Non-exponential reward discounting in deep reinforcement learning

by

Raja Farrukh Ali

B.E., National University of Sciences and Technology, Pakistan, 2009

M.S., National University of Sciences and Technology, Pakistan, 2014

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2024

Abstract

The science of sequential decision making, formalized through reinforcement learning (RL), has driven various recent technological breakthroughs, from mastering complex games that require strategic thinking to driving advancements in natural language processing. Central to an RL agent's learning is how it treats rewards (the learning signal) and adjusts its policy to maximize cumulative rewards. Future rewards are weighed less than immediate rewards, and traditional RL methods employ exponential discounting to balance immediate and future rewards. However, studies from neuroscience and psychology have shown that exponential discounting does not accurately reflect human and animal behavior, who instead exhibit hyperbolic discounting of future rewards.

This dissertation explores non-exponential discounting, such as hyperbolic, in different facets of deep RL such that it can mirror the intricate decision-making processes found in humans, and evaluate its impact on agent performance in a variety of settings. First, I revisit the idea of hyperbolic discounting and the auxiliary task of learning over multiple horizons in RL agents while using off-policy value-based methods, studying its impact on sample efficiency and generalization to new tasks while incorporating architectural and implementation improvements. Second, I introduce a two-parameter discounting model based on generalized hyperbolic discounting in the deep RL setting. With its sensitivity-to-delay parameter, this model enriches temporal decision-making in RL, as evaluated through empirical evidence. Third, I apply hyperbolic discounting to multi-agent systems, examining its influence on collective decision-making and performance, revealing the potential for improved cooperation among agents. These contributions highlight the impact of non-exponential discounting on agent performance, linking theory with AI practice, facilitating human-like decision-making, and paving the way for new research directions.

Non-exponential reward discounting in deep reinforcement learning

by

Raja Farrukh Ali

B.Eng., National University of Sciences and Technology, Pakistan, 2009

M.S., National University of Sciences and Technology, Pakistan, 2014

A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2024

Approved by:

Major Professor
Dr. William H. Hsu

Copyright

Abstract

The science of sequential decision making, formalized through reinforcement learning (RL), has driven various recent technological breakthroughs, from mastering complex games that require strategic thinking to driving advancements in natural language processing. Central to an RL agent's learning is how it treats rewards (the learning signal) and adjusts its policy to maximize cumulative rewards. Future rewards are weighed less than immediate rewards, and traditional RL methods employ exponential discounting to balance immediate and future rewards. However, studies from neuroscience and psychology have shown that exponential discounting does not accurately reflect human and animal behavior, who instead exhibit hyperbolic discounting of future rewards.

This dissertation explores non-exponential discounting, such as hyperbolic, in different facets of deep RL such that it can mirror the intricate decision-making processes found in humans, and evaluate its impact on agent performance in a variety of settings. First, I revisit the idea of hyperbolic discounting and the auxiliary task of learning over multiple horizons in RL agents while using off-policy value-based methods, studying its impact on sample efficiency and generalization to new tasks while incorporating architectural and implementation improvements. Second, I introduce a two-parameter discounting model based on generalized hyperbolic discounting in the deep RL setting. With its sensitivity-to-delay parameter, this model enriches temporal decision-making in RL, as evaluated through empirical evidence. Third, I apply hyperbolic discounting to multi-agent systems, examining its influence on collective decision-making and performance, revealing the potential for improved cooperation among agents. These contributions highlight the impact of non-exponential discounting on agent performance, linking theory with AI practice, facilitating human-like decision-making, and paving the way for new research directions.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I bow down to the Almighty for giving me the opportunity to undertake this PhD, and for letting me complete it. I pray that He enables me to fully use this knowledge, keep adding to it, and guide me in transferring it to a newer generation.

Anyone who has done a PhD will attest to the fact that it is a long and grueling journey. It is best not traversed alone, but it is a singular and lonely journey in the end. When I look back at all the years spent in this academic exercise, I have a number of people to thank for helping me in different ways and shaping me into the person I am today.

I am grateful to my advisor, William Hsu, for his kindness, generosity, and feedback during the course of this PhD. He granted me the liberty to explore concepts that genuinely intrigued me and remained patient with me throughout. He was always willing to share his years of experience with me and mentored me to avail myself of opportunities that were unknown to me. His constant motivation, mentorship, and availability to lend a compassionate ear at all hours made this journey much easier. I can confidently say that knowing him and working closely with him has made me a better human being. Few PhD advisees can claim to have had such an impact. I would like to thank my committee members for their valuable feedback. I am also indebted to Scott DeLoach for his excellent leadership and support as the CS Department Head.

I am grateful to my family for consistently offering me unwavering love, support, and motivation. To my mother, who has been the bedrock of my support system and whose prayers I have always relied upon. To my father, who always felt proud whenever I achieved a milestone, and showed more enthusiasm and jubilation than I have ever felt myself. To my brother for always trying his best to guide me, and for bearing with me for the longest time growing up.

I would also like to thank my collaborators and peers from the KDD research lab. I would like to thank Nasik Nafi, with whom I worked on two research projects, and whose

ideas and wisdom I greatly appreciate. I would also like to thank fellow KDD lab members, especially Vahid Behzadan and Huichen Yang, who provided guidance and comfort that I was not alone in this journey. Thanks also to Talha Zaidi for his ever-present encouragement and for looking towards me for inspiration. I have also had the privilege to mentor and work with many excellent undergraduate research students in the KDD lab, and I would like to thank Kevin Duong, John Woods, Ethan Coleman, Emmanuel Adeniji, Ziyu Zhao, Vinny Sun, Trey Etzel and Mike Hulcy for their collaboration and contribution to this research.

I am grateful to some external mentors and colleagues, whose discussions allowed me to learn a lot. Thanks to Jakob Foerster, my mentor at AAAI Doctoral Consortium, as well as peers from my DC cohort — Esmaeil Seraj and Kaleigh Clary — for their support and insightful discussions. I am grateful to Robert Southern and Travis Smith from the Psychological Sciences department at K-State for sharing their perspectives and being enthusiastic about my ideas.

This PhD would not have been possible without the mentorship of Asad Pirzada and Adnan Kiani, who guided me during my undergraduate and master's studies, respectively, serving as the springboard for my doctoral journey. I am also indebted to some former colleagues who encouraged and helped me to pursue this path - Soban Nazir, Salman Khalid, Zohaib Iqbal, Faisal Bangash, Nauman Masood, and Malik Umer. Special thanks to my friends Umer Jawaid and Khawar Mohiuddin for always providing me with a listening ear whenever I needed to vent.

Finally, I would like to thank my wife, whose love and support allowed me to undertake and complete this journey. She was the sounding board with whom I would discuss the intricate topics of life, and seek advice. I was fortunate to work with her on one research project, and I look forward to working on many more in the coming years. To my son, I appreciate your patience in sacrificing some of the precious time we could have spent together. Your smile and laughter have always lightened my mood and made this journey worthwhile. Thank you.

In loving memory of my mom,

whose support was always there at every step of my life,

and whom I lost during this journey.

Chapter 1

Introduction

Human decision-making is complex, influenced by factors like immediate circumstances, future prospects, personal preferences, and the inherent value of outcomes. Understanding and modeling human decision-making processes have long been subjects of interest across various disciplines, including psychology, neuroscience, economics, and, more recently, artificial intelligence (AI).

One of the fundamental aspects of human decision-making is how individuals value rewards over time, a concept known as reward discounting. Traditional economic models use exponential discounting, where the value of a reward decreases at a constant rate over time. However, empirical studies have consistently shown that hyperbolic discounting better describes human preferences, where the value of future rewards decreases more steeply for earlier delays than for later delays. In other words, hyperbolic discounting suggests a preference for smaller, immediate rewards over larger, delayed ones, and offers a realistic framework for capturing human preferences. This discrepancy may have significant implications for designing AI and reinforcement learning (RL) agents that can mimic human decision-making or optimize long-term outcomes in environments where human interaction is critical.

Understanding how individuals make choices, particularly when faced with intertemporal trade-offs, has significant implications for developing effective decision support systems, designing incentive structures, and creating AI agents that can interact with humans more

naturally and intuitively. One of the key challenges in modeling human preferences lies in the observation that individuals often exhibit time-inconsistent behavior, deviating from the standard exponential discounting model assumed in classical economic theory. This phenomenon, known as hyperbolic discounting, suggests that people tend to place disproportionately higher value on immediate rewards than delayed ones, leading to impulsive and seemingly irrational choices. However, humans can also plan for multiple timescales (or time horizons) simultaneously and can exhibit preference reversals. Both of these characteristics are lacking when exponential discounting is used.

1.1 Problem Statement

The reinforcement learning (RL) paradigm has shown promise as a path towards important aspects of rational utility in autonomous agents. Reward, specifically reward maximization, has been hypothesized as being *enough* to learn intelligent behavior (Silver et al., 2021). For a learning agent to acquire multiple abilities simultaneously (e.g., planning, motor control, language, etc.), the singular goal of reward maximization may be enough to generate complex behavior, rather than learning and reasoning over specialized problem formulations for each ability. How we treat this reward signal is thus central to our quest for intelligent agents.

In the RL problem setting, the objective is to maximize cumulative rewards over time, known as return. Different approaches to calculating this return include aggregating undiscounted rewards over a finite number of steps, calculating a discounted sum (infinite rewards sum to a finite number), or calculating the average reward per time step. The discounted reward formulation remains the most common in contemporary research, in which a discount factor $0 \leq \gamma < 1$ exponentially reduces or *discounts* the present value of future rewards, r_t at step t, as $\gamma^t r_t$. Reward discounting prioritizes sooner rewards over later rewards and enables a convergence proof for the infinite horizon case.

The functional form of the discounting function directly influences the solutions learned. Evidence from psychology and economics shows that human and animal preferences for future rewards can be modeled more accurately using hyperbolic discounting ($\Gamma_k(t) = \frac{1}{1+kt}$

for $k > 0$). Preference reversals can also occur with time, which can be modeled by hyperbolic discounting but not exponential discounting. Fedus et al. (2019) show that a deep RL agent that acts via hyperbolic discounting is indeed feasible while approximating hyperbolic and other non-exponential discounting functions using familiar temporal difference (TD) learning methods like Q-learning. This approximation is made possible by learning many Q-values simultaneously, each for a different discount factor, and in doing so, the agent also learns over multiple horizons, which is shown to be an effective auxiliary task. Exponential discounting is consistent with a prior belief that a known constant risk exists to the agent Sozou (1998). However, hyperbolic and non-exponential discounting is more appropriate when an agent holds uncertainty over the environment's hazard rate (defined as the per-time-step risk the agent incurs as it acts in the environment). Hyperbolic discounting is theorized to be most beneficial when the hazard rate characterizing the environment is unknown.

This dissertation focuses on the use of non-exponential discounting functions and explores them under environment conditions that are inherently hazardous (dynamic hazard), where the severity of the hazard is unknown, and where the agent is unsure about its survival. I investigate this approach in single and multi-agent systems across a wide variety of agent performance dimensions, such as cumulative return (performance), sample efficiency, and generalization tasks.

Integrating hyperbolic discounting into reinforcement learning frameworks poses several challenges. First, the non-stationarity introduced by the time-inconsistent nature of hyperbolic discounting can lead to instability and convergence issues in traditional RL algorithms. Second, the additional parameter representing sensitivity to delay in hyperbolic discounting models adds complexity to the optimization problem and may require novel algorithmic approaches. Despite these challenges, incorporating hyperbolic discounting into RL agents promises to enable more human-like decision-making, improved generalization to real-world scenarios, and better alignment with human preferences. By capturing the tendency to overvalue immediate rewards, hyperbolic discounting could lead to more realistic and interpretable behavior in RL agents, potentially enhancing their ability to interact with humans and make decisions in domains where intertemporal trade-offs are prevalent.

1.2 Contributions

This dissertation aims to examine whether incorporating hyperbolic discounting within RL agents can enhance their decision-making processes, aligning them more closely with human-like decision-making behaviors. Ultimately, this approach seeks to bridge the gap between traditional AI models and the nuanced, often irrational, patterns of human behavior, paving the way for more intelligent, adaptable, and human-centric AI systems. The specific contributions of this dissertation are outlined as follows:

1. **Revisiting Hyperbolic Discounting:** I conduct a comprehensive analysis to investigate the influence of hyperbolic discounting and multi-horizon learning on agent performance. This includes examining aspects such as sample complexity and generalization using contemporary benchmarks, neural network architectures, and simulation environments.

2. **Generalized Hyperbolic Discounting:** I introduce a novel discounting model that merges Rachlin's hyperbolic framework with deep reinforcement learning. The Rachlin model, also known as Generalized Hyperbolic discounting, incorporates a second term called sensitivity-to-delay, which measures how agents perceive the same delay differently in accumulating rewards.

3. **Non-Exponential Discounting in Multi-Agent RL:** I explore the application of hyperbolic discounting in multi-agent reinforcement learning (MARL). This involves assessing its impact on individual agent decisions and collaborative performance. I introduce and analyze six MARL methods across various classes to investigate the effects of hyperbolic discounting on agent behavior and overall performance.

Some of the contributions of this work have already been published in peer-reviewed conferences and workshops:

- Raja Farrukh Ali. Non-exponential reward discounting in reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 16111–16112, 2023 (Ali, 2023).

- Raja Farrukh Ali, Kevin Duong, Nasik Muhammad Nafi, and William H. Hsu. Multi-horizon learning in procedurally-generated environments for off-policy reinforcement learning (student abstract). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 16150–16151, 2023 (Ali et al., 2023a).

- Raja Farrukh Ali, Nasik Muhammad Nafi, Kevin Duong, and William Hsu. Efficient multi-horizon learning for off-policy reinforcement learning. In *NeurIPS Deep Reinforcement Learning Workshop*, 2022 (Ali et al., 2022).

In addition to the above, proposed works mentioned in Chapters 4 and 5 are in the process of being submitted to peer-reviewed conferences. The following publications are additional contributions that I have made during my PhD studies but are not part of this dissertation.

