^{\pi_{n,h_0}}_{1,(n,h_0)}(s) = \beta(\delta)\sigma_{h,n-1}(s_{h_0,(n,h_0)}, a_{h_0,(n,h_0)})$.

Step 9b: Combining Steps 5b-8b we conclude that

$$V_1^\star(s;\beta(\delta)\sigma_{n-1}) \le \sum_{h_0=1}^{H}\sum_{h=1}^{h_0}\Bigg(3\beta(\delta)\sigma_{h,n-1}(s_{h,(n,h_0)}, a_{h,(n,h_0)}) + \frac{4\beta(\delta)}{\sqrt{n}} + \frac{4}{n} + \zeta_{h,(n,h_0)} + \xi_{h,(n,h_0)}\Bigg). \tag{B.15}$$

Step 10b: Combining with previous steps similar to the proof of Theorem 2, and using Azuma-Hoeffding inequality on $\zeta_{h,(n,h_0)}$ and $\xi_{h,(n,h_0)}$, we get, with probability $1-\delta$

$$V_1^\star(s) - V_1^\pi(s) \le 3H(H+1)\beta(\delta)\sqrt{\frac{2\Gamma(N)}{N\log(1+1/\tau^2)}} + \frac{8\beta(\delta)H(H+1)}{\sqrt{N}} + \frac{4H(H+1)(\log(N)+1)}{N} + 2H\sqrt{N(H+1)\log\left(\frac{2}{\delta}\right)}.$$

The expression can be simplified as

$$V_1^\star(s) - V_1^\pi(s) = \mathcal{O}\left(H^3\sqrt{\frac{\Gamma(N)\log(NH/\delta)}{N}}\right).$$

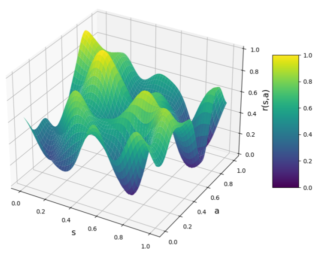
Now, let N_0 be the smallest integer such that the right hand side less than ϵ. For any $N \ge N_0$ the suboptimality gap of the policy is at most ϵ. This completes the proof of Theorem 3.

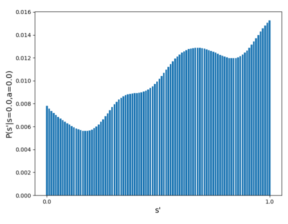
B.4 Experimental Details

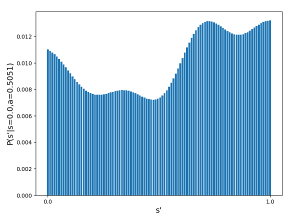
Here, we outline the procedure for generating r and P test functions from the RKHS, the finetuning process of the confidence interval width multiplier β, and the computational resources utilized for the simulations. Additionally, we present further experimental results when various samples of r and P are drawn from the RKHS.

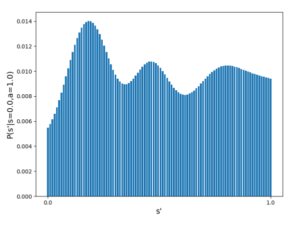
B.4.1 Synthetic Test Functions from the RKHS

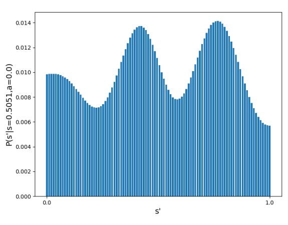
Our reward function r and transition probability P are arbitrarily chosen functions from an RKHS. For the reward function r, we draw a Gaussian Process (GP) sample on a subset of the domain $\mathcal{Z}$. This subset is generated by sampling a set of evenly spaced points on a 10×10 grid spanning the range $[0,1]$ in both dimensions. We then fit kernel ridge regression to these samples and scale the resulting predictions to fit the $[0,1]$ range to obtain r. For $P(s'|s,a)$, we similarly draw a GP sample on a subset of the domain $\mathcal{Z} \times \mathcal{S}$, fit kernel ridge regression to these samples, and then shift and rescale for each z to obtain $P(\cdot|z)$ as a conditional probability distribution. We use the same kernel as the one used in the algorithm. This is a common approach to create functions belonging to an RKHS (e.g., see, [66]). Examples of r and P are visualized in Figures B.1, B.2 and B.3 using SE and Matérn kernels with parameter $\nu = 2.5$ and $\nu = 1.5$, respectively. For all kernels, we use lengthscale of 0.1. With SE kernel, we use $\tau = 0.01$, and with Matérn kernels, we use $\tau = 0.5$.


(a) Reward function $r(s,a)$


(b) $P(s'|(s=0,a=0)$


(c) $P(s'|(s=0,a=0.5051)$


(d) $P(s'|(s=0,a=1)$


(e) $P(s'|(s=0.5051,a=0)$

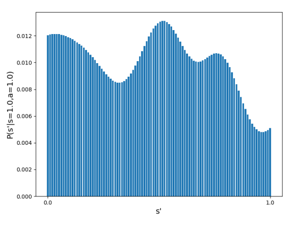
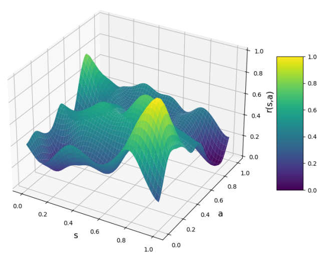
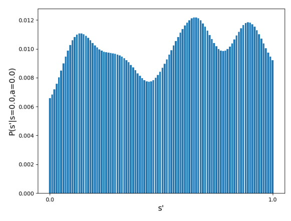

(f) $P(s'|(s=1,a=1)$

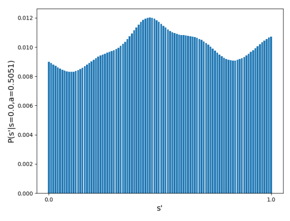
Figure B.1: Reward and transition probability functions generated by kernel ridge regression using SE Kernel with lengthscale = 0.1 and $\tau = 0.01$.



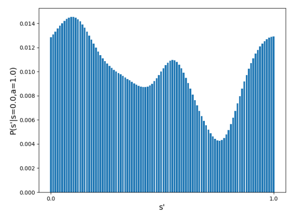
(a) Reward function $r(s,a)$



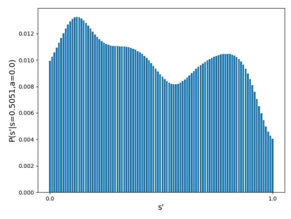
(b) $P(s'|(s=0,a=0)$



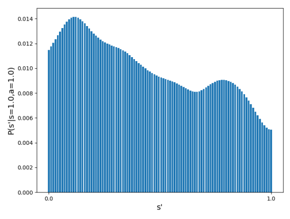
(c) $P(s'|(s=0,a=0.5051)$



(d) $P(s'|(s=0,a=1)$

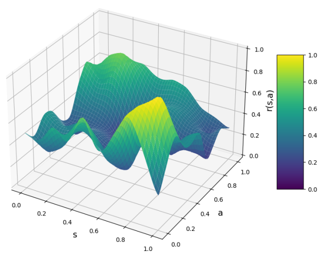


(e) $P(s'|(s=0.5051,a=0)$

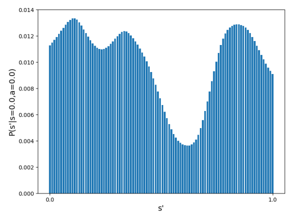


(f) $P(s'|(s=1,a=1)$

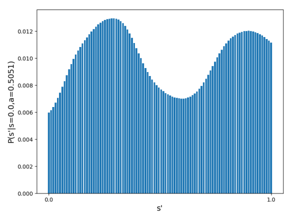
Figure B.2: Reward and transition probability functions generated by kernel ridge regression using Matérn kernel with $\nu = 2.5$, lengthscale $= 0.1$ and $\tau = 0.5$.



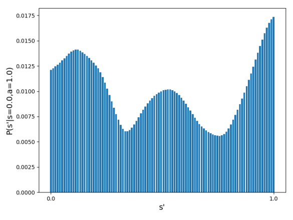
(a) Reward function $r(s,a)$



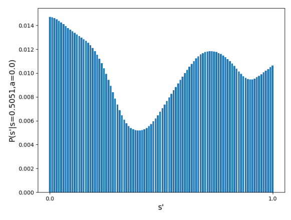
(b) $P(s'|(s=0,a=0)$



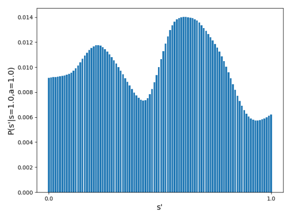
(c) $P(s'|(s=0,a=0.5051)$



(d) $P(s'|(s=0,a=1)$



(e) $P(s'|(s=0.5051,a=0)$

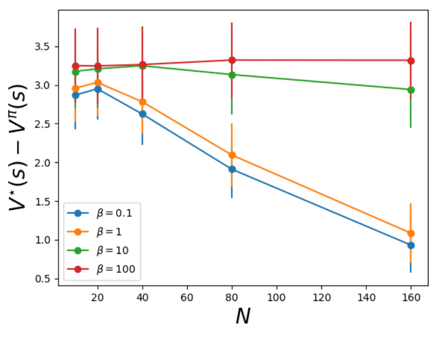


(f) $P(s'|(s=1,a=1)$

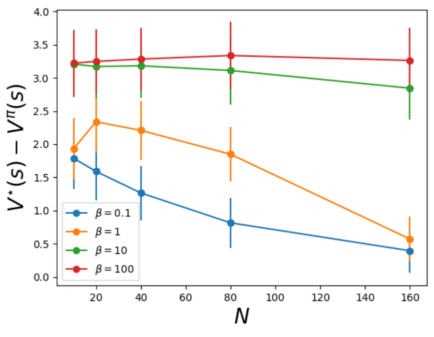
Figure B.3: Reward and transition probability functions generated by kernel ridge regression using Matérn kernel with $\nu = 1.5$, lengthscale $= 0.1$ and $\tau = 0.5$.

B.4.2 Tuning the Confidence Interval Width Multiplier

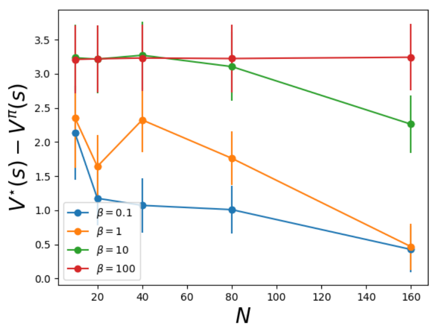
We perform hyperparameter tuning for the confidence interval width multiplier β. The theoretical analysis, especially in [185], leads to high values for β. To ensure a fair comparison between algorithms, we finetune β for [185], our algorithm without a generative model, our algorithm with generative model and Greedy Max Variance, selecting the best value for each. Figures B.4, B.5, B.6 show the simulation results for various values of $\beta \in [0.1, 1, 10, 100]$ for several kernels. The value $\beta = 0.1$ yields the best performance consistently.



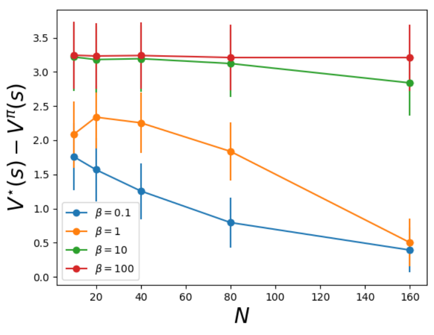
(a) (Qiu et al., 2021)



(b) Without generative model

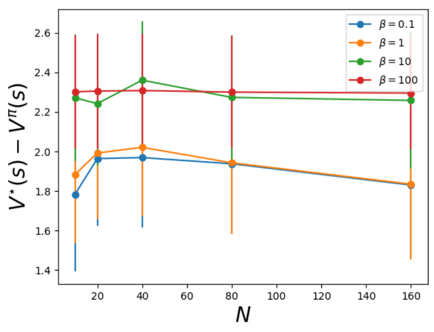


(c) With generative model

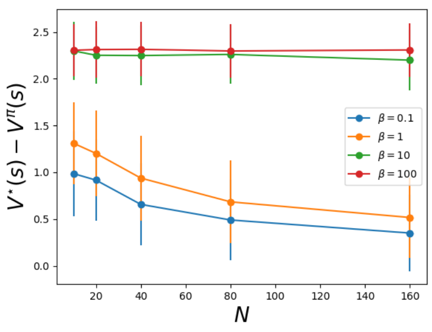


(d) Greedy Max Variance

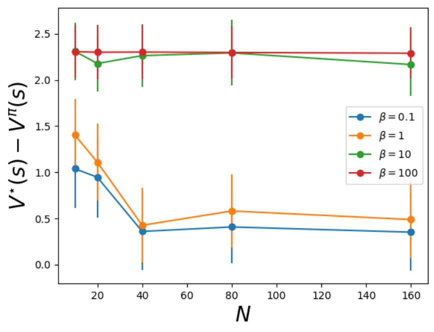
Figure B.4: Average suboptimality gap plotted against the number of episodes N for different values of β in the case of SE kernel.