- Nasik Muhammad Nafi, Raja Farrukh Ali, and William Hsu. Hyperbolically discounted advantage estimation for generalization in reinforcement learning. In *ICML Workshop on Decision Awareness in Reinforcement Learning*, 2022 (Nafi et al., 2022b). Full paper accepted to AAMAS 2024.

- Nasik Muhammad Nafi, Raja Farrukh Ali, and William Hsu. Analyzing the Sensitivity to Policy-Value Decoupling in Deep Reinforcement Learning Generalization. *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2023 (Nafi et al., 2023). Previous version appeared in *NeurIPS Deep Reinforcement Learning Workshop*, 2022 (Nafi et al., 2022a). Full paper accepted to IJCNN 2024.

- Raja Farrukh Ali, Ayesha Farooq, Emmanuel Adeniji, John Woods, Vinny Sun, and William Hsu. Explainable reinforcement learning for alzheimer's disease progression prediction. In *NeurIPS XAI in Action Workshop*, 2023 (Ali et al., 2023b).

1.3 Dissertation Outline

This dissertation is organized as follows:

- **Chapter 2** provides a comprehensive literature review covering the theoretical foundations of reinforcement learning, reward discounting, contemporary deep RL methods, and simulation environments used to study these problems.

- **Chapter 3** presents a detailed analysis of the performance of RL agents that use hyperbolic discounting and learn over multiple horizons, revisiting the idea in new problem settings as well as focusing on new dimensions of learning such as sample complexity and generalization.

- **Chapter 4** introduces the generalized hyperbolic discounting framework in deep RL, which uses the two-parameter Rachlin hyperbolic discounting model. This chapter discusses the proposed model's theoretical foundations, implementation details, and empirical evaluation.

- **Chapter 5** explores the integration of hyperbolic discounting into multi-agent reinforcement learning (MARL) methods, investigating its impact on agent decision-making and performance in cooperative and competitive scenarios.

- **Chapter 6** concludes the dissertation by summarizing the key findings, discussing limitations, and outlining potential future research directions.

All chapters that contain published research results have supplementary material and code available. Source code will typically be available on GitHub[1] or linked from the author's webpage[2] against the respective published article.

[1] https://github.com/rfali
[2] https://rfali.github.io

Chapter 2

Background

In this chapter, we lay down the essential groundwork and theoretical frameworks that serve as prerequisites for the subsequent sections of this dissertation. Specifically, We discuss the following topics: introduction to reinforcement learning (RL) (Section 2.1), its formalism (Section 2.2), fundamental concepts (Section 2.3) and classifications (Section 2.4), reward discounting (Section 2.5), algorithms for single-agent (Section 2.7) and multi-agent RL (Section 2.8), and finally the RL simulation environments used in this dissertation (Section 2.9). This chapter is loosely based on prior surveys (Jaeger and Geiger, 2023; Jaques, 2019; Kwiatkowski et al., 2022), and concepts pertinent only to specific chapters are introduced as additional background within those respective chapters.

2.1 Introduction to Reinforcement Learning

Modern approaches to machine learning are typically categorized into three main types: supervised learning, unsupervised learning, and reinforcement learning. Supervised Learning involves training models on labeled data, where each data point is associated with a known output or label. Abstractly, supervised learning involves finding a function $f : \mathcal{X} \to \mathcal{Y}$ that takes as input $x \in \mathcal{X}$ and gives as output $y \in \mathcal{Y}$ such that $y = f(x)$. On the other hand, Unsupervised Learning, which encompasses Self-Supervised Learning, operates on raw

data without explicit labels, aiming to uncover patterns or structures within the data itself. Reinforcement Learning, inspired by behaviorist psychology and distinct from the previous two paradigms, involves an agent interacting with its environment and making decisions to maximize the total rewards obtained over time. In RL, there is no explicit use of labeled data; instead, the agent learns through trial and error, exploring various actions and observing the resulting rewards. The learning process in RL involves the agent iteratively updating its strategy or policy to achieve higher rewards over time.

To differentiate these concepts further, the aim of supervised learning is to fine-tune a model for accurately mapping inputs x to outputs y. In reinforcement learning, these inputs and outputs are referred to as states s and actions a, respectively, with the model being a *policy* π that links states to actions ($\pi(s) \rightarrow a$). Supervised learning focuses on *minimizing* a loss function $L(h(s), a) \rightarrow r$, where $h(s) \rightarrow a^\star$ assigns states to their ideal actions ($a^\star$=ground truth), and $r \in \mathbb{R}$ is a loss value. This loss function usually denoted as $L(a^\star, a)$, is optimized through gradient-based methods using data from a fixed dataset, with $a^\star$ typically determined by human experts. In contrast, reinforcement learning aims to *maximize* a *reward function* $R(s, a) \rightarrow r$, where r is the reward. Reward functions R broaden the concept of loss functions L as the optimal actions $a^\star = h(s)$ do not need to be known.

2.1.1 Deep Reinforcement Learning

In Deep Reinforcement Learning (DRL), the agent's decision-making process is represented by a deep neural network. At its core, a deep neural network is a series of layers, each performing a nonlinear transformation, allowing the model to capture increasingly abstract representations of the data. Deep learning operates through a function $f : \mathcal{X} \rightarrow \mathcal{Y}$, parameterized by $\theta \in \mathbb{R}^{n_\theta}$, where $n_\theta \in \mathbb{N}$ represents the number of parameters, such that $y = f(x; \theta)$.

Consider a basic example of a feedforward neural network with a single hidden layer. The network takes input x, a column vector of length n_x ($n_x \in \mathbb{N}$), and processes it through the

layers. The computation in the hidden layer is described by $h = A(W_1 \cdot x + b_1)$ where W_1 is a matrix of dimensions $n_h \times n_x$ ($n_h \in \mathbb{N}$), b_1 is a bias vector of length n_h, and A denotes the activation function. This activation function introduces non-linearity, enabling the network to model complex relationships. All these layers undergo training with the aim of minimizing the empirical error, denoted by $I_S[f]$. The prevailing approach for optimizing neural network parameters is gradient descent utilizing the backpropagation algorithm (Rumelhart et al., 1986). In its simplest iteration, the algorithm updates its internal parameters θ to better fit the desired function:

$$\theta \leftarrow \theta - \alpha \nabla_\theta I_S[f], \tag{2.1}$$

where α denotes the learning rate.

In contemporary applications, various neural network layers have emerged, extending beyond the conventional feedforward networks just introduced (also called a multi-layer perceptron (MLP) where there could be one or more hidden layers). Each variant offers distinct advantages tailored to specific applications, such as achieving a favorable balance between bias and overfitting in supervised learning scenarios. Here, we briefly discuss the layer types relevant to the work in this dissertation.

Convolutional layers (LeCun and Bengio, 1995) are well-suited for processing images and sequential data due to their translation invariance property. These layers employ learnable filters with small receptive fields, applying convolution operations to inputs and passing results to subsequent layers. Consequently, the network learns to detect specific features, such as edges, textures, and patterns, in initial layers, progressing to identify object parts and whole objects in subsequent layers (Erhan et al., 2009). Recurrent layers, on the other hand, excel in processing sequential data as they use the output from the previous timestep as input to the current timestep, mimicking a type of memory and enabling them to capture temporal dependencies within the data. Variants like Recurrent Neural Networks (RNNs) and Long Short-Term Memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997) are widely used for processing time series or natural language, due to their ability to capture

temporal dependencies through recurrent connections.

The trend in recent years has been towards increasingly deep networks, with some supervised learning tasks employing over 100 layers (Szegedy et al., 2017). The motivation for employing deep neural networks lies in the Universal Approximation Theorem (Cybenko, 1989), which states that a single hidden layer network can approximate any continuous function with sufficient size. However, due to computational impracticality and optimization difficulties, a single, massive hidden layer is not preferred. Stacking layers enables *deep networks* to learn a hierarchical structure, where subsequent layers capitalize on features extracted by preceding layers. This hierarchical structure facilitates the reuse and composition of features from earlier layers to generate new, higher-level features, thereby enhancing the network's representation and learning capabilities.

Deep reinforcement learning (deep RL) methods arise when deep neural networks are utilized to represent the state or to approximate a value function $\hat{v}(s;\theta)$ or $\hat{q}(s,a;\theta)$ or a policy $\pi(a|s;\theta)$ where the parameters θ correspond to the weights in deep neural networks. Usually, stochastic gradient descent is used to update weight parameters in deep RL. The neural network learns to map environmental states to actions that maximize cumulative rewards. Once trained, the DRL model can adapt to new environments or tasks, making it versatile for various applications where dynamic decision-making is required.

2.1.2 Single vs. Multi-Agent RL

While traditional reinforcement learning (RL) has primarily focused on single-agent settings, where an agent interacts with its environment to maximize its cumulative reward, multi-agent reinforcement learning (MARL) has emerged as a critical area of research due to its applicability in scenarios involving multiple interacting agents. In MARL, each agent must learn an optimal policy while accounting for the presence of other agents, whose actions can influence the environment's dynamics, the rewards received by each agent, and the transition dynamics to subsequent states. This added complexity introduces new challenges, such as the need for coordinated decision-making, handling non-stationarity arising from the evolving

policies of other agents, and the potential for emergent behaviors and equilibria that may be difficult to predict or interpret. Consequently, MARL algorithms must be designed to address issues such as credit assignment, exploration-exploitation trade-offs in multi-agent settings, and the curse of dimensionality that arises from the exponential growth of the joint action space as the number of agents increases. Despite these challenges, MARL holds significant promise for applications in domains such as multi-robot coordination, decentralized control systems, autonomous vehicles, and multi-player games, where the ability to learn and adapt in the presence of multiple interacting agents is crucial.

2.2 Formalism

We now introduce the formal background used to study RL. Fundamentally, an RL problem consists of two parts: an *environment* and an *agent* operating within the environment to achieve some goal(s). The agent observes the environment, which translates to receiving the state or observation from the environment, and executes an action according to its policy. On receiving the agent's action, the environment state changes, and the agent receives a reward signal indicative of the action's efficacy. The agent's objective is to maximize the total reward accumulated during an episode, which commences from the initial state and concludes upon reaching the terminal state. Figure 2.1 visually captures this cyclic interaction.

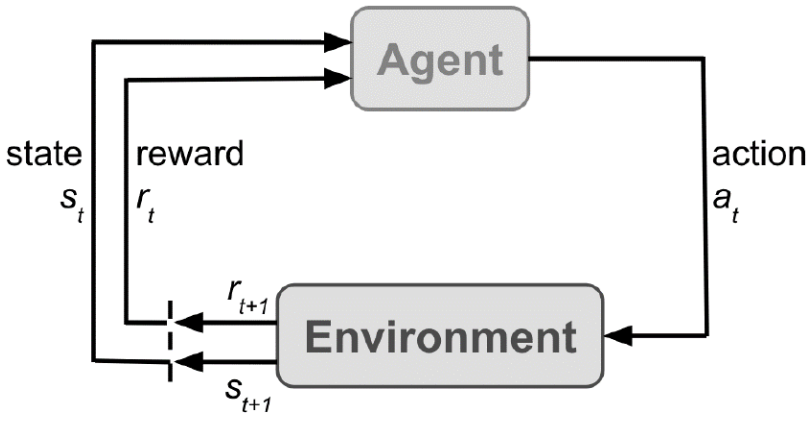


Figure 2.1: The agent interacts with the environment by taking actions given a state. The environment produces the next state and emits a reward (Sutton and Barto, 1998).

2.2.1 MDP

The theoretical foundation of RL is based on the Markov Decision Process (Bellman, 1957b), which is a mathematical framework for modeling sequential decision-making problems. An MDP is defined by the tuple (S, A, T, R, γ), where:

- S is the set of possible states (discrete or continuous).
- A is the set of possible actions (discrete or continuous).
- $T(s'|s, a)$ is the transition probability function, representing the probability of transitioning from state s to state s' after taking action a.
- $R(s, a, s')$ is the reward function, specifying the immediate reward received after transitioning from state s to state s' by taking action a.
- $\gamma \in [0, 1)$ is the discount factor, determining the importance of future rewards.

An initial state $s_0 \in S$ is randomly sampled from the distribution ΔS during each episode. The agent then iteratively selects actions a_t from the action space A, observes a new state $s' \sim T(s, a)$, and receives a reward $r = R(s, a, s')$. This process can repeat indefinitely or until a termination condition is met, defined either by a terminal state in S or a time limit. The agent's objective is to maximize the total discounted reward $\sum_t \gamma^t r_t$, also known as the return $G(t)$. The solution to a Markov Decision Process (MDP) is characterized by an optimal policy $\pi^*(a|s)$, which maps states to actions and, when followed, yields the highest expected discounted cumulative reward.

2.2.2 POMDP

MDPs are characterized by full observability, where agents possess complete knowledge of the current environment state. However, real-world applications often lack this property, leading to the adoption of Partially Observable Markov Decision Processes (POMDPs) (Kaelbling et al., 1998). A POMDP is represented by a tuple $M = (S, A, O, T, R, \Omega, \gamma)$, where

S, A, T, R, γ are defined similarly to MDPs. O denotes a set of possible observations, and $\Omega : S \to \Delta O$ represents the observation function mapping states to observations. Unlike in MDPs, the agent in a POMDP does not directly perceive the true state s_t of the environment but instead observes $o_t \sim O(s_t)$, which may lack complete information due to partial observability.

2.2.3 Markov Games

The two formalisms introduced above, namely MDP and POMDP, address problems involving a single agent, but extending them to accommodate multiple agents varies depending on the required flexibility for a specific application. Markov games (Littman, 1994), sometimes called a stochastic game (Owen, 1982), extend the concept of MDPs to multi-agent settings, where multiple agents interact with each other and the environment. A Markov Game is defined by the tuple $(N, S, \{A_i\}, T, \{R_i\}, \gamma)$, where:

- N is the number of agents
- S is the set of possible states
- A_i is the set of possible actions for agent i
- $T(s'|s, a_1, a_2, \ldots, a_N)$ is the transition probability function, representing the probability of transitioning from state s to state s' when agents take actions $a_1, a_2, \ldots, a_N$
- $R_i(s, a_1, a_2, \ldots, a_N, s')$ is the reward function for agent i, specifying the immediate reward received by agent i after transitioning from state s to state s' when agents take actions $a_1, a_2, \ldots, a_N$
- $\gamma \in [0, 1)$ is the discount factor, determining the importance of future rewards

In a Markov Game, each agent i aims to learn a policy $\pi_i(a_i|s)$ that maximizes its own expected cumulative discounted reward $\sum_t \gamma^t r_i$, given the policies of the other agents.