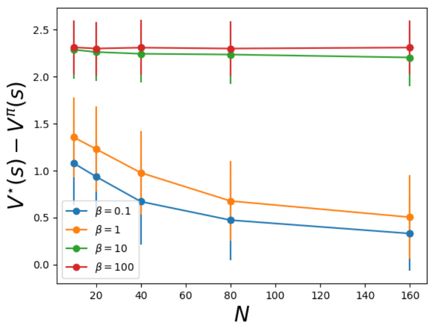
(a) (Qiu et al., 2021)



(b) Without generative model

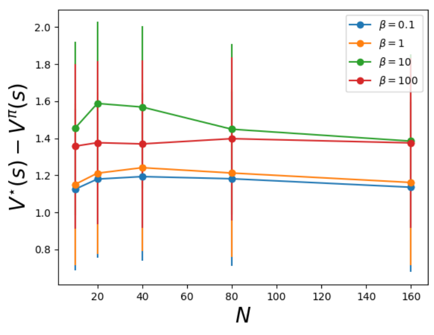


(c) With generative model

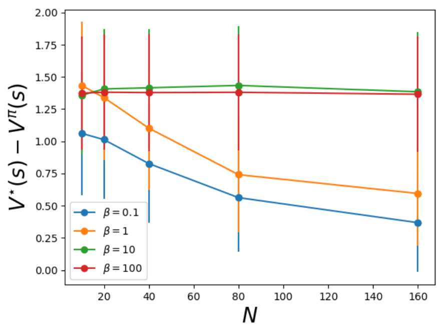


(d) Greedy Max Variance

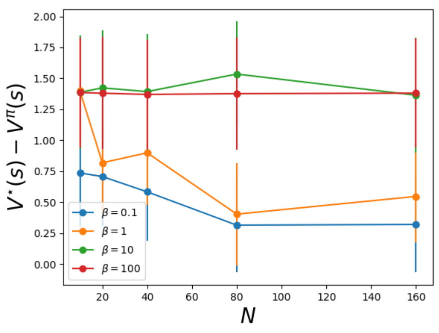
Figure B.5: Average suboptimality gap plotted against the number of episodes N for different values of β in the case of Matérn kernel with $\nu = 2.5$.



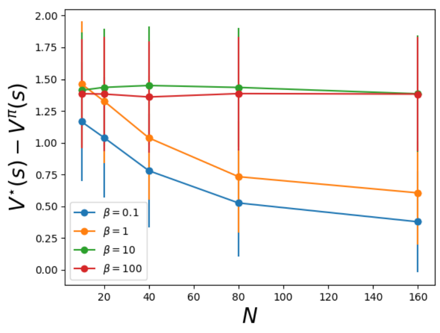
(a) (Qiu et al., 2021)



(b) Without generative model



(c) With generative model



(d) Greedy Max Variance

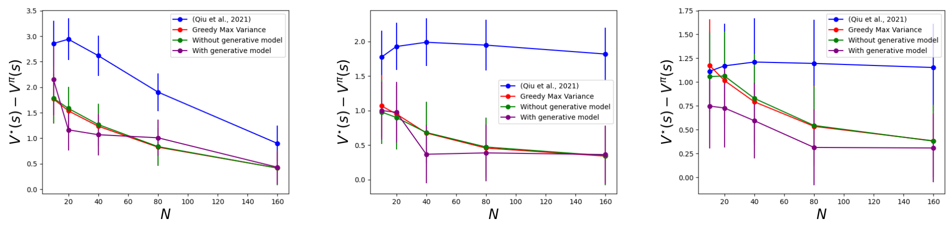
Figure B.6: Average suboptimality gap plotted against the number of episodes N for different values of β in the case of Matérn kernel with $\nu = 1.5$.

B.4.3 Implementation and Computational Resources

For kernel ridge regression, we used Sickit-Learn library [269], which offers robust and efficient tools for implementing and tuning kernel-based machine learning models. The simulations were executed on a cluster which has 376.2 GiB of RAM, and an Intel(R) Xeon(R) Gold 5118 CPU running at 2.30 GHz. The algorithm by [185], our algorithm without a generative model, and the Greedy Max Variance algorithm typically require approximately 2 minutes of CPU time on average per run. However, our algorithm with a generative model requires around 7 minutes per run due to the cost of increasing the number of exploration episodes by a factor of H.

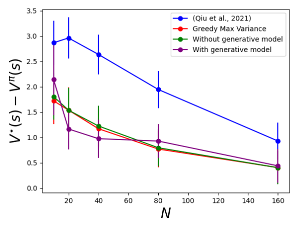
B.4.4 Repeated Experiments for Different Draws of r and P

To validate the robustness of the results against specific environment realizations, we ran the experiments three times, with each repetition using different reward and transition probability functions drawn from the RKHS. We kept the hyperparameters (lengthscale, τ, and β) identical to the ones used in the main text of the thesis. The results remained consistent across all repetitions, as shown in Figures B.7, B.8 and B.9 for different kernels and algorithms.

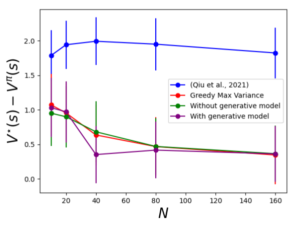


(a) SE Kernel **(b)** Matérn with $\nu = 2.5$ **(c)** Matérn with $\nu = 1.5$

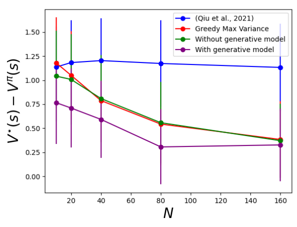
Figure B.7: Average suboptimality gap plotted against N for experiment 1.



(a) SE Kernel

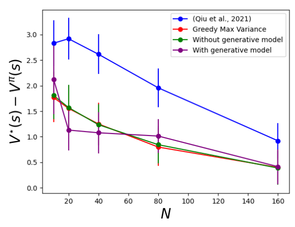


(b) Matérn with $\nu = 2.5$

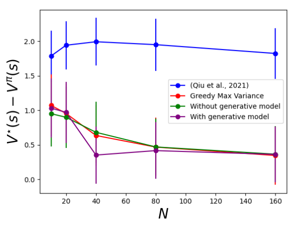


(c) Matérn with $\nu = 1.5$

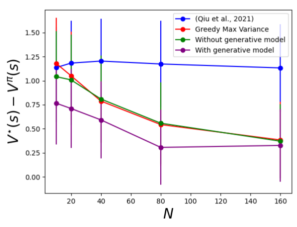
Figure B.8: Average suboptimality gap plotted against N for experiment 2.



(a) SE Kernel



(b) Matérn with $\nu = 2.5$



(c) Matérn with $\nu = 1.5$

Figure B.9: Average suboptimality gap plotted against N for experiment 3.

B.5 RKHS and Mercer Theorem

Mercer theorem [270] provides a representation of the kernel in terms of an infinite dimensional feature map (e.g., see, [271], Theorem 4.49). Let $\mathcal{Z}$ be a compact metric space and μ be a finite Borel measure on $\mathcal{Z}$ (we consider Lebesgue measure in a Euclidean space). Let $L^2_\mu(\mathcal{Z})$ be the set of square-integrable functions on $\mathcal{Z}$ with respect to μ. We further say a kernel is square-integrable if

$$\int_{\mathcal{Z}}\int_{\mathcal{Z}} k^2(z,z')\,d\mu(z)d\mu(z') < \infty.$$

Theorem 6. *(Mercer Theorem) Let $\mathcal{Z}$ be a compact metric space and μ be a finite Borel measure on $\mathcal{Z}$. Let k be a continuous and square-integrable kernel, inducing an integral operator $T_k : L^2_\mu(\mathcal{Z}) \to L^2_\mu(\mathcal{Z})$ defined by*

$$(T_k f)(\cdot) = \int_{\mathcal{Z}} k(\cdot, z') f(z')\,d\mu(z'),$$

where $f \in L^2_\mu(\mathcal{Z})$. Then, there exists a sequence of eigenvalue-eigenfeature pairs $\{(\gamma_m, \varphi_m)\}_{m=1}^\infty$ such that $\gamma_m > 0$, and $T_k\varphi_m = \gamma_m\varphi_m$, for $m \geq 1$. Moreover, *the kernel function can be represented as*

$$k\left(z, z'\right) = \sum_{m=1}^{\infty} \gamma_m \varphi_m(z) \varphi_m\left(z'\right),$$

where the convergence of the series holds uniformly on $\mathcal{Z} \times \mathcal{Z}$.

According to the Mercer representation theorem (e.g., see, [271], Theorem 4.51), the RKHS induced by k can consequently be represented in terms of $\{(\gamma_m, \varphi_m)\}_{m=1}^{\infty}$.

Theorem 7. *(Mercer Representation Theorem) Let $\{(\gamma_m, \varphi_m)\}_{i=1}^{\infty}$ be the Mercer eigenvalue-eigenfeature pairs. Then, the RKHS of k is given by*

$$\mathcal{H}_k = \left\{ f(\cdot) = \sum_{m=1}^{\infty} w_m \gamma_m^{\frac{1}{2}} \varphi_m(\cdot) : w_m \in \mathbb{R}, \|f\|_{\mathcal{H}_k}^2 := \sum_{m=1}^{\infty} w_m^2 < \infty \right\}.$$

Mercer representation theorem indicates that the scaled eigenfeatures $\{\sqrt{\gamma_m}\varphi_m\}_{m=1}^{\infty}$ form an orthonormal basis for $\mathcal{H}_k$.

Appendix C

Appendix of Chapter 6

C.1 Proof of The Regret Bound and Sample Complexities

In this section, we provide a detailed proof of Theorem 4 on the regret bound of MR-LPF and following corollaries.

C.1.1 Proof of Theorem 4

To prove this theorem, we bound the regret for each round and then sum these bounds over all the rounds.

Regret in the first round: The first round consists of $N_1 = \lceil \sqrt{T} \rceil$ samples. We note that for all t,

$$\frac{\mathbb{P}(x^\star \succ x_t) + \mathbb{P}(x^\star \succ x'_t) - 1}{2} \leq \frac{1}{2}. \tag{C.1}$$

Consequently, the regret incurred in the first round in bounded by $\frac{1}{2}\lceil \sqrt{T} \rceil$.

For the second round onwards ($r \geq 2$), we introduce some notation and preliminaries that will assist in bounding the regret.

High probability events: Let us define the event $\mathcal{E}_r$ as the event that all the confidence intervals used in the round r of the MR-LPF algorithm hold true.

Specifically,

$$\mathcal{E}_r = \left\{\forall x,x' \in \mathcal{M}_r : \left|\mu(h_{(N_r,r)}(x,x')) - \mu(h(x,x'))\right| \le \beta_{(r)}(\delta)\sigma_{(N_r,r)}(x,x')\right\} \tag{C.2}$$

Recall that $\beta_{(r)}(\delta) = L\left(B + \sqrt{\frac{\kappa_r}{\lambda}\log\left(\frac{2R|\mathcal{X}|}{\delta}\right)}\right)$. We also define $\mathcal{E} = \bigcup_{r=1}^{R}\mathcal{E}_r$.

Sum of the posterior variances for a sequence of observations: We apply the following bound on the sum of posterior variances in each round (see, e.g., [116], Lemma 14]).

$$\sum_{n=1}^{N_r}\sigma^2_{(n-1,r)}(x_{(n,r)},x'_{(n,r)}) \le \frac{8}{\log(1+4(\lambda\kappa_r)^{-1})}\Gamma_{(\lambda\kappa_r)}(N_r). \tag{C.3}$$

By the selection rule of $(x_{(n,r)},x'_{(n,r)})$ in MR-LPF as the points with the highest variance, we have that $\forall x,x' \in \mathcal{M}_r$, and $\forall n \le N_r$, $\sigma_{(N_r,r)}(x,x') \le \sigma_{(n-1,r)}(x_{(n,r)},x'_{(n,r)})$. Combining this with Equation (C.3), we conclude that $\forall x,x' \in \mathcal{M}_r$,

$$\sigma_{(N_r,r)}(x,x') \le \sqrt{\frac{8}{\log(1+4(\lambda\kappa_r)^{-1})}}\sqrt{\frac{\Gamma_{(\lambda\kappa_r)}(N_r)}{N_r}}. \tag{C.4}$$

The value of $\kappa_r, r \ge 2$: Recall the update rule for $\mathcal{M}_r$ in MR-LPF:

$$\mathcal{M}_{r+1} = \{x \in \mathcal{M}_r | \forall x' \in \mathcal{M}_r : \mu(h_{(N_r,r)}(x,x')) + \beta_{(r)}\sigma_{(N_r,r)}(x,x') \ge \frac{1}{2}\} \tag{C.5}$$

Assuming $\mathcal{E}_1$, for all $x,x' \in \mathcal{M}_2$, we have

$$\begin{aligned}\mu(h(x,x')) + 2\beta_{(1)}\sigma_{(N_1,1)}(x,x') &\ge \mu(h_{(N_1,1)}(x,x')) + \beta_{(1)}\sigma_{(N_1,1)}(x,x')\\ &\ge \frac{1}{2},\end{aligned} \tag{C.6}$$

where the first inequality holds under $\mathcal{E}_1$ and the second inequality is a consequence of the update rule. Similarly, we have

$$\mu(h(x',x)) + 2\beta_{(1)}\sigma_{(N_1,1)}(x',x) \ge \frac{1}{2}. \tag{C.7}$$

We note that $\forall x, x' \in \mathcal{X}$, $\mu(h(x', x)) = 1 - \mu(h(x, x'))$. Thus, Equation (C.7) implies that

$$\mu(h(x,x')) \leq \frac{1}{2} + 2\beta_{(1)}\sigma_{(N_1,1)}(x',x). \tag{C.8}$$

Combining with (C.6), we obtain that

$$-2\beta_{(1)}\sigma_{(N_1,1)}(x,x') \leq \mu(h(x,x')) - \frac{1}{2} \leq 2\beta_{(1)}\sigma_{(N_1,1)}(x',x). \tag{C.9}$$

We previously established a bound on the standard deviation at the end of rounds in (C.4). Applying this to the first round, with length $N_1 = \lceil\sqrt{T}\rceil$, we can bound $\mu(h(x,x'))$ for all $x, x' \in \mathcal{M}_2$ within the interval $[\frac{1}{4}, \frac{3}{4}]$ by ensuring $2\beta_{(1)}\sigma_{(N_1,1)}(x',x) \leq \frac{1}{4}$. Specifically, let T_0 be the smallest integer satisfying

$$2\beta_{(1)}(\delta)\sqrt{\frac{8}{\log(1+4(\lambda\kappa)^{-1})}}\sqrt{\frac{\Gamma_{(\lambda\kappa)}(\lceil\sqrt{T_0}\rceil)}{\lceil\sqrt{T_0}\rceil}} \leq \frac{1}{4}. \tag{C.10}$$

Then, for any $T \geq T_0$, for all $x, x' \in \mathcal{M}_2$, we have $\mu(h(x,x')) \in [\frac{1}{4}, \frac{3}{4}]$. Recall that the derivative of the sigmoid function is given by $\dot{\mu}(x) = \mu(\cdot)(1-\mu(\cdot))$. Consequently, the inverse of the derivative of the sigmoid applied to h, for the values of $x, x' \in \mathcal{M}_2$, is bounded as follows. For all $x, x' \in \mathcal{M}_2$,

$$\frac{1}{\mu(h(x,x'))(1-\mu(h(x,x')))} \leq \frac{16}{3} < 6. \tag{C.11}$$

Thus, we can use $\kappa_r = 6$ for all $r \geq 2$, maintaining the validity of the confidence intervals.

Lemma 9. *For $T \geq T_0$ specified in Equation* (C.10)*, we have $\mathbb{P}(\mathcal{E}) \leq 1-\delta$.*

The proof follows from Theorem 5, a union bound over all action pairs and rounds, and the bound on κ_r derived above. We condition the remainder of the proof on the event $T \geq T_0$ and $\mathcal{E}$.

The best action $x^\star$ will not be removed. Assuming $\mathcal{E}$, the best action will not be removed from the sets $\mathcal{M}_r$ by the MR-LPF algorithm in any round. We formalize this observation in the following lemma.

Lemma 10. *Under event $\mathcal{E}$, $x^\star \in \mathcal{M}_R$.*

The proof follows from the observation that $\mu(h(x^\star,x)) \geq \frac{1}{2}$ for all x. Combining with the confidence intervals in $\mathcal{E}$, $\forall r$, $\forall x \in \mathcal{M}_r$, $\mu(h_{(N_r,r)}(x^\star,x)) + \beta_{(r)}(\delta)\sigma_{(N_r,r)}(x^\star,x) \geq \frac{1}{2}$. Consequently, the best action $x^\star$ will not be removed. We are now ready to bound the regret in rounds $r \geq 2$.

Regret bound in each round $r \geq 2$: For each $x \in \mathcal{M}_r$, we use the update rule of $\mathcal{M}_r$ in MR-LPF to bound the regret with respect to the optimal action. Recall that in Lemma 10, we showed that the optimal action remains in $\mathcal{M}_r$ for all r. We have

$$\begin{aligned}
&\mu(h(x,x^\star)) + 2\beta_{(r-1)}(\delta)\sigma_{(N_{r-1},r-1)}(x,x^\star) \\
&\quad \geq \mu(h_{(N_{r-1},r-1)}(x,x^\star)) + \beta_{(r-1)}(\delta)\sigma_{(N_{r-1},r-1)}(x,x^\star) \\
&\quad \geq \frac{1}{2}, \qquad \text{(C.12)}
\end{aligned}$$

where the first inequality holds under $\mathcal{E}$, and the second inequality follows from the update rule of $\mathcal{M}_r$. Then, we have

$$\begin{aligned}
\mu(h(x^\star,x)) &= 1 - \mu(h(x,x^\star)) \\
&\leq \frac{1}{2} + 2\beta_{(r-1)}(\delta)\sigma_{(N_{r-1},r-1)}(x,x^\star), \qquad \text{(C.13)}
\end{aligned}$$

The equality follows from $\mu(-\cdot) = 1 - \mu(\cdot)$, and the inequality follows from (C.12).

We thus have for all $x \in \mathcal{M}_r$,

$$\begin{aligned}
\mu(h(x^\star,x)) - \frac{1}{2} &\leq 2\beta_{(r-1)}(\delta)\sigma_{(N_{r-1},r-1)}(x,x^\star) \\
&\leq 2\beta_{(r-1)}(\delta) C \sqrt{\frac{\Gamma_{(\lambda\kappa_{r-1})}(N_{r-1})}{N_{r-1}}}, \qquad \text{(C.14)}
\end{aligned}$$

where the second inequality is proven in (C.4), and we use $C = \sqrt{\frac{8}{\log(1+4(6\lambda)^{-1})}}$ to simplify the notation. This bound holds for all points in round r. Therefore, to obtain the regret in round r, it is sufficient to multiply this bound by N_r.

This results in the following bound on the regret in round r:

$$\begin{aligned}\text{Regret in Round } r &\leq 2\beta_{(r-1)}(\delta) C N_r \sqrt{\frac{\Gamma_{(\lambda\kappa_{r-1})}(N_{r-1})}{N_{r-1}}} \\ &\leq 2\beta_{(r-1)}(\delta) C \left(\sqrt{T\Gamma_{(4\lambda)}(T)} + \frac{1}{\sqrt{N_{r-1}}}\sqrt{\Gamma_{(4\lambda)}(T)}\right) \\ &\leq 2\beta_{(r-1)}(\delta) C \left(\sqrt{T\Gamma_{(4\lambda)}(T)} + T^{-1/4}\sqrt{\Gamma_{(4\lambda)}(T)}\right), \quad \text{(C.15)}\end{aligned}$$

where the second inequality is obtained by substituting $N_r = \lceil\sqrt{N_{r-1}T}\rceil$ and using $\lceil\cdot\rceil \leq \cdot + 1$. We also use that $\Gamma_{(\lambda\kappa_{r-1})}(.) \leq \Gamma_{(4\lambda)}(.)$ since $\kappa_{r-1} \geq 4$. The third inequality follows from $N_r \geq \sqrt{T}$ for all $r \geq 1$.

Total regret: The number of rounds R is at most $\lceil \log\log_2(T)\rceil + 1$, see Proposition 1 of [84]. Using the bound on regret in each round, we can bound the total regret of MR-LPF algorithm as follows

$$R(T) \leq 2CR\beta_{(R)}(\delta)\sqrt{T\Gamma_{(4\lambda)}(T)} + 2CR\beta_{(R)}(\delta)T^{-1/4}\sqrt{\Gamma_{(4\lambda)}(T)}. \quad \text{(C.16)}$$

This completes the proof of Theorem 4.

C.1.2 Proof of Corollary 2

Since the size N_r of rounds increase with r, we have $N_R \geq T/R$. In the proof of Theorem 4, in (C.14), we showed that, for all $x \in \mathcal{M}_r$

$$\mu(h(x^\star, x)) - \frac{1}{2} \leq 2\beta_{(r-1)}(\delta) C \sqrt{\frac{\Gamma_{(\lambda\kappa_{r-1})}(N_{r-1})}{N_{r-1}}}$$

Thus, for $x \in \mathcal{M}_{R+1}$, we have

$$\begin{aligned}\mu(h(x^\star, x)) - \frac{1}{2} &\leq 2\beta_{(R)}(\delta) C \sqrt{\frac{\Gamma_{(4\lambda)}(N_R)}{N_R}} \\ &\leq 2\beta_{(R)}(\delta) C \sqrt{\frac{R\Gamma_{(4\lambda)}(N_R)}{T}} \quad \text{(C.17)} \\ &\leq 2\beta_{(R)}(\delta) C \sqrt{\frac{R\Gamma_{(4\lambda)}(T)}{T}}, \quad \text{(C.18)}\end{aligned}$$

where, for the second inequality, we used $N_R \geq \frac{T}{R}$, and for the third inequality, we used $N_R \leq T$.

C.1.3 Proof of Corollary 3

Following the bounds obtained in Corollary 2, we determine T that ensures $\mu(h(\hat{x}_T, x)) - \frac{1}{2} \leq \epsilon$, after T steps. For this, we need specification of $\Gamma_\lambda(T)$.
In the case of linear kernels, we have $\Gamma_\lambda(T) = \mathcal{O}(d\log(T))$. Consequently, a choice of $T = \tilde{\mathcal{O}}\left(\frac{d\log(\frac{1}{\delta})}{\epsilon^2}\right)$ ensures $\mu(h(x^\star, x)) - \frac{1}{2} \leq \epsilon$.
In the case of SE kernel, we have $\Gamma_\lambda(T) = \mathcal{O}(\log^{d+1}(T))$. Consequently, a choice of $T = \tilde{\mathcal{O}}\left(\frac{\log(\frac{1}{\delta})}{\epsilon^2}\right)$ ensures $\mu(h(x^\star, x)) - \frac{1}{2} \leq \epsilon$.
In the case of Matérn kernel, we have $\Gamma_\lambda(T) = \tilde{\mathcal{O}}(T^{\frac{d}{2\nu+d}})$. Consequently, a choice of $T = \tilde{\mathcal{O}}\left(\frac{\log(\frac{1}{\delta})}{\epsilon^{2+\frac{d}{\nu}}}\right)$ ensures $\mu(h(x^\star, x)) - \frac{1}{2} \leq \epsilon$.
For the bound on $\Gamma_\lambda(T)$ see, e.g., [68].

C.2 Proof of Theorem 5

Recall the conventional kernel-based regression discussed in Section 6.2. Various confidence intervals of the form $|f(z) - \hat{f}_t(z)| \leq \beta(\delta)\sigma_t(z)$, where $\hat{f}_t(z)$ and $\sigma_t(z)$ are the conventional prediction and standard deviation, and $\beta(\delta)$ is a confidence interval width multiplier for a $1-\delta$ confidence level, have been demonstrated in several works [213, 66, 70, 72]. As discussed in the preference-based case, the problem becomes more similar to a classification problem with binary feedback, and these confidence intervals are not directly applicable. Moreover, a closed-form solution for h_t is not available, as it is only provided as the minimizer of the loss function given in Equation (6.5). Additionally, as discussed, this loss and its solution can be parameterized using the representer theorem.

$$\begin{aligned}\mathcal{L}_{\Bbbk}(\boldsymbol{\theta}, \mathbb{H}_t) = \sum_{i=1}^{t} & -y_i \log \mu(\boldsymbol{\theta}^\top \Bbbk_t(x_i, x_i')) \\ & - (1-y_i)\log(1-\mu(\boldsymbol{\theta}^\top \Bbbk_t(x_i, x_i')) + \frac{\lambda}{2}||\boldsymbol{\theta}||_2^2, \end{aligned} \tag{C.19}$$

and

$$h_t(\cdot) = \sum_{i=1}^{t} \theta_i \mathbb{k}\left(\cdot, (x_i, x_i')\right). \tag{C.20}$$

For the remainder of the proof, and for simplicity of presentation, we use the notation $z = (x, x')$ and similarly $z_i = (x_i, x_i')$.