2.2.4 POSG

Partially Observable Stochastic Games (POSGs) extend Markov Games to scenarios where agents have partial observability of the environment's state (Hansen et al., 2004). In a POSG, each agent receives an observation o_i that is correlated with the true state s, but may not fully reveal it. A POSG is defined by the tuple $(N, S, \{A_i\}, \{O_i\}, T, \{R_i\}, \{\Omega_i\}, \gamma)$, where $S, \{A_i\}, T, \{R_i\}, \gamma$ are defined similarly to Markov Games. O_i is the set of possible observations for agent i and $\Omega_i : S \to \Delta O_i$ represents the function mapping states to observations for an agent i. In a POSG, each agent i aims to learn a policy $\pi_i(a_i|o_i)$ that maximizes its expected cumulative discounted reward, given the policies of the other agents and the partial observability of the environment's state.

A specialized and popular instance of a POSG for fully cooperative tasks is a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) (Bernstein et al., 2002), where all agents collaborate to optimize a shared reward function.

2.3 Fundamentals

We now discuss the fundamental theorems that form the core of most RL algorithms. We explore the Policy Gradient Theorem, which enables direct optimization of a policy. We also discuss Value Functions, which allow for the estimation of expected utilities associated with the states and/or actions available to the agent in a given state. These two principles serve as the foundation for numerous modern RL algorithms, often combining elements from both approaches. In this section, we mostly use notation of Markov Decision Processes (MDPs) as they offer broad generality, but later extend the discussion to scenarios involving partial observability, where observations replace states, and multi-agent settings, where relevant algorithms are adapted to handle multiple interacting agents.

2.3.1 Value Functions

To maximize rewards, the agent learns a policy π that maps observations to actions. To determine the best action at any timestep, an agent must estimate the expected long-term future reward, a central challenge in RL. Typically, the environment is stochastic, making exact predictions of future rewards impossible. Moreover, rewards acquired later are deemed less important than those acquired earlier. Hence, a discount factor $\gamma \in [0, 1)$ is applied to rewards obtained from later time steps. We can then learn to estimate the value, or **value function $\mathbf{V}^{\pi}(\mathbf{s})$**, which represents the total expected future discounted reward an agent can anticipate when starting in state s_t and acting according to its policy π. It is given by:

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}_{\pi(s_t;\theta)}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \ldots | s_t = s] \\ &= \mathbb{E}_{\pi(s_t;\theta)}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s\right] \end{aligned} \tag{2.2}$$

Instead of solely learning a value estimate, we can also aim to learn **action-value function $\mathbf{Q}^{\pi}(\mathbf{a}, \mathbf{s})$**, representing the anticipated future reward of initiating action a from state s.

$$Q^{\pi}(a, s) = \mathbb{E}_{\pi(s_t;\theta)}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, a_t = a\right] \tag{2.3}$$

The Q-value provides insight into which actions are expected to yield the highest payoff from a given state. With access to these Q-values, an agent can devise a policy by selecting the action with the highest Q-value, $Q(a, s)$ i.e., $\pi(s) = \arg\max_a Q(a, s)$. However, determining these Q-values poses a challenge. As the policy improves, the value estimates should change accordingly to reflect the updated policy. To solve this problem, we use the Bellman equation (Bellman, 1957a), which decomposes value estimation into a recursive definition:

$$Q_{\pi}(a_t, s_t) = r_t + \gamma \mathbb{E}_{\pi(s_{t+1}|s_t,a_t)}\left[\max_{a_{t+1}} Q_{\pi}(a_{t+1}, s_{t+1})\right] \tag{2.4}$$

The Bellman equation reveals that value estimate can be described in terms of the re-

ceived reward at time step t (r_t) and the expected future reward starting from the next state, s_{t+1}, estimated using the current best Q-value estimates for s_{t+1}. This enables Q-value estimation through iterative refinement using observed tuples (s_t, a_t, r_t, s_{t+1}) obtained from interactions with the environment, extending recursively until reaching a terminal state. Generally, the Bellman equation provides a recursive relationship between the value of a state (or state-action pair) and the expected value of the successor state (or successor state-action pair).

This method of updating value estimates by bootstrapping from successor states has given rise to a class of learning algorithms known as Temporal Difference (TD) learning. In particular, we can calculate the TD error, expressed as:

$$\delta_t = [r_t + \gamma \max_{a_{t+1}} Q_\pi(a_{t+1}, s_{t+1})] - Q_\pi(a_t, s_t) \tag{2.5}$$

The left side of the equation represents the observed Q-value (from Eq. 2.4), whereas the right side denotes our current best estimate of the Q-value. Therefore, the TD error indicates the deviation of our current Q estimate from the received reward. Given the stochastic nature of the environment, the same action taken in the same state can yield different rewards. Thus, we avoid overwriting our previous Q estimate using only r_t and Eq. 2.4. Instead, we define a loss function by squaring the TD error:

$$\mathcal{L}(s_t, a_t, r_t, s_{t+1}; \theta) = \left(r_t + \gamma \max_{a_{t+1}} Q_\pi(a_{t+1}, s_{t+1}) - Q_\pi(a_t, s_t) \right)^2 \tag{2.6}$$

Through iterative optimization of this loss function using gradient-based learning techniques, we continually refine our Q-value estimates until they converge to their true expected values.

Another kind of value function known as the **advantage function** $\mathbf{A}^\pi(\mathbf{s}, \mathbf{a})$ serves as an intermediary in estimating the value of actions within an environment. It represents the relative benefit of taking a particular action compared to the average action value, encapsulating the difference between the action-value (Q) function and the state-value (V) function.

Specifically, the advantage function quantifies the additional value gained by selecting a specific action in a given state, beyond what is expected from the state's overall value. Mathematically, it is defined as the difference between the action-value and the state-value functions: Advantage is defined as:

$$\begin{aligned} A(s_t, a_t) &= Q(s_t, a_t) - V(s_t) \\ &= r_t + \gamma V(s_{t+1}) - V(s_t) \end{aligned} \tag{2.7}$$

Specifically, the advantage function quantifies the additional value gained by selecting a specific action in a given state, beyond what is expected from the state's overall value. By explicitly capturing the advantage of each action, the advantage function facilitates more informed decision-making in RL tasks, enabling agents to prioritize actions that offer the greatest potential for reward accumulation. Some algorithms employ Advantage to reduce gradient estimation variance, leading to more stable and efficient training.

These value functions can be estimated using various algorithms, such as temporal difference learning and Monte Carlo methods.

Temporal Difference Learning Temporal difference (TD) learning is a popular class of algorithms in reinforcement learning that updates value estimates based on the difference between successive estimates. In TD learning, agents iteratively update their value functions by bootstrapping from successor states, blending together immediate rewards with estimates of future rewards. One of the most well-known TD algorithms is the Q-learning algorithm, which learns action-values directly from experience without requiring a model of the environment. Through iterative updates, Q-learning converges to the optimal action-value function, enabling agents to make informed decisions in complex environments. TD learning algorithms offer advantages such as online learning, computational efficiency, and the ability to handle non-episodic tasks where trajectories are not explicitly defined.

Monte Carlo Methods Monte Carlo methods offer an alternative approach to estimating value functions by directly sampling episodes of experience. Unlike TD methods, Monte

Carlo methods do not require bootstrapping and rely solely on observed returns from complete episodes. By averaging the returns obtained from multiple episodes, Monte Carlo methods provide unbiased estimates of value functions. One of the key advantages of Monte Carlo methods is their ability to handle episodic tasks where trajectories are well-defined, making them particularly suitable for environments with variable episode lengths or termination conditions. However, Monte Carlo methods may suffer from high variance, especially in environments with long episode lengths or sparse rewards. Appropriate variance reduction techniques such as control variates or importance sampling can be applied to offset this limitation.

2.3.2 Policy Gradient

Policy gradient methods are a class of RL algorithms that directly optimize the policy function $\pi_\theta(a|s)$, parameterized by θ, to maximize the expected cumulative discounted reward. The policy gradient theorem (Williams, 1992) provides a way to estimate the gradient of the expected return with respect to the policy parameters. The policy $\pi : S \to \Delta A$ is typically represented as a neural network, with its free parameters, such as the weights θ, optimized through gradient ascent on the total expected reward. The policy network π_θ takes the state as input and produces an output for each potential action. The output layer learns a probability distribution over the next action, $\pi_\theta(a_t|s_t)$, and for action selection, the network directly samples an action $a_t \sim \pi_\theta(a_t|s_t)$.

Mathematically, it can be expressed as follows. If π_θ denotes a parameterized policy with parameters θ, the objective function $J(\theta)$, representing the expected return under policy π_θ, is given by:

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)] \tag{2.8}$$

where τ denotes a trajectory, and $R(\tau)$ represents the return of trajectory τ. The policy gradient $\nabla_\theta J(\theta)$ with respect to the parameters θ is given by:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot R(\tau)\right] \tag{2.9}$$

where s_t and a_t denote the state and action at time step t, respectively, and T is the time horizon of the trajectory. While it may seem complex initially, we can actually derive the above using the Policy Gradient Theorem. A trajectory in the environment, denoted by τ, consists of a sequence of consecutive states and actions taken by the agent, accompanied by corresponding rewards, expressed as $\tau = (s_0, a_0, r_0, s_1, \ldots)$. Given the parameterized policy π_θ, the probability of a trajectory is expressed as

$$P(\tau) = \mu(s_0)\prod_t P(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t) \tag{2.10}$$

$$\log P(\tau) = \log \rho_0(s_0) + \sum_t (\log P(s_{t+1}|s_t, a_t) + \log \pi_\theta(a_t|s_t)) \tag{2.11}$$

The total reward obtained in the trajectory is denoted as $R(\tau) = \sum_t \gamma^t r_t$. Now, considering the expectation across all trajectories τ, with the optimization target defined as:

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)] \tag{2.12}$$

The policy gradient can be derived as follows:

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)] && (2.13)\\
&= \nabla_\theta \int_\tau P(\tau|\theta)R(\tau) \qquad \text{since } \mathbb{E}_x[f(x)] = \int_x f(x)p(x) && (2.14)\\
&= \int_\tau \nabla_\theta P(\tau|\theta)R(\tau) \qquad \text{bring gradient under integral} && (2.15)\\
&= \int_\tau P(\tau|\theta)\nabla_\theta \log P(\tau|\theta)R(\tau) \qquad \text{since } \nabla_\theta P(\tau|\theta) = P(\tau|\theta)\nabla_\theta \log P(\tau|\theta) && (2.16)\\
&= \mathbb{E}[\nabla_\theta \log P(\tau|\theta)R(\tau)] \qquad \text{expectation form} && (2.17)\\
&= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_t \nabla_\theta \log \pi_\theta(a_t|s_t)R(\tau)\right] \text{From 2.11, non-dependent terms go to 0} && (2.18)
\end{aligned}$$

The policy gradient algorithm is conceptually simple, and its essence lies in increasing the likelihood of actions that yield positive rewards while decreasing the likelihood of those associated with negative rewards. The overarching goal is to maximize the total expected future reward garnered during the learning process. It's important to recognize that the above represents the fundamental policy gradient theorem, and there are several potential adjustments, notably through methods like importance sampling (Kahn and Harris, 1951), or incorporating a baseline into the reward $R(\tau)$ such as subtracting the value of a state $V(s)$.

2.3.3 Actor Critic

Actor-Critic methods combine the strengths of value function estimation and policy gradient methods. The actor is responsible for maintaining the policy function $\pi_\theta(a|s)$, while the critic estimates the value function $V^\pi(s)$ or $Q^\pi(s,a)$. The critic's value function estimates are used to compute an advantage function, which guides the actor's policy updates. From Eq. 2.9, we can substitute the advantage function into the policy gradient as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\cdot A(s_t)\right] \tag{2.19}$$

$$= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\cdot Q(s_t,a_t) - V(s_t)\right] \tag{2.20}$$

$$= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\cdot (r_t + \gamma V_{s_{t+1}} - V(s_t))\right] \tag{2.21}$$

2.4 Classifications in RL

Reinforcement learning encompasses a broad spectrum of methodologies, each with its distinct approach to learning and decision-making. These methods can be classified based on various criteria, such as the type of strategies they use (value-based vs. policy-based), the nature of the learning process (on-policy vs. off-policy), the reliance on environmental models (model-free vs. model-based), and the number of agents involved (single-agent vs.

multi-agent). Understanding these classifications provides a structured way to navigate the RL landscape, highlighting the diversity and applicability of RL techniques across different domains.

2.4.1 Value-based vs. Policy-based Methods

RL algorithms can be broadly categorized into value-based methods and policy-based methods. Value-based methods, such as Q-Learning (Watkins and Dayan, 1992) and SARSA (Rummery and Niranjan, 1994; Sutton and Barto, 1998), focus on estimating the optimal value function $Q^*(s, a)$, which represents the expected cumulative discounted reward an agent can obtain by taking action a in state s and following the optimal policy thereafter. Once the optimal value function is learned, the optimal policy can be derived by selecting the action with the highest value in each state. Value-based methods are often sample-efficient and can converge to the optimal policy in the tabular case, but they can struggle with high-dimensional state and action spaces, as well as continuous domains. Policy-based methods, such as policy gradient methods, directly optimize the policy function $\pi_\theta(a|s)$, parameterized by θ, to maximize the expected cumulative discounted reward. These methods can handle high-dimensional and continuous action spaces more effectively than value-based methods, but they can be less sample-efficient and may converge to local optima. Policy-based methods are particularly useful in scenarios where the value function is difficult to estimate or when the goal is to learn a stochastic policy.