In both [235] and [116], confidence intervals for $|h(z) - h_t(z)|$ are derived, with [116] establishing tighter bounds. Their confidence intervals are based on the results of [239] for logistic bandits and [72] for confidence intervals in kernel bandits. In comparison, our confidence intervals are tighter than those presented in [116] by a factor of $\mathcal{O}(\sqrt{\Gamma_\lambda(T)})$. We achieve this improvement by assuming that the sequence of observation points $\{z_i\}_{i=1}^t$ is independent of the observation values $\{y_i\}_{i=1}^t$, inspired by [70]. This assumption is made possible in our work due to the design of the MR-LPF algorithm, where within each round, the observation points are selected based solely on kernel-based variance, which, by definition, does not depend on the observation values.

The main steps of the proof are similar to those in the proof of the confidence interval in [116], and we will highlight the key differences in our proof. The key idea is that the derivative of the loss $\mathcal{L}_{\mathbb{k}}$, as given in Equation (C.19), is the null operator at the minimizer of the loss:

$$\nabla \mathcal{L}(\boldsymbol{\theta}_t, \mathbb{H}_t) = \sum_{i=1}^{t} -y_i \mathbb{k}(z_i, \cdot) + g_t(\boldsymbol{\theta}_t) = 0, \tag{C.21}$$

where $g_t(\boldsymbol{\theta}) : \mathcal{H}_{\mathbb{k}} \to \mathcal{H}_{\mathbb{k}}$ is a linear operator defined as

$$g_t(\boldsymbol{\theta}) = \sum_{i=1}^{t} \mathbb{k}(z_i, \cdot) \mu(\boldsymbol{\theta}^\top \mathbb{k}(z_i, \cdot)) + \lambda \boldsymbol{\theta}. \tag{C.22}$$

Recall that $\boldsymbol{\theta}_t$ is the minimizer of the loss in Equation (C.19). Consequently, we have $g_t(\boldsymbol{\theta}_t) = \sum_{i=1}^t y_i \mathbb{k}(z_i, \cdot)$.

Then, confidence intervals are proven for the gradient and extended to the preference function itself. We now introduce some auxiliary notation that will be helpful throughout the rest of the proof. Let $\boldsymbol{\Phi}_t =$

$[\Bbbk(z_1,\cdot),\Bbbk(z_2,\cdot),\dots,\Bbbk(z_t,\cdot)]^\top$, from which we define the kernel matrix $\mathbb{K}_t = \boldsymbol{\Phi}_t\boldsymbol{\Phi}_t^\top$ and the operator $S_t = \boldsymbol{\Phi}_t^\top\boldsymbol{\Phi}_t$. We also use I_t to denote the t-dimensional identity matrix and $I_{\mathcal{H}}$ to denote the identity operator in the RKHS. Finally, we define $V_t = S_t + \kappa\lambda I_{\mathcal{H}}$.

We also use the auxiliary notation G_t as in Appendix B of [116], where

$$G_t(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2) = \lambda I_{\mathcal{H}} + \sum_{i=1}^{t}\alpha(z_i;\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)\phi(z_i)\phi^\top(z_i),$$

and

$$\alpha(z,\boldsymbol{\theta}_1,\boldsymbol{\theta}_2) = \int_0^1 \dot{\mu}\left(\nu\,\boldsymbol{\theta}_2^\top\phi(z) + (1-\nu)\,\boldsymbol{\theta}_1^\top\phi(z)\right)d\nu$$

is the coefficient arising from the mean value theorem, such that

$$\mu(\boldsymbol{\theta}_2^\top\phi(z)) - \mu(\boldsymbol{\theta}_1^\top\phi(z)) = \alpha(z,\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)(\boldsymbol{\theta}_2-\boldsymbol{\theta}_1)^\top\phi(z).$$

See, [116], Lemma 11 for details. It then follows that

$$g_t(\boldsymbol{\theta}_2) - g_t(\boldsymbol{\theta}_1) = G_t(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)(\boldsymbol{\theta}_2-\boldsymbol{\theta}_1), \tag{C.23}$$

as shown in the proof of Lemma 12 in [116]. We use this relation, along with the inequality

$$G_t(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2) \succeq \kappa^{-1}V_t, \tag{C.24}$$

where $\succeq$ denotes the Loewner order, also from the proof of Lemma 12, in our analysis.

We use the notation $h(z) = \boldsymbol{\phi}^\top(z)\boldsymbol{\theta}^\star$ for the underlying preference function and $\varepsilon_i = y_i - \mu(h(z_i))$ to represent the sequence of observation noise.

Inspired by the proof of confidence intervals in [70], we express the prediction error as:

$$\begin{aligned}
&|\mu(h_t(z)) - \mu(h(z))| \\
&\le L\,|h_t(z) - h(z)| \\
&= L\left|\boldsymbol{\phi}^\top(z)(\boldsymbol{\theta}_t - \boldsymbol{\theta}^\star)\right| \\
&= L\left|\boldsymbol{\phi}^\top(z)\,G_t(\boldsymbol{\theta}^\star,\boldsymbol{\theta}_t)^{-1}\,(g_t(\boldsymbol{\theta}_t) - g_t(\boldsymbol{\theta}^\star))\right| \\
&= L\left|\boldsymbol{\phi}^\top(z)\,G_t(\boldsymbol{\theta}^\star,\boldsymbol{\theta}_t)^{-1}\left(\sum_{i=1}^t (y_i - \mu(h(z_i)))\,\boldsymbol{\phi}(z_i) - \lambda\boldsymbol{\theta}^\star\right)\right| \\
&= L\left|\boldsymbol{\phi}^\top(z)\,G_t(\boldsymbol{\theta}^\star,\boldsymbol{\theta}_t)^{-1}\left(\sum_{i=1}^t \varepsilon_i\,\boldsymbol{\phi}(z_i) - \lambda\boldsymbol{\theta}^\star\right)\right| \\
&\le \underbrace{L\left|\boldsymbol{\phi}^\top(z)\,G_t(\boldsymbol{\theta}^\star,\boldsymbol{\theta}_t)^{-1}\left(\sum_{i=1}^t \varepsilon_i\,\boldsymbol{\phi}(z_i)\right)\right|}_{\text{Stochastic Term}} + \underbrace{L\lambda\left|\boldsymbol{\phi}^\top(z)\,G_t(\boldsymbol{\theta}^\star,\boldsymbol{\theta}_t)^{-1}\boldsymbol{\theta}^\star\right|}_{\text{Bias Term}}
\end{aligned}$$

The first line follows from the Lipschitz continuity of the sigmoid function. The second line uses the representer theorem to express $h_t(z) = \boldsymbol{\phi}^\top(z)\boldsymbol{\theta}_t$ and $h(z) = \boldsymbol{\phi}^\top(z)\boldsymbol{\theta}^\star$, where $\boldsymbol{\phi}(z) = \Bbbk(z,\cdot)$, defined similarly to Appendix A of [116]. The third line uses (C.23). The fourth line uses that $\boldsymbol{\theta}_t$ is the minimizer of the loss in Equation (C.19). The fifth line uses the notation $\varepsilon_i = y_i - \mu(h(z_i))$ for the observation noise. Finally, the expression is split into a stochastic term and a bias term, allowing us to follow the proof structure of the confidence bound in Theorem 1 of [70].

The stochastic term is a sub-Gaussian random variable and can be bounded with high probability using standard concentration results. In particular, the sub-Gaussian parameter is determined by the norm of the coefficients applied to the independent noise terms ε_i, which are 1/2-sub-Gaussian. This follows from the fact that $\varepsilon_i = y_i - \mu(h(z_i)) \in [-\mu(h(z_i)), 1 - \mu(h(z_i))]$, and therefore the noise sequence has bounded support of length 1.

$$\begin{aligned}
\frac{1}{2}L\left\|\boldsymbol{\phi}^\top(z)\,G_t(\boldsymbol{\theta}^\star,\boldsymbol{\theta}_t)^{-1}\boldsymbol{\Phi}_t\right\| &\le \frac{1}{2}L\|\boldsymbol{\phi}(z)\|_{G_t(\boldsymbol{\theta}^\star,\boldsymbol{\theta}_t)^{-1}}\|\boldsymbol{\Phi}_t G_t(\boldsymbol{\theta}^\star,\boldsymbol{\theta}_t)^{-1}\boldsymbol{\Phi}_t^\top\|_{\text{op}}^{1/2} \\
&\le \frac{1}{2}L\kappa\|\boldsymbol{\phi}(z)\|_{V_t^{-1}}\|\boldsymbol{\Phi}_t V_t^{-1}\boldsymbol{\Phi}_t^\top\|_{\text{op}}^{1/2}
\end{aligned}$$

$$\leq \frac{1}{2} L \sqrt{\frac{\kappa}{\lambda}} \sigma_t(z), \tag{C.25}$$

where $\|\cdot\|_{\mathrm{op}}$ denotes the operator (spectral) norm. The first inequality follows from matrix arithmetic and the definition of operator norm. The second uses (C.24). The third uses the identity $\|\boldsymbol{\phi}(z)\|_{V_t^{-1}} = \frac{1}{\sqrt{\lambda\kappa}}\sigma_t(z)$ (see, e.g., [116]), along with $\|\boldsymbol{\phi}_t V_t^{-1} \boldsymbol{\phi}_t^\top\|_{\mathrm{op}} \leq 1$, which follows from the eigenvalue bounds of $\boldsymbol{\phi}_t\boldsymbol{\phi}_t^\top$ and V_t^{-1}.

Therefore, by the concentration inequality for sub-Gaussian random variables (see, e.g., [272]), with probability at least $1-\delta$,

$$L\left|\boldsymbol{\phi}^\top(z)\, G_t(\boldsymbol{\theta}^\star, \boldsymbol{\theta}_t)^{-1} \left(\sum_{i=1}^{t} \varepsilon_i\, \boldsymbol{\phi}(z_i)\right)\right| \leq \frac{1}{2} L \sqrt{\frac{\kappa}{\lambda}} \sigma_t(z) \sqrt{2\log(2/\delta)}.$$

The bias term is bounded as:

$$\begin{aligned} L\lambda\left|\boldsymbol{\phi}^\top(z)\, G_t(\boldsymbol{\theta}^\star, \boldsymbol{\theta}_t)^{-1} \boldsymbol{\theta}^\star\right| &\leq L\lambda \|\boldsymbol{\phi}(z)\|_{G_t(\boldsymbol{\theta}^\star,\boldsymbol{\theta}_t)^{-1}} \|\boldsymbol{\theta}^\star\|_{G_t(\boldsymbol{\theta}^\star,\boldsymbol{\theta}_t)^{-1}} \\ &\leq L\lambda\kappa \|\boldsymbol{\phi}(z)\|_{V_t^{-1}} \|\boldsymbol{\theta}^\star\|_{V_t^{-1}} \\ &\leq LB\sigma_t(z), \end{aligned} \tag{C.26}$$

where the second line uses (C.24), and the third line uses $\|\boldsymbol{\phi}(z)\|_{V_t^{-1}} = \frac{1}{\sqrt{\lambda\kappa}}\sigma_t(z)$, as discussed above. It also uses the bound $\|\boldsymbol{\theta}^\star\|_{V_t^{-1}} \leq \frac{1}{\sqrt{\lambda\kappa}}B$, which follows from:

$$\lambda\|\boldsymbol{\theta}^\star\|_{V_t^{-1}} \leq \frac{\lambda}{\sqrt{\lambda\kappa}}\|\boldsymbol{\theta}^\star\| \leq \sqrt{\frac{\lambda}{\kappa}} B, \tag{C.27}$$

where the first inequality follows from the fact that the smallest eigenvalue of V_t is at least $\lambda\kappa$, and the second follows from the RKHS norm bound $\|\boldsymbol{\theta}^\star\| \leq B$.

Combining both bounds gives the following expression for $\beta(\delta)$:

$$\beta(\delta) = LB + \frac{L}{2}\sqrt{\frac{2\kappa}{\lambda}\log(2/\delta)}. \tag{C.28}$$

C.3 Experimental Details

In this section, we provide details on the experimental setting. We describe the RKHS test functions, the Ackley function, and the Yelp Open Dataset used in our experiments. Additionally, we outline the selected hyperparameters and the computational resources utilized in our simulations. We also present the MaxMinLCB algorithm of [116].

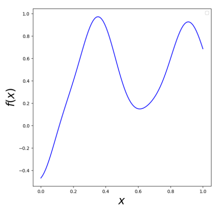
RKHS test functions: In Section 6.5, we outlined the procedure for generating the test function f as an arbitrary function within the RKHS of a given kernel. In Figure C.1, we display the test functions generated in the RKHS for the SE kernel and the Matérn kernels with $\nu = 2.5$ and $\nu = 1.5$. The figure includes plots of the utility function f, the preference function $h(x,x') = f(x) - f(x')$, and the probability of preference $\mu(h(x,x'))$.

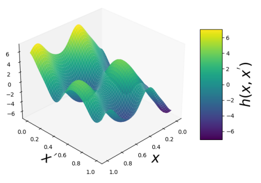
Ackley test function: It is defined as follows (with $d = 1$ and $\mathcal{X} = [-5,5]$):

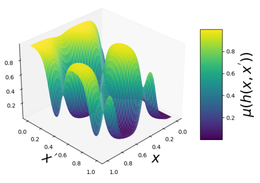
$$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right)\exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + \exp(1)$$

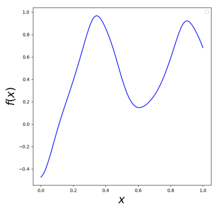
The preference function h (difference in utilities) is then scaled to the range $[-3,3]$. The Ackley function is shown in Figure C.1.

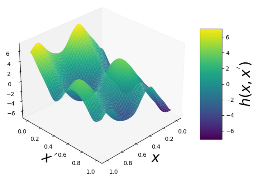
Yelp Dataset: We use a subset of the Yelp Dataset, filtering it to include only restaurants in Philadelphia, USA, with at least 500 reviews and users who review at least 90 restaurants. The final dataset consists of 275 restaurants, 20 users, and 2563 reviews. Reviews for each restaurant are concatenated and processed using OpenAI's `TEXT-EMBEDDING-3-LARGE` model to generate 32-dimensional vector embeddings, which serve as the action set in the BOHF framework. User ratings (ranging from 1 to 5) are considered as the utility function f, which are then scaled to the range $[-3,3]$. Missing ratings are handled using collaborative filtering. In each experimental run, we sample a random user from the set of 20 and conduct the experiment independently. We average the regret over 60 runs to produce the final plot.

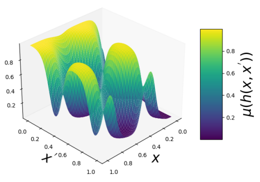

(a) $f(x)$, SE kernel

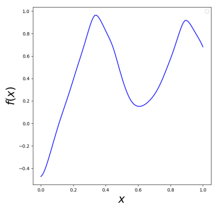

(b) $h(x,x')$, SE kernel

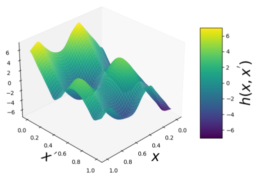

(c) $\mu(h(x,x'))$, SE kernel

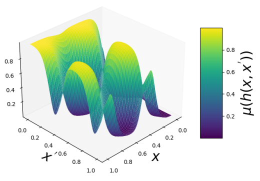

(d) $f(x)$, Matérn ($\nu = 2.5$)

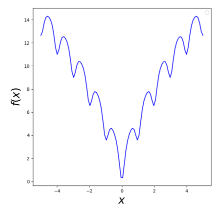

(e) $h(x,x')$, Matérn ($\nu = 2.5$)

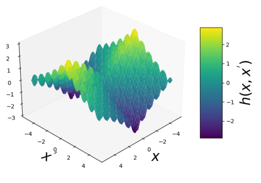

(f) $\mu(h(x,x'))$, Matérn ($\nu = 2.5$)


(g) $f(x)$, Matérn ($\nu = 1.5$)


(h) $h(x,x')$, Matérn ($\nu = 1.5$)


(i) $\mu(h(x,x'))$, Matérn ($\nu = 1.5$)


(j) $f(x)$, Ackley function


(k) $h(x,x')$, Ackley function

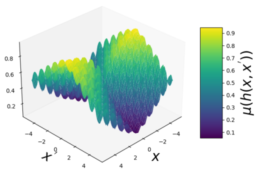

(l) $\mu(h(x,x'))$, Ackley function

Figure C.1: Plots of the utility function $f(x)$, the preference function $h(x,x') = f(x) - f(x')$, and the probability of preference $\mu(h(x,x'))$ for synthetic experiments. The rows correspond to: **(1st row)** SE kernel (RKHS), **(2nd row)** Matérn kernel with $\nu = 2.5$ (RKHS), **(3rd row)** Matérn kernel with $\nu = 1.5$ (RKHS), and **(4th row)** Ackley function.

Loss function optimization: To minimize the loss function given in (6.7) and obtain the parameters $\boldsymbol{\theta}$, any standard optimization algorithm can be used. In our experiments, we employ gradient descent. The learning rate is individually tuned for each algorithm, kernel, and test function by selecting the best-performing value from the grid $\{0.01, 0.005, 0.001, 0.0005, 0.0001\}$ in each scenario.

Hyperparameters: We choose $l = 0.1$ as the length scale and $\lambda = 0.05$ as the kernel-based learning parameter across all cases. The horizon T is set to 300 for RKHS test functions and 2000 for the Ackley function and the Yelp Dataset. For the RKHS and Ackley functions, the confidence interval width β is fixed at 1 for both MR-LPF and MaxMinLCB. For the Yelp dataset, we conduct a grid search to tune β over $\{0.01, 0.1, 0.5, 1, 2\}$ for both MR-LPF and MaxMinLCB algorithms. We determine $\beta = 2$ as optimal for MaxMinLCB and $\beta = 0.1$ for MR-LPF.

Computational Resources: For the experiments with the synthetic RKHS and Ackley functions, we utilize the Scikit-Learn library [269] for implementing Gaussian Process (GP) regression. The code is executed on a cluster with 376.2 GiB of RAM and an Intel(R) Xeon(R) Gold 5118 CPU running at 2.30 GHz. In the case of the Yelp Dataset experiments, we use the BoTorch library [273] and its dependencies, including GPyTorch [274], which offer efficient GP regression tools with GPU support. The simulations are carried out on a computing node equipped with an NVIDIA GeForce RTX 2080 Ti GPU featuring 11 GB of VRAM, an Intel(R) Xeon(R) Gold 5118 CPU running at 2.40 GHz with 24 cores, and 92 GB of RAM.

MaxMinLCB algorithm: [116] proposed a zero-sum Stackelberg (Leader–Follower) game for action selection, where the leader x_t maximizes the lower confidence bound (LCB), and the follower x'_t minimizes it, according to the following:

$$
\begin{aligned}
x_t &= \arg\max_{x \in \mathcal{M}_t} \mu(h_t(x, x'(x)) - \beta_t \sigma_t(x, x'(x)),\\
x'(x) &= \arg\min_{x' \in \mathcal{M}_t} \mu(h_t(x, x')) - \beta_t \sigma_t(x, x').
\end{aligned}
$$

Bibliography

[1] Scott Mayer McKinney, Marcin Sieniek, Varun Godbole, Jonathan Godwin, Natasha Antropova, Hutan Ashrafian, et al. International evaluation of an AI system for breast cancer screening. *Nature*, 577(7788):89–94, 2020.

[2] Prabal Datta Barua, Jahmunah Vicnesh, Raj Gururajan, Shu Lih Oh, Elizabeth Palmer, Muhammad Mokhzaini Azizan, et al. Artificial intelligence enabled personalised assistive tools to enhance education of children with neurodevelopmental disorders—a review. *International Journal of Environmental Research and Public Health*, 19(3):1192, 2022.

[3] Qingkai Kong, Daniel T Trugman, Zachary E Ross, Michael J Bianco, Brendan J Meade, and Peter Gerstoft. Machine learning in seismology: Turning data into insights. *Seismological Research Letters*, 90(1):3–14, 2019.

[4] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[5] Shirin Akbarinasaji, Can Kavaklioglu, Ayşe Başar, and Adam Neal. Partially observable markov decision process to generate policies in software defect management. *Journal of Systems and Software*, 163:110518, 2020.

[6] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40:e253, 2017.

[7] Andreas Holzinger, Anna Saranti, Alessa Angerschmid, Bettina Finzel, Ute Schmid, and Heimo Mueller. Toward human-level concept learning: Pattern benchmarking for AI algorithms. *Patterns*, 4(8), 2023.

[8] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[9] Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadami Barekatain, et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022.

[10] Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

[11] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, et al. Deepseek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[12] Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, et al. Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021.

[13] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289. PMLR, 2019.

[14] Aimee Van Wynsberghe. Sustainable AI: AI for sustainability and the sustainability of AI. *AI and Ethics*, 1(3):213–218, 2021.

[15] Ben Cottier, Robi Rahman, Loredana Fattorini, Nestor Maslej, Tamay Besiroglu, and David Owen. The rising costs of training frontier AI models. *arXiv preprint arXiv:2405.21015*, 2024.

[16] Kate Gibson. AI data startup turing triples revenue to $300 million. Reuters, 2025.

[17] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is Q-learning provably efficient? *Advances in Neural Information Processing Systems*, 31, 2018.

[18] Peter Auer, Thomas Jaksch, and Ronald Ortner. Near-optimal regret bounds for reinforcement learning. *Advances in Neural Information Processing Systems*, 21, 2008.

[19] Peter L Bartlett and Ambuj Tewari. Regal: A regularization based algorithm for reinforcement learning in weakly communicating MDPs. *arXiv preprint arXiv:1205.2661*, 2012.

[20] Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*, pages 2137–2143. PMLR, 2020.

[21] Hengshuai Yao, Csaba Szepesvári, Bernardo Avila Pires, and Xinhua Zhang. Pseudo-MDPs and factored linear action models. In *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pages 1–9. IEEE, 2014.

[22] Daniel Russo. Worst-case regret bounds for exploration via randomized value functions. *Advances in Neural Information Processing Systems*, 32, 2019.

[23] Andrea Zanette, David Brandfonbrener, Emma Brunskill, Matteo Pirotta, and Alessandro Lazaric. Frequentist regret bounds for ran-

domized least-squares value iteration. In *International Conference on Artificial Intelligence and Statistics*, pages 1954–1964. PMLR, 2020.

[24] Gergely Neu and Ciara Pike-Burke. A unifying view of optimism in episodic reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1392–1403, 2020.

[25] Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 85:1–22, 2022.

[26] Eduardo Pignatelli, Johan Ferret, Matthieu Geist, Thomas Mesnard, Hado van Hasselt, and Laura Toni. A survey of temporal credit assignment in deep reinforcement learning. *Transactions on Machine Learning Research*, 2024. Survey Certification.

[27] Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76:201–264, 2023.

[28] Richard Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, pages 679–684, 1957.

[29] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

[30] Dimitri Bertsekas and John N Tsitsiklis. *Neuro-dynamic programming.* Athena Scientific, 1996.

[31] Zhuoran Yang, Chi Jin, Zhaoran Wang, Mengdi Wang, and Michael Jordan. Provably efficient reinforcement learning with kernel and neural function approximations. *Advances in Neural Information Processing Systems*, 33:13903–13916, 2020.

[32] Sattar Vakili. Open problem: Order optimal regret bounds for kernel-based reinforcement learning. In *The 37th Annual Conference on Learning Theory*, pages 5340–5344. PMLR, 2024.

[33] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: Support vector machines, regularization, optimization, and beyond.* MIT press, 2002.

[34] Kyowoon Lee, Sol-A Kim, Jaesik Choi, and Seong-Whan Lee. Deep reinforcement learning in continuous action spaces: a case study in the game of simulated curling. In *International Conference on Machine Learning*, pages 2937–2946. PMLR, 2018.

[35] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[36] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pages 651–673. PMLR, 2018.

[37] Gregory Kahn, Adam Villaflor, Vitchyr Pong, Pieter Abbeel, and Sergey Levine. Uncertainty-aware reinforcement learning for collision avoidance. *arXiv preprint arXiv:1702.01182*, 2017.

[38] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.

[39] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, et al. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[40] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, et al. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937. PMLR, 2016.

[41] Yuhuai Wu, Elman MansimovShun, Shun Liao, Alec Radford, and John Schulman. OpenAI baselines: ACKTR & A2C. `https://openai.com/blog/baselines-acktr-a2c/`, 2017.

[42] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[43] Aditya Mahajan and Demosthenis Teneketzis. Multi-armed bandit problems. In *Foundations and Applications of Sensor Management*, pages 121–151. Springer, 2008.

[44] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.

[45] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multi-armed bandit problem. *Machine Learning*, 47:235–256, 2002.

[46] Robert David Kleinberg. *Online decision problems with large strategy sets*. PhD thesis, Massachusetts Institute of Technology, 2005.

[47] Sébastien Bubeck, Nicolo Cesa-Bianchi, and Gábor Lugosi. Bandits with heavy tail. *IEEE Transactions on Information Theory*, 59(11):7711–7717, 2013.

[48] Sattar Vakili, Keqin Liu, and Qing Zhao. Deterministic sequencing of exploration and exploitation for multi-armed bandit problems. *IEEE Journal of Selected Topics in Signal Processing*, 7(5):759–767, 2013.

[49] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

[50] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on Learning Theory*, pages 39–1. JMLR Workshop and Conference Proceedings, 2012.

[51] Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. Thompson sampling: an asymptotically optimal finite-time analysis. In *International Conference on Algorithmic Learning Theory*, pages 199–213. Springer, 2012.

[52] Peter Auer. Using confidence bounds for exploitation-exploration tradeoffs. *Journal of Machine Learning Research*, 3:397–422, 2002.