2.4.2 On-policy vs. Off-policy

RL algorithms can also be classified based on whether they learn from the same policy that is used to make decisions (on-policy) or from a different policy (off-policy). On-policy methods, such as SARSA (Rummery and Niranjan, 1994) and actor-critic algorithms, learn the value function or policy based on the same policy that is used to make decisions. These methods update their estimates based on the actual experiences generated by the current policy, ensuring that the learned policy is consistent with the behavior policy. On-policy

methods can be more stable and converge to the optimal policy in the tabular case, but they can be less sample-efficient and may struggle with exploration in complex environments. Off-policy methods, such as Q-Learning and Deep Q-Networks (DQN) (Mnih et al., 2015), learn the value function or policy from experiences generated by a different behavior policy. These methods can reuse past experiences, often stored in a replay buffer, to improve sample efficiency and stability. Off-policy methods can be more sample-efficient and facilitate better exploration, but they may introduce additional biases and divergence issues, especially in non-tabular cases.

2.4.3 Model-free vs. Model-based

RL algorithms can be classified based on whether they learn directly from interactions with the environment (model-free) or leverage a learned model of the environment dynamics (model-based). Model-free methods, such as Q-Learning, SARSA, and policy gradient methods, learn directly from interactions with the environment, without explicitly modeling the transition dynamics or reward function. These methods can be more sample-efficient and robust to model inaccuracies, but they may require more interactions with the environment to learn an optimal policy, especially in complex domains. Model-based methods, such as Dyna (Sutton, 1991) and AlphaZero (Silver et al., 2018), learn a model of the environment dynamics, typically represented as a transition function $P(s'|s, a)$ and a reward function $R(s, a, s')$. These methods can leverage the learned model to plan ahead and simulate future trajectories, potentially improving sample efficiency and enabling more efficient exploration. However, model-based methods can be sensitive to model inaccuracies and may struggle in environments with complex or stochastic dynamics.

2.4.4 Single-agent vs. Multi-agent

RL algorithms can be designed for single-agent settings, where an agent learns to interact with an environment, or multi-agent settings, where multiple agents interact with each other and the environment. Single-agent methods, such as Q-Learning, DQN, and policy gradient

methods, focus on learning an optimal policy for a single agent interacting with an environment. These methods have been extensively studied and applied in various domains, including game playing, robotics, and control systems. Multi-agent reinforcement learning (MARL) methods, such as Independent Q-Learning, Counterfactual Multi-Agent Policy Gradients (COMA) (Foerster et al., 2018), and Multi-Agent Deep Deterministic Policy Gradient (MADDPG) (Lowe et al., 2017), address scenarios where multiple agents interact with each other and the environment. MARL algorithms must account for the presence of other agents, whose actions can influence the environment's dynamics and the rewards received by each agent. These methods introduce additional challenges, such as the need for coordinated decision-making, handling non-stationarity arising from the evolving policies of other agents, and the curse of dimensionality due to the exponential growth of the joint action space.

These classifications provide a framework for understanding the strengths, limitations, and applicability of different RL algorithms. In practice, many modern RL algorithms combine elements from multiple categories, leveraging the advantages of different approaches to tackle complex real-world problems.

2.5 Reward Discounting

It is valuable to examine the foundational assumption underpinning all Reinforcement Learning research, often referred to as the Reward Hypothesis. As formulated by Richard Sutton, it posits that "All goals and purposes can be effectively regarded as the maximization of the expected value of the cumulative sum of a received scalar signal (reward)" (Sutton and Barto, 2018). This principle is ingrained in the formalisms and equations of RL, manifesting through the inclusion of a reward function R, where agents aim to maximize the total reward accumulated throughout their lifetime. While some contend that a single reward signal adequately represents the goals of intelligent agents (Silver et al., 2021), others argue that certain objectives cannot be captured solely by a scalar reward (Vamplew et al., 2022). The latter formulation of vector reward can be thought of as a general case, with the scalar reward being a special case. However, there is a consensus that reward is central to learning,

regardless of its representation.

Since reward is central, reward discounting is also a fundamental concept in RL, reflecting the idea that future rewards are valued less than immediate rewards. The discount factor γ determines the extent to which future rewards are discounted, with a higher value of γ placing more emphasis on long-term rewards. We now discuss how reward is treated in terms of its use in learning algorithms.

2.5.1 Exponential Discounting

2.6 Reward Discounting in Reinforcement Learning

Reinforcement learning agents must strike a delicate balance between pursuing immediate rewards and considering long-term benefits. Discounting is a fundamental technique that enables this trade-off, facilitating the evaluation and comparison of action sequences by accounting for both short-term gains and potential future rewards.

2.6.1 Conventional Exponential Discounting

Traditional economic models and early RL algorithms assumed exponential discounting, where the value of a reward decreases at a constant rate over time. This approach introduces a discount factor, γ, ranging from 0 to 1. Values closer to 1 place greater emphasis on distant rewards, while lower values prioritize immediate rewards Sutton and Barto (1998). The exponential discount function is given by:

$$V(r, t) = r \cdot \gamma^t \tag{2.22}$$

where r is the reward, t is the delay, and $\gamma \in [0, 1)$ is the discount factor. The present value of future rewards is computed by multiplying each successive reward by γ raised to the power of the time step:

$$G_t = \sum_{t=0}^{\infty} \gamma^t R_t \tag{2.23}$$

This method has become the standard due to its theoretical convergence guarantees as well as ease of computation. However, it assumes that a single constant discount factor can accurately model the uncertainty or hazard rate across all future states, an assumption that may not hold true in many real-world scenarios.

2.6.2 Hyperbolic Discounting

Empirical studies in psychology and behavioral economics have consistently shown that hyperbolic discounting better describes human preferences, where the value of future rewards decreases more steeply for earlier delays than for later delays. The hyperbolic discount function proposed by Mazur (1987) is given by:

$$V(r, t) = \frac{r}{1 + kt} \tag{2.24}$$

where r is the reward, t is the delay, and k is a free parameter representing the discount rate. Researchers from psychology ((Myerson and Green, 1995; Rachlin, 1989)) have proposed alternative formulations of the hyperbolic discounting model, capturing the tendency for individuals to discount future rewards at a higher rate when the delay is shorter, and at a lower rate when the delay is longer.

The conventional approach assumes that a fixed discount factor can reliably represent the uncertainty or hazard rate associated with an agent's survival across all future states. However, recent research has called this assumption into question, as it may not hold true in many real-world scenarios. Fedus et al. (2019) argue that the actual uncertainty or hazard rate can vary over time or depend on the specific context, rendering a fixed discount rate unrealistic. Additionally, Hu et al. (2022) empirically demonstrated that optimal discount rates can correlate with the quality and size of datasets in offline learning settings, further highlighting the limitations of a static discount factor.

2.6.3 Exponential vs. Hyperbolic Discounting

The traditional approach to discounting in RL and economics is exponential discounting, where the value of future rewards decreases geometrically over time. This method assumes a consistent rate of discount over time, leading to a time-consistent valuation of rewards. However, empirical studies in psychology and behavioral economics (e.g.,(Ainslie, 1975)) have demonstrated that human and animal decision-making does not always follow this model. Instead, hyperbolic discounting, where the discount rate decreases over time, provides a better fit for observed behaviors. This model accounts for preference reversals over time, a phenomenon not explained by exponential discounting.

The application of hyperbolic discounting within the context of Deep RL poses unique challenges and opportunities. Recent works have begun exploring these avenues, aiming to incorporate more psychologically realistic models of temporal decision-making into AI. These efforts include adapting network architectures, loss functions, and training procedures to accommodate non-exponential discounting. Such advancements are crucial for developing AI systems that can better model and predict human-like decision-making behaviors, with significant implications for areas ranging from autonomous systems to personalized AI assistants.

2.7 Single Agent RL Algorithms

We now discuss the single-agent RL algorithms that have been used in this dissertation.

2.7.1 Off-Policy, Value-Based Methods

As described in prior sections, RL algorithms can be classified along multiple axes. In this dissertation, for single agent scenarios, we focus on the Off-Policy, Value-Based Methods, which is one of the two most commonly used methods. This focus is primarily due to the fact that reward discounting is applicable to all methods, and hence, restricting ourselves to one class of methods allowed us to do a much deeper investigative analysis of the relative

merits of different discounting functions. For the single agent setting in this dissertation, we specifically focus on one of the two most widely used value-based methods called Rainbow (Hessel et al., 2018), the other being PPO (Schulman et al., 2017), which belongs to the on-policy policy-based methods. These two methods combined feature in a majority of RL algorithmic research in the past five years (e.g., (Cobbe et al., 2020; Hafner, 2021)), and as such, using one was thought to be sufficient for the purposes of this research. We have also made contributions to a parallel stream of research involving reward discounting in policy-based methods, which can be accessed in Nafi et al. (2022b).

DQN

Q-learning is a widely used model-free RL algorithm that learns the optimal action-value function $Q^*(s, a)$, which represents the maximum expected cumulative discounted reward an agent can obtain from a given state-action pair following the optimal policy. The Q-Learning update rule is given by:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right] \tag{2.25}$$

where α is the learning rate, and γ is the discount factor.

The Deep Q-Network (DQN) algorithm (Mnih et al., 2015), revolutionized the field of RL by demonstrating the ability of deep neural networks to learn successful policies directly from high-dimensional visual inputs. DQN uses a deep convolutional neural network to approximate the action-value function $Q(s, a; \theta)$, where θ represents the network parameters.

Rainbow

Rainbow (Hessel et al., 2018) is an extension of the DQN algorithm that combines several improvements and extensions, including double Q-learning, prioritized experience replay, dueling networks, and distributional reinforcement learning. By integrating these techniques, Rainbow achieves state-of-the-art performance on the Atari 2600 benchmark and demonstrates the potential for combining multiple enhancements to improve the performance of

value-based RL algorithms.

2.8 RL Algorithms (Multi-Agent)

2.8.1 Independent Learning

Independent learning is a simple approach to multi-agent reinforcement learning (MARL) where each agent learns its policy independently, treating the other agents as part of the environment. While this approach is computationally efficient and easy to implement, it often fails to achieve optimal performance in scenarios where coordination among agents is required, as each agent is unaware of the policies and objectives of the other agents.

2.8.2 Centralized Training Decentralized Execution

Centralized Training Decentralized Execution (CTDE) is a learning paradigm that has recently gained significant attention in multi-agent reinforcement learning (MARL). It addresses the challenge of coordinating and optimizing the behaviors of multiple agents in a shared environment. In CTDE, the learning process occurs in a centralized fashion, where a global perspective, including the states and actions of all agents, is utilized to train the agents. This centralized approach allows for the exploitation of additional information during training to enhance learning efficiency and policy effectiveness. However, during execution, each agent operates independently, making decisions based solely on its local perception of the environment. This setup mirrors real-world applications where agents must act based on limited information but can benefit from centralized, coordinated training procedures.

Value Factorization

In value factorization, the joint value function for all agents in a team is learned, and this joint value function is decomposed into individual value functions for each agent. The central critic learns a factored Q-value function that estimates the expected return for each agent based on the joint action and state. This allows for efficient training and scalability to larger

multi-agent systems. During execution, each agent can independently select actions based on its own value function, enabling decentralized decision-making.

Some of the earliest value factorization methods, such as Value Decomposition Networks (VDN) (Sunehag et al., 2017) and QMIX (Rashid et al., 2018, 2020), have shown promising results in cooperative multi-agent tasks. VDN assumes additivity of the individual value functions, while QMIX allows for a more flexible factorization by using a mixing network to combine the individual Q-values. These approaches strike a balance between the expressiveness of the joint value function and the efficiency of decentralized execution

Centralized Policy Gradient

Centralized Policy gradient methods are another class of algorithms used in CTDE for multi-agent reinforcement learning. In contrast to value-based methods, policy gradient methods directly optimize the agents' policies through gradient ascent on the expected return. The central critic estimates the gradient of the joint policy with respect to the parameters, which is then used to update the individual agent policies.

Multi-agent policy gradient algorithms, such as Multi-Agent Deep Deterministic Policy Gradient (MADDPG) (Lowe et al., 2017) and Multi-Agent Proximal Policy Optimization (MAPPO) (Yu et al., 2022), have been successfully applied to various multi-agent tasks. MADDPG extends the single-agent DDPG algorithm (Silver et al., 2014) to the multi-agent setting by using a centralized critic and decentralized actors. MAPPO, on the other hand, is based on the single-agent PPO algorithm and employs a centralized value function for training while allowing decentralized execution.

Policy gradient methods offer several advantages in multi-agent settings. They can handle continuous action spaces and stochastic policies, making them suitable for a wide range of tasks. Additionally, they can be combined with value-based methods to form actor-critic architectures, leveraging the benefits of both approaches.

2.9 Simulation Environments

Simulation environments play a crucial role in RL research. These environments provide controlled and reproducible settings for training and evaluating RL agents, enabling researchers to study various problems and challenges in a systematic and rigorous manner. The use of simulation environments offers several advantages, such as:

1. **Controlled settings**: Simulation environments allow researchers to precisely control the dynamics, complexity, and difficulty of the tasks, enabling them to isolate and study specific aspects of RL algorithms or agent behaviors.

2. **Safety and cost-effectiveness**: Many real-world applications of RL involve high-risk or costly scenarios, such as robotics, autonomous vehicles, or financial trading. Simulation environments provide a safe and cost-effective alternative for training and testing RL agents before deploying them in real-world settings.

3. **Diverse challenges**: Simulation environments can be designed to present a wide range of challenges, from simple gridworld environments to complex 3D environments with high-dimensional observations and continuous action spaces, allowing researchers to evaluate the robustness and generalization capabilities of their algorithms.

4. **Benchmarking and comparison**: Well-established simulation environments, such as the Arcade Learning Environment (ALE) or the StarCraft Multi-Agent Challenge (SMAC), serve as benchmarks for comparing the performance of different RL algorithms and approaches, fostering collaboration and driving progress in the field.