[53] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, pages 661–670, 2010.

[54] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pages 208–214. JMLR Workshop and Conference Proceedings, 2011.

[55] Varsha Dani, Thomas P Hayes, and Sham M Kakade. Stochastic linear optimization under bandit feedback. In *Proceedings of the 21st Annual Conference on Learning Theory*, pages 355–366, 2008.

[56] Yasin Abbasi-Yadkori, András Antos, and Csaba Szepesvári. Forced-exploration based algorithms for playing in stochastic linear bandits. In *COLT Workshop on Online Learning with Limited Feedback*, volume 92, page 236, 2009.

[57] Paat Rusmevichientong and John N Tsitsiklis. Linearly parameterized bandits. *Mathematics of Operations Research*, 35(2):395–411, 2010.

[58] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. *Advances in Neural Information Processing Systems*, 24, 2011.

[59] Victor H Peña, Michael Klass, and Tze Leung Lai. Self-normalized processes: exponential inequalities, moment bounds and iterated logarithm laws. *The Annals of Probability*, 32(3):1902–1933, 2004.

[60] Victor H Peña, Tze Leung Lai, and Qi-Man Shao. *Self-normalized processes: limit theory and statistical applications*. Springer Science & Business Media, 2008.

[61] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *International Conference on Machine Learning*, pages 127–135. PMLR, 2013.

[62] Marc Abeille and Alessandro Lazaric. Linear thompson sampling revisited. In *International Conference on Artificial Intelligence and Statistics*, pages 176–184. PMLR, 2017.

[63] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[64] Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur. Gaussian processes and kernel methods: A review on connections and equivalences. *arXiv preprint arXiv:1807.02582*, 2018.

[65] Jonathan Scarlett, Ilija Bogunovic, and Volkan Cevher. Lower bounds on regret for noisy gaussian process bandit optimization. In *Conference on Learning Theory*, pages 1723–1742. PMLR, 2017.

[66] Sayak Ray Chowdhury and Aditya Gopalan. On kernelized multi-armed bandits. In *International Conference on Machine Learning*, pages 844–853. PMLR, 2017.

[67] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning*, 2010.

[68] Sattar Vakili, Kia Khezeli, and Victor Picheny. On information gain and regret bounds in gaussian process bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 82–90. PMLR, 2021.

[69] Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc Peter Deisenroth. Matérn gaussian processes on Riemannian manifolds. *Advances in Neural Information Processing Systems*, 33:12426–12437, 2020.

[70] Sattar Vakili, Nacime Bouziani, Sepehr Jalali, Alberto Bernacchia, and Da-shan Shiu. Optimal order simple regret for gaussian process bandits. *Advances in Neural Information Processing Systems*, 34:21202–21215, 2021.

[71] Sattar Vakili, Jonathan Scarlett, and Tara Javidi. Open problem: Tight online confidence intervals for RKHS elements. In *Conference on Learning Theory*, pages 4647–4652. PMLR, 2021.

[72] Justin Whitehouse, Aaditya Ramdas, and Steven Z Wu. On the sublinear regret of GP-UCB. *Advances in Neural Information Processing Systems*, 36, 2023.

[73] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

[74] Hung Tran-The, Sunil Gupta, Santu Rana, and Svetha Venkatesh. Regret bounds for expected improvement algorithms in gaussian process bandit optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 8715–8737. PMLR, 2022.

[75] Vu Nguyen, Sunil Gupta, Santu Rana, Cheng Li, and Svetha Venkatesh. Regret for expected improvement over the best-observed value and stopping condition. In *Asian Conference on Machine Learning*, pages 279–294. PMLR, 2017.

[76] Matthew Hoffman, Eric Brochu, and Nando De Freitas. Portfolio allocation for bayesian optimization. In *Conference on Uncertainty in Artificial Intelligence*, volume 11, pages 327–336, 2011.

[77] Sattar Vakili. Open problem: Regret bounds for noise-free kernel-based bandits. In *Conference on Learning Theory*, pages 5624–5629. PMLR, 2022.

[78] Madison Lee, Shubhanshu Shekhar, and Tara Javidi. Multi-scale zero-order optimization of smooth functions in an RKHS. In *IEEE International Symposium on Information Theory (ISIT)*, pages 288–293. IEEE, 2022.

[79] David Janz, David Burt, and Javier González. Bandit optimisation of functions in the matérn kernel RKHS. In *International Conference on Artificial Intelligence and Statistics*, pages 2486–2495. PMLR, 2020.

[80] Michal Valko, Nathan Korda, Rémi Munos, Ilias Flaounas, and Nello Cristianini. Finite-time analysis of kernelised contextual bandits. In *Conference on Uncertainty in Artificial Intelligence*, 2013.

[81] Daniele Calandriello, Luigi Carratino, Alessandro Lazaric, Michal Valko, and Lorenzo Rosasco. Gaussian process optimization with adaptive sketching: scalable and no regret. In *Conference on Learning Theory*, pages 533–557. PMLR, 2019.

[82] Romain Camilleri, Kevin Jamieson, and Julian Katz-Samuels. High-dimensional experimental design and kernel bandits. In *International Conference on Machine Learning*, pages 1227–1237. PMLR, 2021.

[83] Sudeep Salgia, Sattar Vakili, and Qing Zhao. A domain-shrinking based bayesian optimization algorithm with order-optimal regret performance. *Advances in Neural Information Processing Systems*, 34:28836–28847, 2021.

[84] Zihan Li and Jonathan Scarlett. Gaussian process bandit optimization with few batches. In *International Conference on Artificial Intelligence and Statistics*, pages 92–107. PMLR, 2022.

[85] Ilija Bogunovic, Jonathan Scarlett, Andreas Krause, and Volkan Cevher. Truncated variance reduction: A unified approach to bayesian optimization and level-set estimation. *Advances in Neural Information Processing Systems*, 29, 2016.

[86] Xi Chen, Paul N Bennett, Kevyn Collins-Thompson, and Eric Horvitz. Pairwise ranking aggregation in a crowdsourced setting. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining*, pages 193–202, 2013.

[87] Yisong Yue and Thorsten Joachims. Interactively optimizing information retrieval systems as a dueling bandits problem. In *Proceedings of the 26th International Conference on Machine Learning*, pages 1201–1208, 2009.

[88] Yanan Sui and Joel Burdick. Clinical online recommendation with subgroup rank feedback. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pages 289–292, 2014.

[89] Tom Minka, Ryan Cleven, and Yordan Zaykov. Trueskill 2: An improved bayesian skill rating system. *Technical Report*, 2018.

[90] Viktor Bengs, Róbert Busa-Fekete, Adil El Mesaoudi-Paul, and Eyke Hüllermeier. Preference-based online learning with dueling bandits: A survey. *Journal of Machine Learning Research*, 22(7):1–108, 2021.

[91] Tanguy Urvoy, Fabrice Clerot, Raphael Féraud, and Sami Naamane. Generic exploration and k-armed voting bandits. In *International Conference on Machine Learning*, pages 91–99. PMLR, 2013.

[92] Yue Wu, Tao Jin, Hao Lou, Farzad Farnoud, and Quanquan Gu. Borda regret minimization for generalized linear dueling bandits. *arXiv preprint arXiv:2303.08816*, 2023.

[93] Miroslav Dudík, Katja Hofmann, Robert E Schapire, Aleksandrs Slivkins, and Masrour Zoghi. Contextual dueling bandits. In *Conference on Learning Theory*, pages 563–587. PMLR, 2015.

[94] Yisong Yue, Josef Broder, Robert Kleinberg, and Thorsten Joachims. The k-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556, 2012.

[95] Yisong Yue and Thorsten Joachims. Beat the mean bandit. In *Proceedings of the 28th International Conference on Machine Learning*, pages 241–248, 2011.

[96] Masrour Zoghi, Shimon Whiteson, Maarten De Rijke, and Rémi Munos. Relative confidence sampling for efficient online ranker evaluation. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, pages 73–82, 2014.

[97] Masrour Zoghi, Shimon Whiteson, Rémi Munos, and Maarten De Rijke. Relative upper confidence bound for the k-armed dueling bandit problem. In *International Conference on Machine Learning*, pages 10–18. PMLR, 2014.

[98] Nir Ailon, Zohar Karnin, and Thorsten Joachims. Reducing dueling bandits to cardinal bandits. In *International Conference on Machine Learning*, pages 856–864. PMLR, 2014.

[99] Masrour Zoghi, Shimon Whiteson, and Maarten De Rijke. MergeRUCB: a method for large-scale online ranker evaluation. In *Proceedings of the 8th ACM International Conference on Web Search and Data Mining*, pages 17–26, 2015.

[100] Moein Falahatgar, Yi Hao, Alon Orlitsky, Venkatadheeraj Pichapati, and Vaishakh Ravindrakumar. Maxing and ranking with few assumptions. *Advances in Neural Information Processing Systems*, 30, 2017.

[101] Moein Falahatgar, Ayush Jain, Alon Orlitsky, Venkatadheeraj Pichapati, and Vaishakh Ravindrakumar. The limits of maxing, ranking, and preference learning. In *International Conference on Machine Learning*, pages 1427–1436. PMLR, 2018.

[102] Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[103] Aadirupa Saha and Akshay Krishnamurthy. Efficient and optimal algorithms for contextual dueling bandits under realizability. In *International Conference on Algorithmic Learning Theory*, pages 968–994. PMLR, 2022.

[104] Xuheng Li, Heyang Zhao, and Quanquan Gu. Feel-good thompson sampling for contextual dueling bandits. In *The 41st International Conference on Machine Learning*, 2024.

[105] Viktor Bengs, Aadirupa Saha, and Eyke Hüllermeier. Stochastic contextual dueling bandits under linear stochastic transitivity models. In *International Conference on Machine Learning*, pages 1764–1786. PMLR, 2022.

[106] Qiwei Di, Tao Jin, Yue Wu, Heyang Zhao, Farzad Farnoud, and Quanquan Gu. Variance-aware regret bounds for stochastic contextual dueling bandits. In *The 12th International Conference on Learning Representations*, 2024.

[107] Aadirupa Saha. Optimal algorithms for stochastic contextual preference bandits. *Advances in Neural Information Processing Systems*, 34:30050–30062, 2021.

[108] Aadirupa Saha, Aldo Pacchiano, and Jonathan Lee. Dueling RL: Reinforcement learning with trajectory preferences. In *International Conference on Artificial Intelligence and Statistics*, pages 6263–6289. PMLR, 2023.

[109] Wenhao Zhan, Masatoshi Uehara, Nathan Kallus, Jason D. Lee, and Wen Sun. Provable offline preference-based reinforcement learning. In *The 12th International Conference on Learning Representations*, 2024.

[110] Banghua Zhu, Michael Jordan, and Jiantao Jiao. Principled reinforcement learning with human feedback from pairwise or k-wise comparisons. In *International Conference on Machine Learning*, pages 43037–43067. PMLR, 2023.

[111] Xiang Ji, Huazheng Wang, Minshuo Chen, Tuo Zhao, and Mengdi Wang. Provable benefits of policy learning from human preferences in contextual bandit problems. *arXiv preprint arXiv:2307.12975*, 2023.

[112] Rémi Munos, Michal Valko, Daniele Calandriello, Mohammad Gheshlaghi Azar, Mark Rowland, Zhaohan Daniel Guo, et al. Nash learning from human feedback. *arXiv preprint arXiv:2312.00886*, 18, 2023.

[113] Yichong Xu, Aparna Joshi, Aarti Singh, and Artur Dubrawski. Zeroth order non-convex optimization with dueling-choice bandits. In *Conference on Uncertainty in Artificial Intelligence*, pages 899–908. PMLR, 2020.

[114] Viraj Mehta, Vikramjeet Das, Ojash Neopane, Yijia Dai, Ilija Bogunovic, Jeff Schneider, and Willie Neiswanger. Sample efficient reinforcement learning from human feedback via active exploration. *arXiv preprint arXiv:2312.00267*, 2023.

[115] Viraj Mehta, Ojash Neopane, Vikramjeet Das, Sen Lin, Jeff Schneider, and Willie Neiswanger. Kernelized offline contextual dueling bandits. In *ICML Workshop on The Many Facets of Preference-Based Learning*, 2023.

[116] Barna Pásztor, Parnian Kassraie, and Andreas Krause. Bandits with preference feedback: A stackelberg game perspective. In *Advances in Neural Information Processing Systems*, 2024.

[117] Wenjie Xu, Wenbin Wang, Yuning Jiang, Bratislav Svetozarevic, and Colin Jones. Principled preferential bayesian optimization. In *International Conference on Machine Learning*, pages 55305–55336. PMLR, 2024.

[118] Arun Verma, Zhongxiang Dai, Xiaoqiang Lin, Patrick Jaillet, and Bryan Kian Hsiang Low. Neural dueling bandits: Preference-based optimization with human feedback. In *The 13th International Conference on Learning Representations*, 2025.