By leveraging simulation environments, researchers can study various RL problems, such as exploration-exploitation trade-offs, credit assignment in multi-agent settings, transfer learning, and the integration of different techniques like hierarchical reinforcement learning or meta-learning. These controlled settings enable researchers to gain insights into the strengths and limitations of their algorithms, leading to the development of more robust and generalizable solutions that can ultimately be applied to real-world applications.

2.9.1 Single Agent

The following single-agent RL environments were studied in this work.

Arcade Learning Environment

The Arcade Learning Environment (ALE) is a popular benchmark for evaluating RL algorithms on Atari 2600 games (Bellemare et al., 2013). ALE provides a standardized interface for interacting with Atari games, allowing researchers to test their algorithms on a diverse set of challenging tasks with high-dimensional visual inputs.

Procgen

Procgen (Cobbe et al., 2020) is a suite of procedurally generated environments designed to test the generalization capabilities of RL agents. These environments feature diverse and challenging tasks, with procedural generation ensuring that each episode presents a unique set of challenges, forcing agents to learn robust and generalizable policies.

Crafter

Crafter is a 3D environment that simulates a crafting task, where an agent must learn to collect resources, craft tools, and build structures (Hafner, 2021). Crafter provides a challenging testbed for RL algorithms, requiring agents to develop long-term planning and hierarchical decision-making skills.

2.9.2 Multi-Agent

The following multi-agent RL environments were studied in this work.

LBF

Level-Based Foraging (LBF) is a multi-agent environment where agents must cooperate to collect resources scattered across a gridworld (Papoudakis et al., 2021). LBF features

different levels of increasing complexity, allowing researchers to evaluate the scalability and generalization capabilities of MARL algorithms.

RWARE

Multi-Robot Warehouse (RWARE) is a multi-agent environment that simulates a warehouse setting, where robots must coordinate to efficiently transport and sort packages (Papoudakis et al., 2021). RWARE provides a challenging testbed for MARL algorithms, requiring agents to develop sophisticated coordination strategies and handle partial observability.

MPE

Multi-Agent Particle Environments (MPE) is a suite of multi-agent environments featuring various tasks, such as cooperative navigation, predator-prey scenarios, and physical deception games (Lowe et al., 2017; Mordatch and Abbeel, 2017). MPE provides a diverse set of challenges for evaluating MARL algorithms, ranging from simple coordination tasks to more complex scenarios involving adversarial interactions.

SMAC

The StarCraft Multi-Agent Challenge (SMAC) is a multi-agent reinforcement learning environment (Samvelyan et al., 2019) based on the popular real-time strategy game StarCraft II. SMAC presents a challenging testbed for MARL algorithms, requiring agents to develop sophisticated strategies, handle partial observability, and coordinate their actions in complex and dynamic scenarios.

Chapter 3

Revisiting Hyperbolic Discounting and Learning Over Multiple Horizons

Learning over multiple horizons has been proposed as an effective auxiliary task for reinforcement learning agents (Fedus et al., 2019). Value estimates at multiple timescales can help create advanced discounting functions and allow agents to form more effective predictive models of their environment. In this work, we revisit the idea of learning over multiple horizons in off-policy RL through the lens of generalization as well as sample efficiency. We empirically analyze this question while incorporating recent architectural, methodological, and implementation improvements in training RL agents. We compare and contrast single-horizon vs multi-horizon learning on Rainbow, a popular value-based, off-policy algorithm, as the agent learns over multiple horizons simultaneously while using either an exponential or hyperbolic discounting function to estimate the value to guide its policy. We report results on ALE, Procgen, and Crafter benchmarks, analyze the effectiveness of different approaches, and present new insights on how such learning affects the agent's generalization performance.

3.1 Introduction

A reinforcement learning agent learns to maximize the rewards it receives over its expected lifetime. But what if its estimate of the expected lifetime is biased? While RL algorithms usually plan for a single, fixed, *distant* horizon by setting the discount factor γ to a value closer to 1, the agent may not live long enough as it had previously expected. Thus, it might have fared better by basing its policy on estimates over multiple time horizons; short, intermediate, and long. For example, humans make plans for the immediate future (work day), short-term (professional goals), and long-term (retirement savings), but a person's policy (based on their valuation of distant rewards) may change drastically because of a terminal diagnosis (curtailment of their expected lifetime). Since the goal of an RL agent is to optimize its return (cumulative reward) through a combination of prediction and control, the prediction method used to estimate the value function (estimated return) plays a central role in the agent's performance.

The discount factor γ determines the timescale of the return. When γ is closer to 0, the agent becomes short-sighted and only maximizes near-term reward, whereas when γ reaches closer to 1, the agent values rewards far into the future. It has been shown that a lower discount value early in learning can help policies to converge, although too low a discount can lead to suboptimal policies (Bertsekas and Tsitsiklis, 1995). Thus, learning a value representation that considers multiple discount factors is important. In deep RL, previous works have attempted to learn a value function at multi-timescale either by concurrently learning value estimate for a fixed set of γs or conditioning the value estimate over γ. All of these methods require multiple fully connected layers at the end of the network to enable γ-specific value prediction and introduce computational overhead compared to the single-valued approach. Off-policy reinforcement learning is generally considered data-efficient in the sense that it can reuse transitions collected from old policies. However, the added computational complexity for multi-horizon learning entails a necessity for an efficient neural network architecture that can enhance the learning process.

In this work, we explore multi-horizon learning from the perspective of enabling per-

formance improvement in off-policy algorithms through deep feature learning and better exploration in diverse settings. We present results on procedurally generated environments, which are designed to test an agent's ability to learn a robust and generalizable policy, while also confirming earlier published results on the Arcade Learning Environment benchmark (Bellemare et al., 2013). We revisit and extend the ideas proposed in Fedus et al. (2019) by using previously unused Rainbow Hessel et al. (2018) extensions (dueling DQN architecture (Wang et al., 2016) and Noisy Nets (Fortunato et al., 2018), coupled with parametric noise injection in weights. We additionally use implementation improvements proposed for Rainbow by Schmidt and Schmied (2021) such as spectral normalization (Miyato et al., 2018) in the residual blocks, mixed precision training, and faster, batched training by using parallel environments. We evaluate both exponential and hyperbolic discounting functions in the multi-horizon setting and empirically analyze these with learning over a fixed, single-horizon setting, across the Procgen (Cobbe et al., 2020) and Crafter (Hafner, 2021) benchmarks.

Our novel contributions can be summarized as follows:

- We evaluate the generalization capability of an agent while learning a policy over multiple timescales on previously untested benchmarks. We employ recent architectural and methodological improvements to shed light on whether multi-horizon learning can boost performance

- We analyze the effect of varying the number of simultaneous horizons, which in turn affects the underlying discounting rates. We further analyze this impact by evaluating different (hyperbolic discounting exponent k and exponential discount factor γ values). We investigate the effect of learning over multiple horizons while taking actions according to the shortest, longest, and intermediate horizons. We do ablations using no-noisy and no-dueling, both of which were not part of prior work, as well as adding results using alternate priorities.

- Our analysis reveals that fewer simultaneous horizons fare better empirically. Better performance in different environments corresponds to different values of hyperbolic exponent k and thus should be considered a critical hyper-parameter to tune.

3.2 Related Work

The need for abstract planning (such as modeling over multiple timescales) is a fundamental problem in AI. Sutton (1995) extend temporal-difference methods to make predictions that are not specific to a single time scale. Studies from psychology and neuroscience show that both humans and animals estimate rewards at numerous timescales (Tanaka et al., 2016) and do not discount future rewards exponentially using a single discount factor γ (Mazur, 1997), instead discounting them hyperbolically (Green et al., 1994). Kurth-Nelson and Redish (2009) proposed the modeling of hyperbolic discounting via distributed exponential discounting, and Fedus et al. (2019) extended this formulation to deep reinforcement learning by approximating hyperbolic discounting from exponential discounting while using multi-horizon (multi-timescale) learning as an auxiliary task. Xu et al. (2018) present a meta-learning approach and propose to use γ as a learn-able parameter of their network. Sherstan et al. (2020) propose Γ-nets to predict value-function for any discount factor γ by using timescale as an input to the value estimator.

3.3 Methodology

We aim to evaluate multi-horizon learning for value-based algorithm by incorporating several existing extensions to the DQN algorithm (Mnih et al., 2015), some of which have not been used in similar prior works Fedus et al. (2019) such as the use of a deeper network for the function approximation and addition of the dueling networks (Wang et al., 2016) to the architecture. The network predicts Q-values for each discount factor γ, and these Q-values are used for the agent's learning (loss calculation). We use the Q-values values from the dueling heads for action selection. We experiment with four methods: 1) single-horizon agent that learns over a fixed horizon with a single γ value and discounts rewards exponentially; multi-horizon agent that simultaneously learns Q-values for n_γ values, but has the choice between acting either using 2) a hyperbolically discounted Q-value, or exponentially discounted Q-value corresponding to either the 3) smallest gamma value (γ_0) or the 4) largest

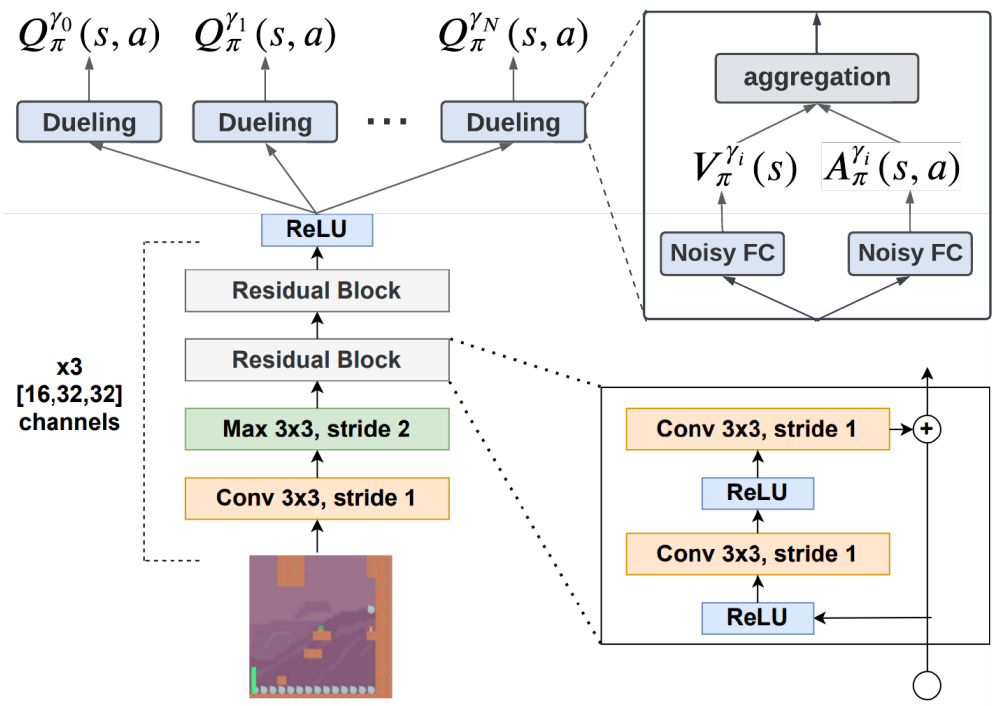


Figure 3.1: Multi-Horizon network architecture. Output layers predict Q-values for different discount factors using an individual output block for each γ. Q-Values from the output head for each γ value are used for action selection.

gamma value (γ_N).

3.3.1 Network Architecture and Rainbow Extensions

The proposed network architecture is presented in Figure 3.1. We employ the deeper IMPALA-CNN architecture (Espeholt et al., 2018) with 15 convolutional layers instead of the small 3-layer network (Nature-CNN) used in Fedus et al. (2019) for multi-horizon learning. The residual blocks (He et al., 2016) in the IMPALA-CNN architecture keep the optimization process light while enabling substantially deeper feature learning. We use the Rainbow (Hessel et al., 2018) agent, which combines several independent improvements on top of the Deep Q-Learning framework. Rainbow uses six extensions to DQN: double DQN (van Hasselt et al., 2016), dueling DQN (Wang et al., 2016), noisy nets (Fortunato et al., 2018), Distributional RL (C51) (Bellemare et al., 2017), prioritized experience replay buffer (Schaul et al., 2016), and n-step returns (Sutton, 1988). We include five of the six; double DQN, du-

eling DQN, noisy nets, prioritized experience replay buffer, and n-step returns; however, we exclude the Distributional RL (C51) component as we trade off implementation complexity with performance benefits, particularly by evaluating a lesser number of time steps (25M). Hessel et al. (2018) show that optimizing the distribution of returns helps in the long run, such as training beyond 40M time steps. Schmidt and Schmied (2021) also note marginal performance improvement with the inclusion of Distributional RL when training for limited (10M) time steps and suggest the exclusion of this component. In contrast to Fedus et al. (2019), our approach includes the dueling network and exploration through noisy nets in the multi-horizon learning setting.

3.3.2 The γ Interval

The number of concurrent horizons (n_γ) is an important factor to consider, along with the values of γ that enforce the minimum and maximum horizon for the agent, beyond which the rewards are negligible. We follow the same scheme as suggested by Fedus et al. (2019) for calculating the γ interval, which are the values of γ on which the integral is approximated using a Riemann sum. These values are produced in a way that taking the exponent of the largest value in this interval with respect to the hyperbolic exponent k approximately equals the γ_{max}, which is the maximum value of horizon we want the agent to use (γ_{max} is set to 0.99). If using a power-method for choosing the γ interval, the base b must satisfy the relation:

$$\gamma_{max} = (1 - b^{n_\gamma})^k \tag{3.1}$$

where the base b can be solved as $b = exp(ln(1 - \gamma_{max}^{1/k})/n_\gamma)$ which bounds γ_{max} to a known stable value instead of it being arbitrarily close to 1. In our experiments, we set $n_\gamma = 5$, yielding the γ interval to be $[\gamma = 0.374,\ 0.608,\ 0.755,\ 0.847,\ 0.904]$. For multi-horizon learning with hyperbolic discounting, if we wish to estimate discounting with the form $\Gamma_k(t) = \frac{1}{1+kt}$ where the hyperbolic exponent $k \leq 1.0$, we need to consider the Q-values for γ^k, as these are the actual γ values being learned via Bellman updates. Hence taking an exponent of the last value from the γ interval (0.904) with respect to k (0.1) yields γ_{max}

(0.99).

3.4 Implementation

Using a fast and data-efficient implementation of Rainbow Schmidt and Schmied (2021), we implement multi-horizon learning for both hyperbolic discounting and exponential discounting (using Q-value corresponding to the largest gamma for action selection). Our multi-horizon learning implementation is inspired by Fedus et al. (2019) but differs in a number of ways; we use hyperbolically discounted Q-value estimates instead of hyperbolically discounted Q-values for the action selection, an IMPALA large (2 channel) network instead of a Nature CNN network architecture, an addition of dueling networks (Wang et al., 2016) to the architecture, and faster training through hardware improvements (mixed precision training and batched training). The Rainbow implementation in Fedus et al. (2019) used three of the six improvements proposed in Hessel et al. (2018), namely Distributional RL, prioritized replay buffer, and n-step returns, whereas our implementation uses five of the six improvements of Rainbow (with the exception of distributional DQN). We build upon the Rainbow implementation of Schmidt and Schmied (2021) to evaluate multi-horizon learning in off-policy reinforcement learning. However, there are some key differences; our evaluation methodology follows pausing training every 250k environment steps to evaluate the agent's current policy for 10 episodes, as opposed to using reloading saved model checkpoints for evaluation. Moreover, we used hyperparameters in line with standard implementations such as Dopamine (Castro et al., 2018) and used environment steps as the standard formulation instead of frames. We use Procgen (Cobbe et al., 2020) to evaluate our approach. The Procgen benchmark is designed to study sample efficiency and generalization in reinforcement learning. The agent is generally trained on a smaller number of levels and expected to perform well in diverse unseen levels. We model the environment as a Partially Observable Markov Decision Process (POMDP), and each level of the environment is a sampled POMDP instance.

3.5 Results

In our experiments, the single-horizon agent uses vanilla Rainbow and learns over a fixed horizon with a single $\gamma = 0.99$ and discounts rewards exponentially. For multi-horizon, the agent simultaneously learns Q-values for N many γ (denoted as N_γ), but has the choice between acting either using a hyperbolically discounted Q-value, $Q^{\Gamma}_{\pi}(s, a)$, or exponentially discounted Q-value, $Q^{\gamma_n}_{\pi}(s, a)$. This exponentially discounted Q-value can correspond either to the smallest gamma value (γ_0), making the agent's actions myopic, or the largest gamma value (γ_N), resulting in a far-sighted agent.

3.5.1 Statistically reliable results

In order to reliably evaluate results and reduce statistical uncertainty in the comparisons, we analyze the performance of our approach as proposed in *rliable* (Agarwal et al., 2021). This was all the more important since our experiments used only 5 seeds. In order to perform a robust evaluation of the results, we compute performance statistics across all environments within the Procgen benchmark using the proposed metrics. This also provides a holistic view of each method's performance across the benchmark, as each individual environment's characteristics and dynamics vary, and hence the resulting performance for each method also varies. We briefly review the rliable metrics as follows.

- **Interquartile Mean (IQM)** takes the second and third quartiles (the middle 50%) of the runs combined across seeds and environments, and calculates the mean score. IQM is not affected by outliers and reduces the statistical uncertainty even with a few runs.

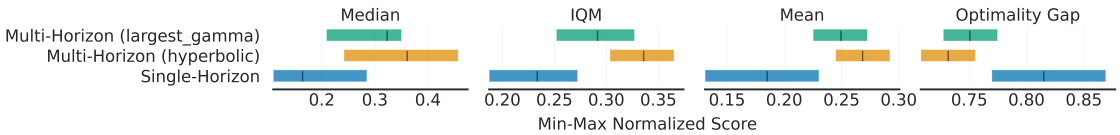
- **Optimality gap (OG)** measures the proportion of performance that fails to meet a minimum threshold score of $\alpha = 1.0$, such that scores beyond α are not considered very important. For both IQM and OG, instead of taking point estimates, interval estimates using stratified bootstrap confidence intervals are computed instead.

- **Performance profile** of an algorithm depicts its score distribution as the fraction of runs above a certain normalized score. These profiles visualize the empirical tail distribution function of a random score, but with stratified bootstrap sampling to produce point-wise confidence bands. The higher the curve of a method, the better it is. The scores are normalized either using PPO mean (taken from results published as part of the *rliable* library (Agarwal, 2021)), or through the achievable min-max scores given for each environment (Appendix C, Normalization Constants of Cobbe et al. (2020))

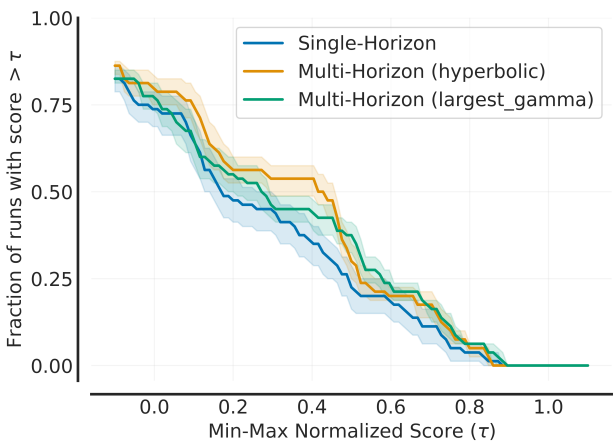
The results shown in Figures 3.2(a) confirm that Multi-Horizon (hyperbolic) variant performs better than Multi-Horizon (largest gamma), alluding to the fact that for procedurally generated environments (and as such any environment where we want an agent to learn a robust policy - unlike memorizing trajectories), learning over multiple horizons can not only be thought of as an auxiliary task. Discounting rewards hyperbolically and *utilizing value estimates from all time horizons to make decisions (acting policy)* are important considerations for learning intelligent behavior. We also provide results for the Crafter (Hafner, 2021) in Figure 3.2(c), which show that multi-horizon learning is able to generalize well using both hyperbolic discounting and exponential discounting (largest gamma).

3.5.2 Performance on a subset of 5 ALE environments

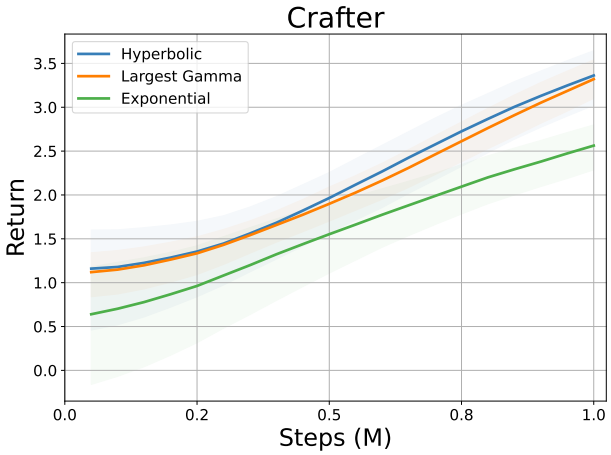
We first recreate and reproduce results on we conduct experiments on a subset of 5 environments from the Arcade Learning Environment (ALE) (Bellemare et al., 2013) as presented by Aitchison et al. (2023). The results, shown in Figure 3.3, confirm that multi-horizon learning performs better than single-horizon learning in most of the environments, which is in line with earlier findings presented by (Fedus et al., 2019). While using an improved and faster variant of Rainbow, we report significantly better results in fewer timesteps.



(a) Procgen - Aggregate Metrics (min-max normalized)



(b) Procgen - Performance Profiles



(c) Crafter - Return

Figure 3.2: Aggregate metrics (IQM, Mean, Median, and Optimality Gap) for all 16 Procgen games (a), Performance profile of the single horizon and multi-horizon Rainbow variants on Procgen (b), and Reward in the Crafter environment for the three methods (c)

3.5.3 Test performance on all Procgen environments

Figure 3.4 shows the test performance of Single-Horizon, Multi-Horizon (largest gamma), and Multi-Horizon (hyperbolic discounting) Rainbow variants on all Procgen environments for 25M timesteps. Results indicate that learning over multiple horizons yields benefits when an agent is trained on a smaller subset of levels and tested on the full distribution of unseen levels in procedurally generated environments. Multi-horizon methods (either the exponential discounting variant, which acts according to the farthest horizon [largest gamma], or the hyperbolically discounted variant) perform better or at par with the single horizon method (with the exception of Heist, whose test scores are much lower than training scores, and Starpilot). When comparing the discounting methods within the multi-horizon setting, we observe no particular method (between exponential and hyperbolic) performing better than the other in all of the environments, although we do observe hyperbolic discounting to be marginally better in a majority of the environments. This finding is similar to Fedus et al.

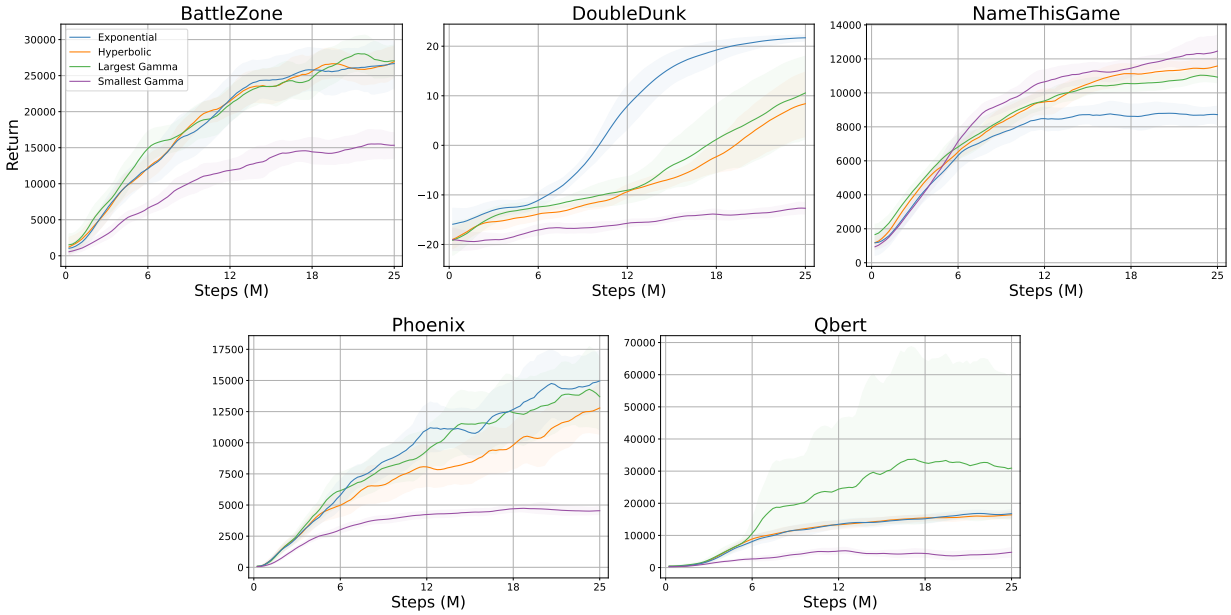


Figure 3.3: Test performance of Single-Horizon, Multi-Horizon (hyperbolic discounting), Multi-Horizon (largest gamma) and Multi-Horizon (smallest gamma) on 5 ALE representative environments from the Atari-5 benchmark (Aitchison et al., 2023). The mean (dark line) and 95% bootstrapped confidence interval (shaded region) are shown, calculated over 5 seeds for each experiment.

(2019), who found multi-horizon learning to be an effective auxiliary task for the agent, irrespective of which discounting function is employed. However, as shown in the next section, hyperbolic discounting does show improved results when compared to exponential discounting with the largest γ across the entire Procgen benchmark.