[119] David Abel, Cameron Allen, Dilip Arumugam, D Ellis Hershkowitz, Michael L Littman, and Lawson LS Wong. Bad-policy density: A measure of reinforcement learning hardness. *arXiv preprint arXiv:2110.03424*, 2021.

[120] Stephen Zhen Gou and Yuyang Liu. DQN with model-based exploration: Efficient learning on environments with sparse rewards. *arXiv preprint arXiv:1903.09295*, 2019.

[121] Marco Alexander Wiering. *Explorations in efficient reinforcement learning*. PhD thesis, University of Amsterdam, 1999.

[122] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, and Joelle Pineau. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4):219–354, 2018.

[123] Susan Amin, Maziar Gomrokchi, Harsh Satija, Herke van Hoof, and Doina Precup. A survey of exploration methods in reinforcement learning. *arXiv preprint arXiv:2109.00157*, 2021.

[124] Arthur Aubret, Laetitia Matignon, and Salima Hassas. A survey on intrinsic motivation in reinforcement learning. *arXiv preprint arXiv:1908.06976*, 2019.

[125] Satinder Singh, Richard L Lewis, Andrew G Barto, and Jonathan Sorg. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, 2010.

[126] Nuttapong Chentanez, Andrew Barto, and Satinder Singh. Intrinsically motivated reinforcement learning. *Advances in Neural Information Processing Systems*, 17, 2004.

[127] Richard M Ryan and Edward L Deci. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist*, 55(1):68–78, 2000.

[128] Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in Neurorobotics*, 1:6, 2007.

[129] Cédric Colas, Tristan Karch, Olivier Sigaud, and Pierre-Yves Oudeyer. Autotelic agents with intrinsically motivated goal-conditioned reinforcement learning: A short survey. *Journal of Artificial Intelligence Research*, 74:1159–1199, 2022.

[130] Nazmul Siddique, Paresh Dhakan, Inaki Rano, and Kathryn Merrick. A review of the relationship between novelty, intrinsic motivation and reinforcement learning. *Paladyn, Journal of Behavioral Robotics*, 8(1):58–69, 2017.

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

[132] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, et al. # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 30, 2017.

[133] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[134] Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, et al. Never give up: Learning directed exploration strategies. *arXiv preprint arXiv:2002.06038*, 2020.

[135] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, pages 2778–2787. PMLR, 2017.

[136] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. VIME: Variational information maximizing exploration. *Advances in Neural Information Processing Systems*, 29, 2016.

[137] Arthur Aubret, Laetitia Matignon, and Salima Hassas. An information-theoretic perspective on intrinsic motivation in reinforcement learning: A survey. *Entropy*, 25(2):327, 2023.

[138] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

[139] Andrew Cohen, Lei Yu, Xingye Qiao, and Xiangrong Tong. Maximum entropy diverse exploration: Disentangling maximum entropy reinforcement learning. *arXiv preprint arXiv:1911.00828*, 2019.

[140] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

[141] Matteo Bettini, Ryan Kortvelesy, and Amanda Prorok. Controlling behavioral diversity in multi-agent reinforcement learning. *arXiv preprint arXiv:2405.15054*, 2024.

[142] Takayuki Osa and Tatsuya Harada. Discovering multiple solutions from a single task in offline reinforcement learning. *arXiv preprint arXiv:2406.05993*, 2024.

[143] Saurabh Kumar, Aviral Kumar, Sergey Levine, and Chelsea Finn. One solution is not all you need: Few-shot extrapolation via structured MaxEnt RL. *Advances in Neural Information Processing Systems*, 33:8198–8210, 2020.

[144] Tom Zahavy, Yannick Schroecker, Feryal Behbahani, Kate Baumli, Sebastian Flennerhag, Shaobo Hou, and Satinder Singh. Discovering policies with domino: Diversity optimization maintaining near optimality. *arXiv preprint arXiv:2205.13521*, 2022.

[145] Luca Grillotti, Maxence Faldor, Borja G León, and Antoine Cully. Quality-diversity actor-critic: Learning high-performing and diverse behaviors via value and successor features critics. *arXiv preprint arXiv:2403.09930*, 2024.

[146] Kevin R McKee, Joel Z Leibo, Charlie Beattie, and Richard Everett. Quantifying the effects of environment and population diversity in multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 36(1):21, 2022.

[147] Wentse Chen, Shiyu Huang, Yuan Chiang, Tim Pearce, Wei-Wei Tu, Ting Chen, and Jun Zhu. DGPO: discovering multiple strategies with diversity-guided policy optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 11390–11398, 2024.

[148] Geoffrey Cideron, Andrea Agostinelli, Johan Ferret, Sertan Girgin, Romuald Elie, Olivier Bachem, et al. Diversity-rewarded CFG distillation. *arXiv preprint arXiv:2410.06084*, 2024.

[149] Jianye Hao, Tianpei Yang, Hongyao Tang, Chenjia Bai, Jinyi Liu, Zhaopeng Meng, et al. Exploration in deep reinforcement learning: From single-agent to multi-agent domain. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[150] Alain Andres, Esther Villar-Rodriguez, and Javier Del Ser. An evaluation study of intrinsic motivation techniques applied to reinforcement learning over hard exploration environments. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 201–220. Springer, 2022.

[151] Adrien Ali Taiga, William Fedus, Marlos C Machado, Aaron Courville, and Marc G Bellemare. On bonus-based exploration methods in the arcade learning environment. *arXiv preprint arXiv:2109.11052*, 2021.

[152] Mingqi Yuan, Roger Creus Castanyer, Bo Li, Xin Jin, Glen Berseth, and Wenjun Zeng. Rlexplore: Accelerating research in intrinsically-motivated reinforcement learning. *arXiv preprint arXiv:2405.19548*, 2024.

[153] Michael Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, et al. URLB: Unsupervised reinforcement learning benchmark. *arXiv preprint arXiv:2110.15191*, 2021.

[154] Kaixin Wang, Kuangqi Zhou, Bingyi Kang, Jiashi Feng, and Shuicheng Yan. Revisiting intrinsic reward for exploration in procedurally generated environments. In *The 11th International Conference on Learning Representations*, 2022.

[155] Mikael Henaff, Minqi Jiang, and Roberta Raileanu. A study of global and episodic bonuses for exploration in contextual MDPs. *arXiv preprint arXiv:2306.03236*, 2023.

[156] Toru Lin and Allan Jabri. MIMEx: Intrinsic rewards from masked input modeling. *Advances in Neural Information Processing Systems*, 36, 2024.

[157] Tom Zahavy, Brendan O'Donoghue, Guillaume Desjardins, and Satinder Singh. Reward is enough for convex MDPs. *Advances in Neural Information Processing Systems*, 34:25746–25759, 2021.

[158] Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, et al. Minigrid & Miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, abs/2306.13831, 2023.

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

[160] Roberta Raileanu and Tim Rocktäschel. RIDE: Rewarding impact-driven exploration for procedurally-generated environments. *arXiv preprint arXiv:2002.12292*, 2020.

[161] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

[162] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[163] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *International Conference on Machine Learning*, pages 5062–5071. PMLR, 2019.

[164] Younggyo Seo, Lili Chen, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. State entropy maximization with random encoders for efficient exploration. In *International Conference on Machine Learning*, pages 9443–9454. PMLR, 2021.

[165] Mikael Henaff, Roberta Raileanu, Minqi Jiang, and Tim Rocktäschel. Exploration via elliptical episodic bonuses. *Advances in Neural Information Processing Systems*, 35:37631–37646, 2022.

[166] Mikayel Samvelyan, Robert Kirk, Vitaly Kurin, Jack Parker-Holder, Minqi Jiang, Eric Hambro, et al. MiniHack the planet: A sandbox for open-ended reinforcement learning research. In *The 35th Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[167] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

[168] Jingbin Liu, Xinyang Gu, and Shuai Liu. Policy optimization reinforcement learning with entropy regularization. *arXiv preprint arXiv:1912.01557*, 2019.

[169] Andrew Kachites McCallum. *Reinforcement learning with selective perception and hidden state.* PhD thesis, University of Rochester, 1996.

[170] Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. In *International Conference on Machine Learning*, pages 151–160. PMLR, 2019.

[171] Zhuang Liu, Xuanlin Li, Kang Bingyi, and Trevor Darrell. Regularization matters in policy optimization. *arXiv preprint arXiv:1910.09191*, 2019.

[172] Seungyul Han and Youngchul Sung. A max-min entropy framework for reinforcement learning. *Advances in Neural Information Processing Systems*, 34:25732–25745, 2021.

[173] Rushuai Yang, Chenjia Bai, Hongyi Guo, Siyuan Li, Bin Zhao, Zhen Wang, Peng Liu, and Xuelong Li. Behavior contrastive learning for unsupervised skill discovery. In *International Conference on Machine Learning*, pages 39183–39204. PMLR, 2023.

[174] Paul-Antoine Le Tolguenec, Yann Besse, Florent Teichteil-Konigsbuch, Dennis G Wilson, and Emmanuel Rachelson. Exploration by learning diverse skills through successor state measures. *arXiv preprint arXiv:2406.10127*, 2024.

[175] Hyunseung Kim, Byung Kun Lee, Hojoon Lee, Dongyoon Hwang, Sejik Park, Kyushik Min, et al. Learning to discover skills through guidance. *Advances in Neural Information Processing Systems*, 36, 2024.

[176] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[177] Minqi Jiang, Michael Dennis, Edward Grefenstette, and Tim Rocktäschel. Minimax: Efficient baselines for autocurricula in jax. *arXiv preprint arXiv:2311.12716*, 2023.

[178] Zhixuan Lin, Pierluca D'Oro, Evgenii Nikishin, and Aaron Courville. The curse of diversity in ensemble-based exploration. *arXiv preprint arXiv:2405.04342*, 2024.

[179] Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

[180] Cédric Colas, Pierre Fournier, Mohamed Chetouani, Olivier Sigaud, and Pierre-Yves Oudeyer. Curious: Intrinsically motivated modular multi-goal reinforcement learning. In *International Conference on Machine Learning*, pages 1331–1340. PMLR, 2019.

[181] Sattar Vakili and Julia Olkhovskaya. Kernelized reinforcement learning with order optimal regret bounds. *Advances in Neural Information Processing Systems*, 36, 2023.

[182] Sayak Ray Chowdhury and Rafael Oliveira. Value function approximations via kernel embeddings for no-regret reinforcement learning. In *Asian Conference on Machine Learning*, pages 249–264. PMLR, 2023.

[183] Chi Jin, Akshay Krishnamurthy, Max Simchowitz, and Tiancheng Yu. Reward-free exploration for reinforcement learning. In *International Conference on Machine Learning*, pages 4870–4879. PMLR, 2020.

[184] Ruosong Wang, Simon S Du, Lin Yang, and Russ R Salakhutdinov. On reward-free reinforcement learning with linear function approximation. *Advances in Neural Information Processing Systems*, 33:17816–17826, 2020.

[185] Shuang Qiu, Jieping Ye, Zhaoran Wang, and Zhuoran Yang. On reward-free RL with kernel and neural function approximations: Single-agent MDP and markov game. In *International Conference on Machine Learning*, pages 8737–8747. PMLR, 2021.

[186] Doina Precup. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, page 80, 2000.

[187] András Antos, Csaba Szepesvári, and Rémi Munos. Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71:89–129, 2008.

[188] Rémi Munos and Csaba Szepesvári. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9(5), 2008.

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

[190] Tengyang Xie, Ching-An Cheng, Nan Jiang, Paul Mineiro, and Alekh Agarwal. Bellman-consistent pessimism for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34:6683–6694, 2021.

[191] Jinglin Chen and Nan Jiang. Information-theoretic considerations in batch reinforcement learning. In *International Conference on Machine Learning*, pages 1042–1051. PMLR, 2019.

[192] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International Conference on Machine Learning*, pages 22–31. PMLR, 2017.

[193] Eitan Altman. *Constrained markov decision processes*. Routledge, 2021.

[194] Chen Tessler, Daniel J Mankowitz, and Shie Mannor. Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074*, 2018.

[195] Pihe Hu, Yu Chen, and Longbo Huang. Towards minimax optimal reward-free reinforcement learning in linear MDPs. In *The 11th International Conference on Learning Representations*, 2022.

[196] Andrew J Wagenmaker, Yifang Chen, Max Simchowitz, Simon Du, and Kevin Jamieson. Reward-free RL is no harder than reward-aware RL in linear markov decision processes. In *International Conference on Machine Learning*, pages 22430–22456. PMLR, 2022.

[197] Joongkyu Lee and Min-hwan Oh. Demystifying linear MDPs and novel dynamics aggregation framework. In *The 12th International Conference on Learning Representations*, 2023.