3.5.4 Varying the number of simultaneous horizons

Historically, RL has primarily been conducted by maximizing the exponentially discounted returns via a single discount factor. For multi-horizon learning, how many horizons should an agent consider? From the theoretical results presented in Fedus et al. (2019), non-exponential discounting functions can be approximated by learning many exponential discounted Q-values, each for a different discount factor γ, which enforces an effective horizon on the agent's learning. In order to perfectly calculate this, an agent must learn over an infinite number of horizons simultaneously, which is not practical. In order to analyze how this

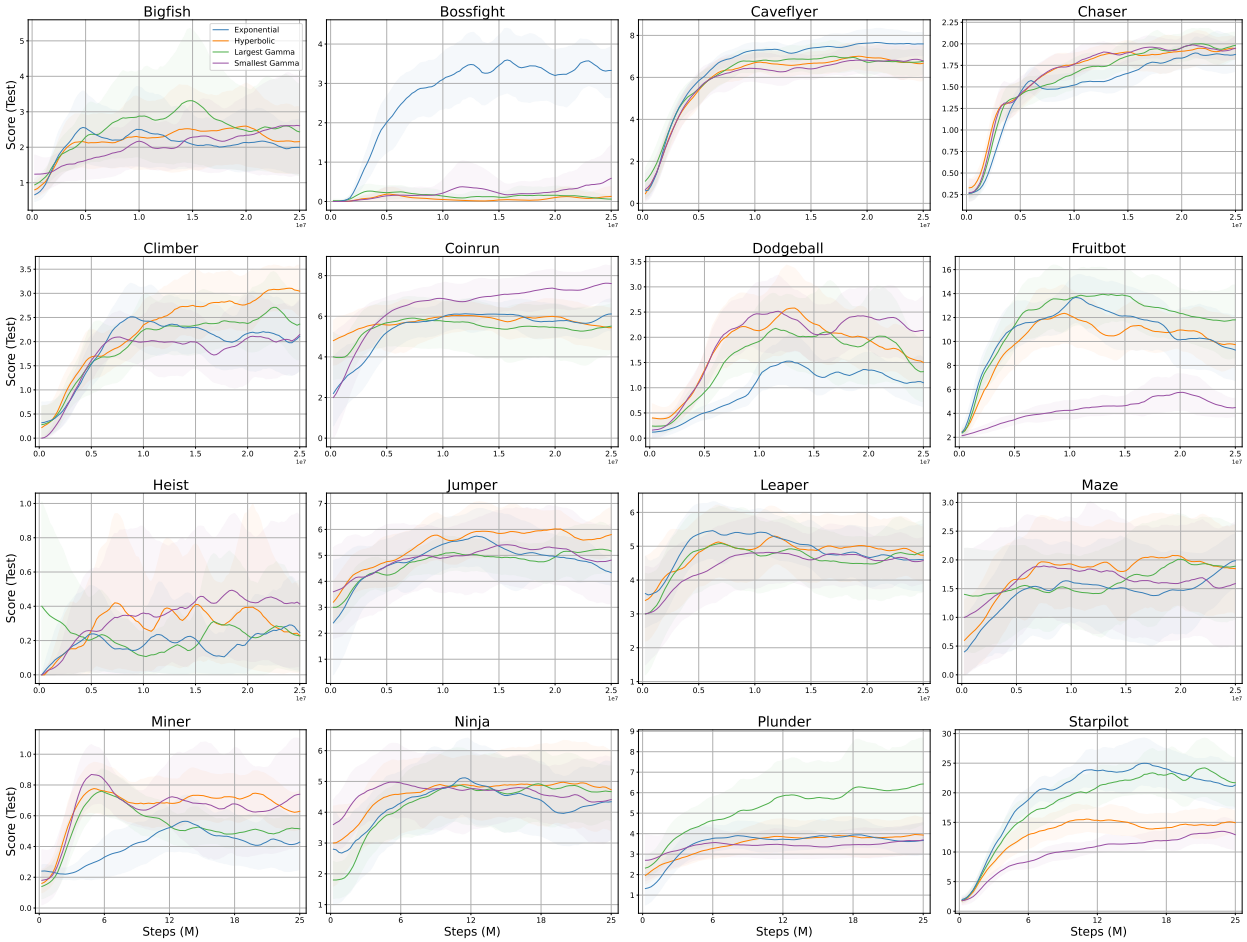


Figure 3.4: Test performance for Single-Horizon, Multi-Horizon (hyperbolic discounting), Multi-Horizon (largest gamma), and Multi-Horizon (smallest gamma) variants of Rainbow across the entire Procgen benchmark. Mean and 95% confidence intervals are shown.

hyperparameter can affect an agent's learning, we present our findings on Procgen in Figure 3.5, showing the effect of varying the number of simultaneous horizons. Having an agent learn over multiple discount factors allows for a more broad scope of learning and decision-making. We explore results on Procgen games using 5, 10, and 20 discount factors (gammas). We find that 5 gammas generally performed the best in these Procgen environments, except notably in the 'miner' environment, which performed best at 10 gammas. With that exception, across these environments, 5 gammas performed either the best or with no disadvantage (either equivalent performance or no statistical difference) compared to the other options. Results show that while learning over multiple horizons is effective (number of horizons $>$ 1), having too many horizons inhibits an agent's learning.

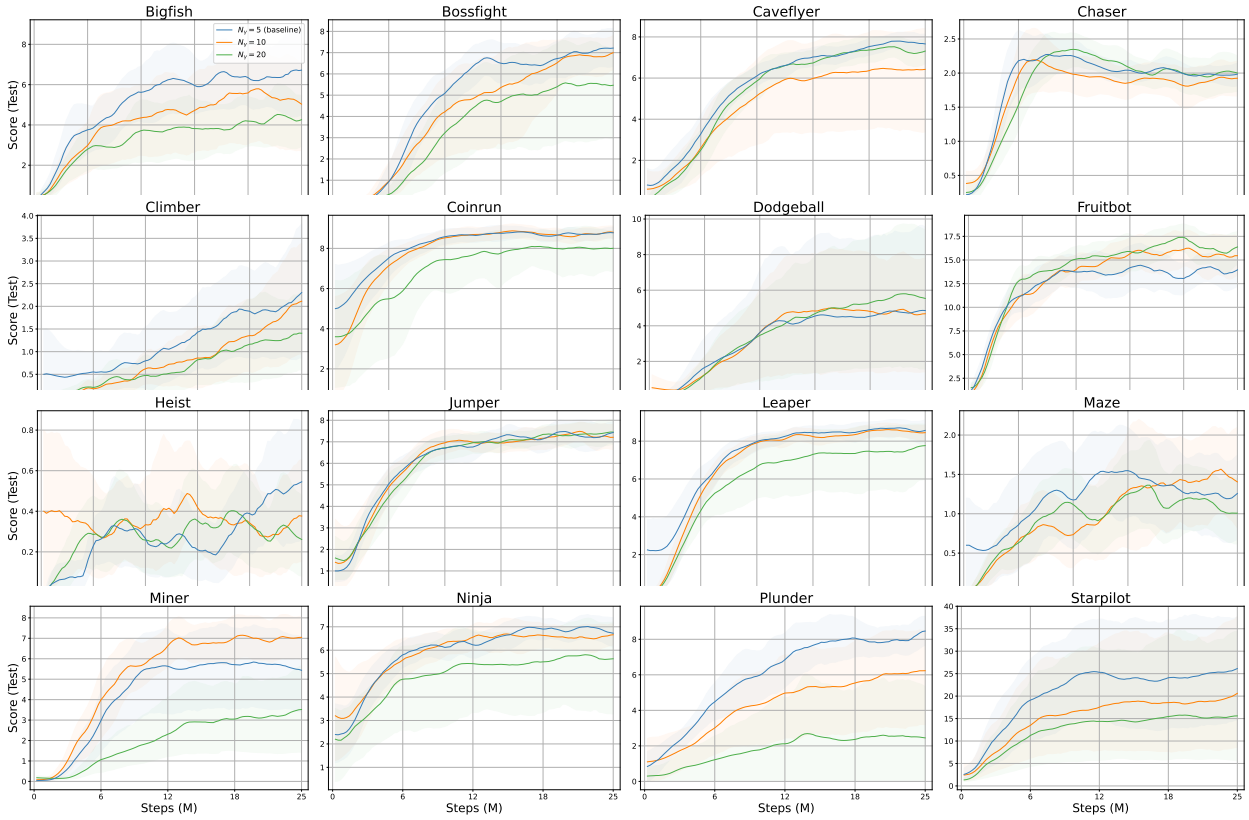


Figure 3.5: Varying the number of simultaneous horizons N_γ for 5, 10 and 20 for the 16 environments in Procgen.

3.5.5 Varying the hyperbolic coefficient

Different environments have different performances across varied hyperbolic exponents. We show this by running each of the Procgen environments with coefficients k = 0.1, 0.01, and 0.001. The plots in Figure 3.6 show that certain k values are suited better for different environments. Certain environments do not seem to have significant performance variations across different k values, such as 'caveflyer' or 'jumper' environments - both of which have little change in return when varying the value of k. On the other hand, some environments show much bigger changes in return when varying k. Two examples of this are the 'miner' and 'plunder' environments - both of which show significantly higher returns with the largest k value (0.1) as opposed to the other lower values (0.01 or 0.001). It is notable that different environments have different hazard rates (and therefore different survival rates) - which can explain the differences in performance using different k-values. If the model is able to more

accurately estimate the hazard (meaning the k-value is optimized for the environment), it would be able to more accurately discount rewards accordingly and would be expected to have higher rewards. On the other hand, less optimized k-values would result in disparities in the estimated hazard versus the actual hazard - meaning less accurate reward estimation and worse expected performance. Figure 3.6 shows the effect of varying the hyperbolic coefficient k. The value of k affects an agent's horizon, as lower values of k tend to elongate the horizon. Results show that lower values of $k = 0.1$ are the most effective for an agent's learning.

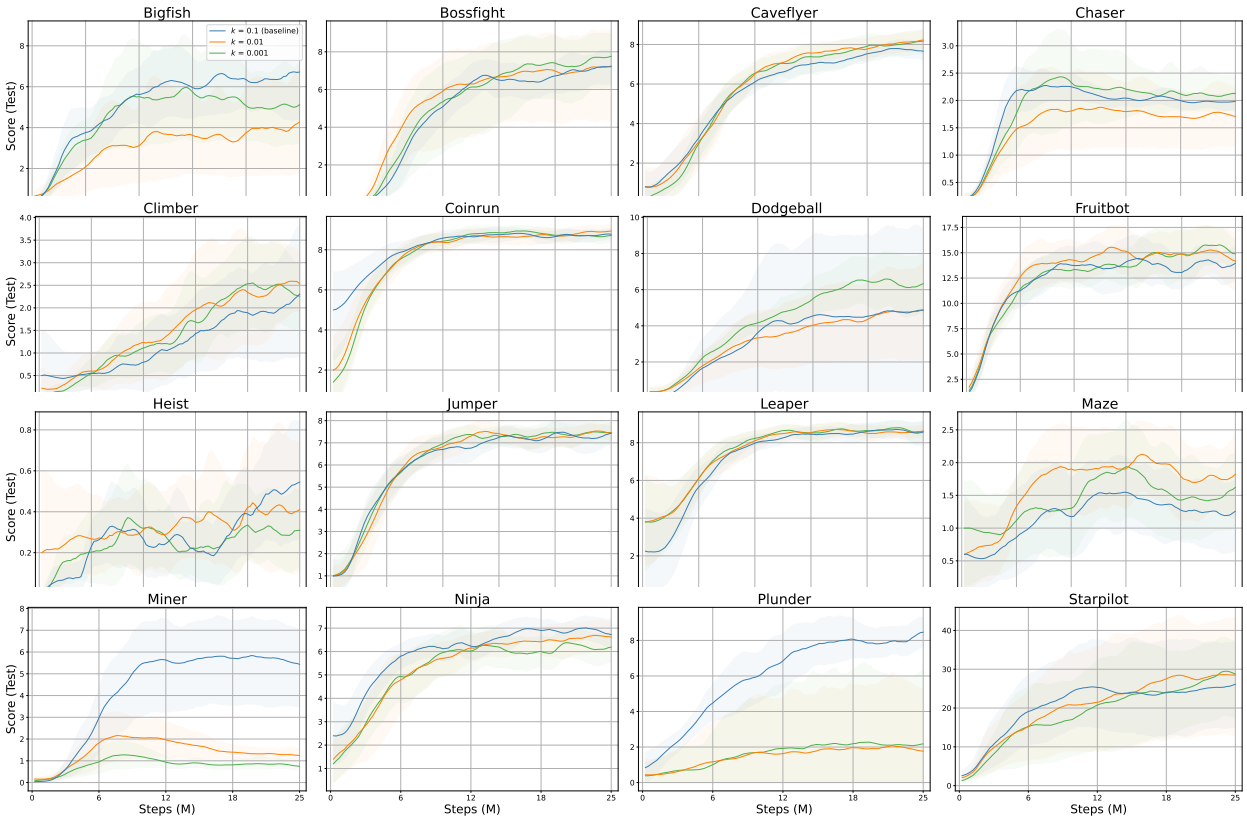


Figure 3.6: Varying the hyperbolic exponent k for three different values (k=0.1, 0.01, 0.001) for the 16 environments in Procgen.

3.6 Conclusion

We report results on the use of multi-horizon learning along multiple dimensions of interest, such as sample efficiency on a wide array of tasks. Our results indicate agents that model value estimates over multiple timescales generally perform better than their single-horizon

counterparts. Moreover, alternative or advanced (non-exponential) discounting functions (e.g., hyperbolic) that are leveraged through the use of multiple timescales perform better than, or at par with, exponentially-discounted, multi-horizon (largest gamma) variants. Agents that learn over multiple horizons but act myopically (smallest gamma) sometimes fare better than single-horizon baselines, which may be explained by their ability to learn from farther horizons. The reason why no particular method performs well across all environments may be attributed to the fact that an agent's prior belief of the risk in the environment influences the specific discounting function they use (Fedus et al., 2019). However, for trials across the full suite of 16 environments over 25M timesteps, the performance profile of multi-horizon (hyperbolic) is higher (and better) than the single-horizon and multi-horizon (largest gamma) variants. Our contribution can be seen as validating the impact of multi-horizon learning on an agent's policy through learning accurate value estimates, achieving higher sample efficiency, and better generalization.

Chapter 4

Generalized Hyperbolic Discounting for Delay-Sensitive Agents

Valuing future rewards impacts an agent's learning. While exponential discounting $d(t) = \gamma^t$ has been widely used because of its time-consistent preferences and ease of use, hyperbolic discounting $d(t) = \frac{1}{1+kt}$ has been shown to better capture human and animal preference behavior. While it has been established that valuing present-term rewards higher than future rewards and hence devaluing later rewards, is an acceptable and rational preference for an agent, how can we differentiate between two agents that have the same discount factor, i.e., quantification of preference for later rewards, but different valuation of the time it takes to receive a reward? Just like there are individual variations in the rate at which humans discount the delay, not everyone exhibits the same level of sensitivity to the delay. Both the exponential and hyperbolic reward discounting functions are single-parameter models used to capture human preference behavior. However, more sophisticated, two-parameter hyperbolic discounting functions have been proposed in the literature and evaluated to provide an even better fit for the studied human behavior. In this chapter, we study one of these two-parameter models and introduce a novel discounting methodology that integrates Rachlin's generalized hyperbolic framework with deep reinforcement learning, incorporating both the discount factor and sensitivity-to-delay parameter. We show empirical results and discuss

the strengths and limitations of this approach.

4.1 Introduction

In RL, rewards are typically discounted exponentially, meaning that a reward obtained t timesteps in the future is discounted by a factor of γ^t (Bellman, 1957b; Sutton and Barto, 1998). This approach establishes a fixed learning horizon for the agent: a smaller γ value prioritizes short-term rewards, while a larger γ value emphasizes long-term rewards. However, human and animal behavior often follows hyperbolic discounting patterns (Mazur, 1987), characterized by a discounting factor of $\frac{1}{1+kt}$, where $k > 0$ represents the hyperbolic discounting rate. Unlike exponential models, hyperbolic discounting allows for preference reversal over time (Smith et al., 2023) and offers better alignment with decision-making scenarios involving multiple reward variables (such as delay, magnitude, and probability) (Green and Myerson, 2004).