[198] Sattar Vakili, Farhang Nabiei, Da-shan Shiu, and Alberto Bernacchia. Reward-free kernel-based reinforcement learning. In *41st International Conference on Machine Learning*, 2024.

[199] Sham Machandranath Kakade. *On the sample complexity of reinforcement learning.* University of London, University College London (United Kingdom), 2003.

[200] Michael Kearns and Satinder Singh. Finite-sample convergence rates for Q-learning and indirect algorithms. In *Advances in Neural Information Processing Systems*, volume 11, 1998.

[201] Mohammad Gheshlaghi Azar, Rémi Munos, and Hilbert J Kappen. Minimax PAC bounds on the sample complexity of reinforcement learning with a generative model. *Machine Learning*, 91:325–349, 2013.

[202] Aaron Sidford, Mengdi Wang, Xian Wu, Lin Yang, and Yinyu Ye. Near-optimal time and sample complexities for solving markov decision processes with a generative model. *Advances in Neural Information Processing Systems*, 31, 2018.

[203] Aaron Sidford, Mengdi Wang, Xian Wu, and Yinyu Ye. Variance reduced value iteration and faster algorithms for solving markov decision processes. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 770–787. SIAM, 2018.

[204] Alekh Agarwal, Sham Kakade, and Lin F Yang. Model-based reinforcement learning with a generative model is minimax optimal. In *Conference on Learning Theory*, pages 67–83. PMLR, 2020.

[205] Lin Yang and Mengdi Wang. Sample-optimal parametric Q-learning using linearly additive features. In *International Conference on Machine Learning*, pages 6995–7004. PMLR, 2019.

[206] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in Neural Information Processing Systems*, 31, 2018.

[207] Sattar Vakili, Michael Bromberg, Jezabel Garcia, Da-shan Shiu, and Alberto Bernacchia. Information gain and uniform generalization bounds for neural kernel models. In *2023 IEEE International Symposium on Information Theory (ISIT)*, pages 555–560. IEEE, 2023.

[208] Michael Kearns and Satinder Singh. Finite-sample convergence rates for Q-learning and indirect algorithms. *Advances in Neural Information Processing Systems*, 11, 1998.

[209] Lin Yang and Mengdi Wang. Reinforcement learning in feature space: Matrix bandit, kernels, and regret bound. In *International Conference on Machine Learning*, pages 10746–10756. PMLR, 2020.

[210] Sayak Ray Chowdhury and Aditya Gopalan. Online learning in kernelized markov decision processes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3197–3205. PMLR, 2019.

[211] Omar Darwiche Domingues, Pierre Ménard, Matteo Pirotta, Emilie Kaufmann, and Michal Valko. Kernel-based reinforcement learning: A finite-time analysis. In *International Conference on Machine Learning*, pages 2783–2792. PMLR, 2021.

[212] Elad Hazan, Sham Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pages 2681–2691. PMLR, 2019.

[213] Yasin Abbasi-Yadkori. Online learning for linearly parametrized control problems. 2013.

[214] Tor Lattimore. A lower bound for linear and kernel regression with adaptive covariates. In *The 36th Annual Conference on Learning Theory*, pages 2095–2113. PMLR, 2023.

[215] Martin L Puterman. *Markov decision processes: Discrete stochastic dynamic programming.* John Wiley & Sons, 2014.

[216] Sing-Yuan Yeh, Fu-Chieh Chang, Chang-Wei Yueh, Pei-Yuan Wu, Alberto Bernacchia, and Sattar Vakili. Sample complexity of kernel-based Q-learning. In *International Conference on Artificial Intelligence and Statistics*, pages 453–469. PMLR, 2023.

[217] Sattar Vakili, Jonathan Scarlett, Da-shan Shiu, and Alberto Bernacchia. Improved convergence rates for sparse approximation methods in kernel-based learning. In *International Conference on Machine Learning*, pages 21960–21983. PMLR, 2022.

[218] Gellért Weisz, Philip Amortila, and Csaba Szepesvári. Exponential lower bounds for planning in MDPs with linearly-realizable optimal action-value functions. In *Algorithmic Learning Theory*, pages 1237–1264. PMLR, 2021.

[219] Simon S Du, Sham M Kakade, Ruosong Wang, and Lin F Yang. Is a good representation sufficient for sample efficient reinforcement learning? *arXiv preprint arXiv:1910.03016*, 2019.

[220] Yining Wang, Ruosong Wang, Simon S Du, and Akshay Krishnamurthy. Optimism in reinforcement learning with generalized linear function approximation. *arXiv preprint arXiv:1912.04136*, 2019.

[221] Xiaoqiang Lin, Zhongxiang Dai, Arun Verma, See-Kiong Ng, Patrick Jaillet, and Bryan Kian Hsiang Low. Prompt optimization with human feedback. In *ICML 2024 Workshop on Models of Human Feedback for AI Alignment*, 2024.

[222] Lichang Chen, Jiuhai Chen, Tom Goldstein, Heng Huang, and Tianyi Zhou. Instructzero: Efficient instruction optimization for black-box large language models. In *International Conference on Machine Learning*, pages 6503–6518. PMLR, 2024.

[223] Xiaoqiang Lin, Zhaoxuan Wu, Zhongxiang Dai, Wenyang Hu, Yao Shu, See-Kiong Ng, et al. Use your instinct: Instruction optimization using neural bandits coupled with transformers. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*, 2023.

[224] Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[225] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[226] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *Advances in Neural Information Processing Systems*, 32, 2019.

[227] Javier González, Zhenwen Dai, Andreas Damianou, and Neil D Lawrence. Preferential bayesian optimization. In *International Conference on Machine Learning*, pages 1282–1291. PMLR, 2017.

[228] Petrus Mikkola, Milica Todorović, Jari Järvi, Patrick Rinke, and Samuel Kaski. Projective preferential bayesian optimization. In *International Conference on Machine Learning*, pages 6884–6892. PMLR, 2020.

[229] Shion Takeno, Masahiro Nomura, and Masayuki Karasuyama. Towards practical preferential bayesian optimization with skew gaussian processes. In *International Conference on Machine Learning*, pages 33516–33533. PMLR, 2023.

[230] Masrour Zoghi, Shimon Whiteson, Rémi Munos, and Maarten Rijke. Relative upper confidence bound for the k-armed dueling bandit problem. In *International Conference on Machine Learning*, pages 10–18. PMLR, 2014.

[231] Moein Falahatgar, Alon Orlitsky, Venkatadheeraj Pichapati, and Ananda Theertha Suresh. Maximum selection and ranking under noisy comparisons. In *International Conference on Machine Learning*, pages 1088–1096. PMLR, 2017.

[232] Nirjhar Das, Souradip Chakraborty, Aldo Pacchiano, and Sayak Ray Chowdhury. Active preference optimization for sample efficient RLHF. In *ICML 2024 Workshop on Theoretical Foundations of Foundation Models*, 2024.

[233] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, and Andrea L Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. *Advances in Neural Information Processing Systems*, 26, 2013.

[234] Ellen Novoseller, Yibing Wei, Yanan Sui, Yisong Yue, and Joel Burdick. Dueling posterior sampling for preference-based reinforcement learning.

In *Conference on Uncertainty in Artificial Intelligence*, pages 1029–1038. PMLR, 2020.

[235] Yichong Xu, Ruosong Wang, Lin Yang, Aarti Singh, and Artur Dubrawski. Preference-based reinforcement learning with finite-time guarantees. *Advances in Neural Information Processing Systems*, 33:18784–18794, 2020.

[236] Runzhe Wu and Wen Sun. Making RL with preference-based feedback efficient via randomization. In *The 12th International Conference on Learning Representations*, 2024.

[237] Xiaoyu Chen, Han Zhong, Zhuoran Yang, Zhaoran Wang, and Liwei Wang. Human-in-the-loop: Provably efficient preference-based reinforcement learning with general function approximation. In *International Conference on Machine Learning*, pages 3773–3793. PMLR, 2022.

[238] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[239] Louis Faury, Marc Abeille, Clément Calauzènes, and Olivier Fercoq. Improved optimistic algorithms for logistic bandits. In *International Conference on Machine Learning*, pages 3052–3060. PMLR, 2020.

[240] Momin Jamil and Xin-She Yang. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150–194, 2013.

[241] Yiding Jiang, J Zico Kolter, and Roberta Raileanu. On the importance of exploration for generalization in reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[242] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[243] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in Neural Information Processing Systems*, 30, 2017.

[244] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, et al. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

[245] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

[246] Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, et al. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*, 2023.

[247] Shangmin Guo, Biao Zhang, Tianlin Liu, Tianqi Liu, Misha Khalman, Felipe Llinares, et al. Direct language model alignment from online AI feedback. *arXiv preprint arXiv:2402.04792*, 2024.

[248] Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, et al. RLHF workflow: From reward modeling to online RLHF. *arXiv preprint arXiv:2405.07863*, 2024.

[249] J Zico Kolter and Andrew Y Ng. Near-bayesian exploration in polynomial time. In *International Conference on Machine Learning*, pages 513–520, 2009.

[250] Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E Gonzalez, et al. NovelD: A simple yet effective exploration criterion. *Advances in Neural Information Processing Systems*, 34:25217–25230, 2021.

[251] Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

[252] Mirco Mutti, Lorenzo Pratissoli, and Marcello Restelli. A policy gradient method for task-agnostic exploration. In *4th Lifelong Machine Learning Workshop at ICML*, 2020.

[253] Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training. *Advances in Neural Information Processing Systems*, 34:18459–18473, 2021.

[254] Dongyoung Kim, Jinwoo Shin, Pieter Abbeel, and Younggyo Seo. Accelerating reinforcement learning with value-conditional state entropy exploration. *Advances in Neural Information Processing Systems*, 36, 2024.

[255] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

[256] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, et al. Agent57: Outperforming the Atari human benchmark. In *International Conference on Machine Learning*, pages 507–517. PMLR, 2020.

[257] Nikolay Savinov, Anton Raichuk, Raphaël Marinier, Damien Vincent, Marc Pollefeys, Timothy Lillicrap, et al. Episodic curiosity through reachability. *arXiv preprint arXiv:1810.02274*, 2018.

[258] Yiming Wang, Ming Yang, Renzhi Dong, Binbin Sun, Furui Liu, and Leong Hou U. Efficient potential-based exploration in reinforcement learning using inverse dynamic bisimulation metric. *Advances in Neural Information Processing Systems*, 36, 2024.

[259] Anjie Zhu, Peng-Fei Zhang, Ruihong Qiu, Zetao Zheng, Zi Huang, and Jie Shao. Abstract and explore: A novel behavioral metric with cyclic dynamics in reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17150–17158, 2024.

[260] Marlos C Machado, Marc G Bellemare, and Michael Bowling. Count-based exploration with the successor representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5125–5133, 2020.

[261] Changmin Yu, Neil Burgess, Maneesh Sahani, and Samuel J Gershman. Successor-predecessor intrinsic exploration. *Advances in Neural Information Processing Systems*, 36, 2024.

[262] Benjamin Eysenbach and Sergey Levine. Maximum entropy RL (provably) solves some robust RL problems. *arXiv preprint arXiv:2103.06257*, 2021.

[263] Zhang-Wei Hong, Tzu-Yun Shann, Shih-Yang Su, Yi-Hsiang Chang, Tsu-Jui Fu, and Chun-Yi Lee. Diversity-driven exploration strategy for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[264] Yannis Flet-Berliac, Johan Ferret, Olivier Pietquin, Philippe Preux, and Matthieu Geist. Adversarially guided actor-critic. *arXiv preprint arXiv:2102.04376*, 2021.

[265] Edoardo Conti, Vashisht Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth Stanley, and Jeff Clune. Improving exploration in evolution

strategies for deep reinforcement learning via a population of novelty-seeking agents. *Advances in Neural Information Processing Systems*, 31, 2018.

[266] Jack Parker-Holder, Aldo Pacchiano, Krzysztof M Choromanski, and Stephen J Roberts. Effective diversity in population-based reinforcement learning. *Advances in Neural Information Processing Systems*, 33:18050–18062, 2020.

[267] Pengyi Li, Jianye Hao, Hongyao Tang, Xian Fu, Yan Zhen, and Ke Tang. Bridging evolutionary algorithms and reinforcement learning: A comprehensive survey on hybrid algorithms. *IEEE Transactions on Evolutionary Computation*, 2024.

[268] Jean-Baptiste Gaya, Laure Soulier, and Ludovic Denoyer. Learning a subspace of policies for online adaptation in reinforcement learning. *arXiv preprint arXiv:2110.05169*, 2021.

[269] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[270] J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 209:415–446, 1909.

[271] Andreas Christmann and Ingo Steinwart. *Support vector machines*. Springer New York, NY, 2008.

[272] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge University Press, 2018.

[273] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, et al. BoTorch: A framework for efficient monte-carlo bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[274] Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. GPyTorch: Blackbox matrix-matrix gaussian process inference with GPU acceleration. In *Advances in Neural Information Processing Systems*, 2018.