Individual differences in the perception of reward delays also play a crucial role. While one agent may exhibit impatience for immediate rewards, another may display patience and willingness to wait. Human studies have shown that the base hyperbolic model tends to overestimate the perceived value of shorter delays and underestimate the value of longer delays (McKerchar et al., 2009). To address such individual variations, a sensitivity to delay parameter, denoted as s or σ where $0 < s < 1$, can be incorporated into the hyperbolic function. Two methods have been proposed for this purpose: one suggested by Myerson and Green (1995), applying s to the entire denominator resulting in $\frac{1}{(1+kt)^s}$, and another proposed by Rachlin (1989), where s is solely applied to the delay term yielding $\frac{1}{1+kt^s}$ (Eq. 4.1).

In this work, We study the Rachlin model as there are three significant practical aspects of the Rachlin model (Franck et al., 2023). Firstly, the Rachlin model offers flexibility and aligns closely with empirical discounting data. Secondly, compared to other available two-parameter discounting models, the Rachlin model provides the advantage of easily obtaining unique, best estimates for parameters across a wide range of potential discounting patterns.

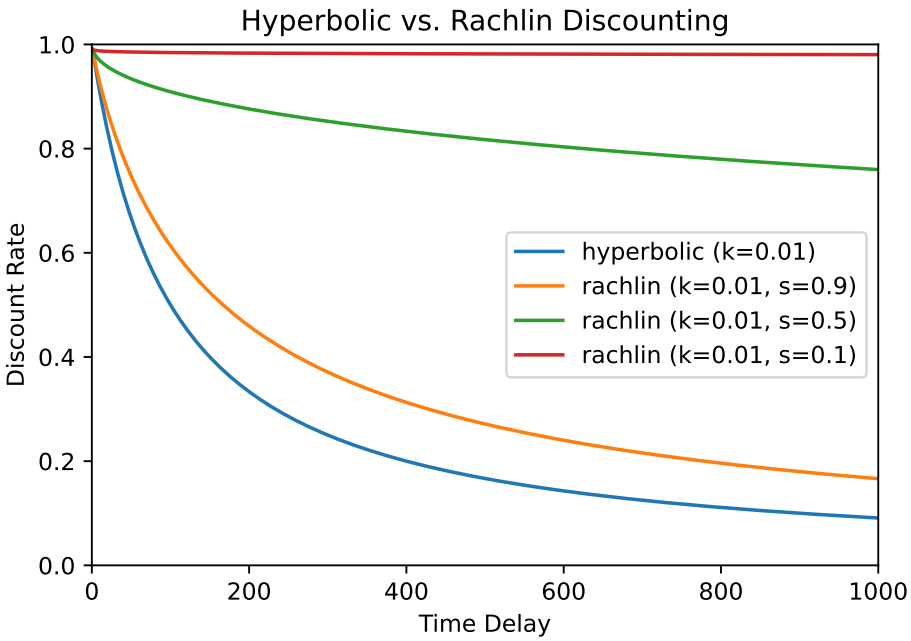


Figure 4.1: Rachlin Hyperbolic vs Base Hyperbolic Discounting. The Rachlin hyperbolic discounting allows variation not only in the rate of discounting but also in sensitivity to delay (denoted here by s). A lower value of s indicates a lesser sensitivity to delay, meaning that the subjective value of a reward decreases less rapidly as the delay increases. Conversely, a value closer to 1 indicates a higher sensitivity to delay, with the subjective value decreasing more rapidly.

Third, the Temporal Difference learning method for deep reinforcement learning introduced by Fedus et al. (2019) can be easily modified to implement the Rachlin model, than the Myserson-Green model. Figure 4.1 depicts the one-parameter hyperbolic discounting and two-parameter Rachlin model of hyperbolic discounting, with three different values of the sensitivity-to-delay parameter s shown.

Mathematically, $V(t)$, which is the discounted value as a function of delay t, for the Rachlin two-parameter hyperboloid models, can be expressed as follows.

$$V(r_t) = \frac{r_t}{1 + kt^s} \tag{4.1}$$

When fixed at $s = 1$, the Rachlin model simplifies to Mazur's one-parameter hyperbolic discounting model (Mazur, 1987). Compared to the one-parameter Mazur model, the two-parameter Rachlin model has the advantage of flexibility to move closer to the observed

data

The purpose of this investigation of integrating two-parameter Rachlin hyperbolic discounting into deep RL is to evaluate whether RL agents are also susceptible to this sensitivity to the delay in the reward and whether there are any benefits and drawbacks of using such a technique in artificial agents. We implement Rachlin discounting in the off-policy value-based RL algorithm Rainbow (Hessel et al., 2018) and evaluate the performance of the Rachlin model for three different values of the sensitivity-to-delay parameter s on multiple RL benchmarks such as Atari-5 (Aitchison et al., 2023), Procgen (Cobbe et al., 2020) and Crafter (Hafner, 2021).

4.2 Related Work

4.2.1 Economics and Behavioral Sciences

The concept of discounted utility has been a fundamental principle in economics and decision theory, reflecting the idea that individuals tend to value immediate rewards more than delayed rewards. Traditional economic models have relied on the exponential discounting model, where the value of a reward decreases at a constant rate over time (Samuelson, 1937). However, empirical studies in psychology and behavioral economics have consistently demonstrated that human preferences are better described by hyperbolic discounting models, which capture the tendency for individuals to discount future rewards more steeply for shorter delays than for longer delays (Ainslie, 1975; Mazur, 1987). Investigations into alternative discounting methods have primarily been conducted within the domains of psychology and behavioral sciences. Human studies have sought to discern individuals' valuation of smaller, immediate rewards compared to larger, delayed rewards, aiming to identify the point at which individuals switch their preferences (McKerchar et al., 2009; Young, 2017).

While attempts were made to fit the data with an exponential model, the hyperbolic model was found to more accurately capture human preferences and preference reversals (McKerchar et al., 2009; Smith et al., 2023; Young, 2017). Sensitivity-to-delay models were

introduced to accommodate individuals' tendency to overvalue immediate rewards and undervalue delayed rewards (Rachlin, 1989) by adding a second parameter to the base hyperbolic discounting model. These 2-parameter models have been observed to provide a better fit than 1-parameter models (McKerchar et al., 2009).

Rachlin (1989) proposed a two-parameter hyperbolic discounting model, which incorporates both a discount factor and a sensitivity-to-delay parameter. This model has been adopted in the study of intertemporal choice and has been shown to provide a better fit to human decision-making data compared to the exponential discounting model (Green et al., 1994; Kirby, 1997). The Rachlin hyperboloid model introduces a delay sensitivity parameter (s) to the hyperbolic equation, adjusting the effect of delay on subjective value and capturing the nonlinear perception of time as a power function. Alternatively, Myerson and Green (1995) incorporate an s parameter that influences both delay and amount sensitivities in the hyperbolic equation, allowing for a broader range of non-linear effects on subjective value, though it offers less specificity in parameter interpretation compared to the Rachlin. In terms of variance accounted for, the two-parameter Rachlin model provided even better fits to median discounting data compared to the one-parameter hyperbolic and exponential models (McKerchar et al., 2009).

4.2.2 Deep Reinforcement Learning

Traditional RL algorithms, such as Q-Learning and policy gradient methods, have primarily relied on the exponential discounting model to handle delayed rewards (Sutton and Barto, 1998). Only a handful of prior works have studied hyperbolic discounting in deep RL (Fedus et al., 2019; Kwiatkowski et al., 2023; Nafi et al., 2022b). However, as RL agents become more sophisticated and are expected to interact with humans in complex environments, there is a growing need to incorporate more realistic models of human preferences, such as hyperbolic discounting. To the best of our knowledge, there is no study that uses two-parameter hyperbolic discounting models, like Rachlin or Myerson-Green (Myerson and Green, 1995), in deep RL.

4.3 Methodology

In order to use Rachlin hyperbolic discounting in deep RL, we need to incorporate this reward discounting function in a learning algorithm. Following Fedus et al. (2019), we demonstrate a method to use exponentially discounted Q-values to compute Rachlin hyperbolic discounted Q-values. The Bellman equation (Bellman, 1957a) is given by

$$Q_{\pi}^{\gamma^t}(s,a) = \mathbb{E}_{\pi,P}[R(s,a) + \gamma Q_{\pi}(s',a')] \tag{4.2}$$

where the expectation $\mathbb{E}_{\pi,P}$ involves sampling an action from a policy $a \sim \pi(\cdot|s)$, next state from transition function $s' \sim P(\cdot|s,a)$, and next action from policy given the next state $a' \sim \pi(\cdot|s')$. Let's consider estimating the value function under hyperbolic discounting. We denote Rachlin hyperbolic Q-values as $Q_{\pi}^{\Gamma_{k\sigma}}$, as shown in Equation 4.4:

$$Q_{\pi}^{\Gamma_{k\sigma}}(s,a) = \mathbb{E}_{\pi}\left[\Gamma_{k\sigma}(1)R(s_1,a_1) + \Gamma_{k\sigma}(2)R(s_2,a_2) + \cdots \middle| s,a\right] \tag{4.3}$$

$$= \mathbb{E}_{\pi}\left[\sum_t \Gamma_{k\sigma}(t)R(s_t,a_t) \middle| s,a\right] \tag{4.4}$$

Remember that the Rachlin hyperbolic discount for a reward r_t at timestep t with hyperbolic exponent k and sensitivity-to-delay σ is give by:

$$V(r_t) = \frac{r_t}{1+kt^s} \tag{4.5}$$

We establish a relationship between the hyperbolic $Q_{\pi}^{\Gamma_{k\sigma}}$-value and the values obtained through conventional Q-learning. The hyperbolic discount $\Gamma_{k\sigma}$ can be expressed as the integral of a specific function $f(\gamma,t)$ for $\gamma = [0,1)$ and $\sigma = [0,1]$:

$$\int_0^1 \gamma^{kt^\sigma} d\gamma = \frac{1}{1+kt^\sigma} = \Gamma_{k\sigma}(t) \tag{4.6}$$

The integral of the function $f(\gamma,t) = \gamma^{kt^\sigma}$ yields the hyperbolic discount factor $\Gamma_k(t)$.

This process involves an infinite set of exponential discount factors over the domain $\gamma \in [0, 1)$, suggesting that γ^{kt} is the standard exponential discount factor and indicating a link to conventional Q-learning. By integrating over this infinite set, we generate hyperbolic discounts for each time step t. Using Equation 4.6, we compute the Q_π^Γ-value associated with the hyperbolic discount factor by aggregating an infinite set of $Q_\pi^{\gamma^k}$-values derived through standard Q-learning, as shown in Equation 4.10.

$$
\begin{aligned}
Q_\pi^{\Gamma\sigma}(s,a) =& \mathbb{E}_\pi \left[\sum_t \Gamma_{k\sigma}(t) R(s_t, a_t) \middle| s, a \right] && (4.7)\\
=& \mathbb{E}_\pi \left[\sum_t \left(\int_{\gamma=0}^{1} \gamma^{kt^\sigma} d\gamma \right) R(s_t, a_t) \middle| s, a \right] && (4.8)\\
=& \int_{\gamma=0}^{1} \mathbb{E}_\pi \left[\sum_t R(s_t, a_t)(\gamma^{kt})^\sigma \middle| s, a \right] d\gamma && (4.9)\\
=& \int_{\gamma=0}^{1} Q_\pi^{(\gamma^{kt})^\sigma}(s,a) d\gamma && (4.10)
\end{aligned}
$$

This approach demonstrates how to compute hyperbolic Q-values by considering an infinite set of exponential Q-values, each corresponding to a different discount factor γ. The number of concurrent horizons (n_γ) is an important factor to consider, along with the values of γ that enforce the minimum and maximum horizon for the agent, beyond which the rewards are negligible. In practice, we start by calculating the γ interval, which are the values of γ on which the integral is approximated using a Riemann sum, and are specified by γ^k. The γ^k value thus obtained is raised to the power of t^σ, and the factor γ^{kt^σ} can be considered as the Bellman gamma, the value of gamma that is used for learning in Q-Learning (Eq. 4.2). Note that Rachlin hyperbolic discounting necessitates the use of **n-step** temporal difference learning, as traditional 1-step learning with $t = 1$, renders the sensitivity to delay parameter σ ineffective, i.e., $t^\sigma = 1$.

4.4 Results

We evaluate our proposed approach on a suite of tasks designed to test the agent's ability to make decisions involving intertemporal trade-offs. These tasks include delayed gratification scenarios, where the agent must choose between an immediate smaller reward or a larger delayed reward, and more complex environments that require long-term planning and decision-making. We conducted experiments to assess the effectiveness of Rachlin hyperbolic discounting across three benchmark environments: ALE (Bellemare et al., 2013), Procgen (Cobbe et al., 2020), Crafter (Hafner, 2021). For all experiments, the hyperbolic discount factor $k = 0.1$ remained constant. We varied the n-step values between 3 and 20 and compared the performance of Rachlin discounting across three arbitrary selections of $\sigma \in \{0.1, 0.5, 0.9\}$. Additionally, We included exponential discounting and hyperbolic discounting as baseline comparisons. We analyze the agent's behavior and overall performance in terms of cumulative discounted rewards.

4.4.1 Atari-5

We first conduct experiments on a subset of 5 environments from the Arcade Learning Environment (ALE) (Bellemare et al., 2013) as presented by Aitchison et al. (2023). The results, shown in Figure 4.2, confirm that the sensitivity-to-delay parameter has an effect on learning, with the base hyperbolic performing best on only 1 of the 5 environments. It must be noted that Atari-5 is a representative benchmark for the full 57-environment Arcade Learning Environment Benchmark (ALE), and as such, the results are expected to reflect on the full suite of environments as well. We also note that different environments behave differently to the sensitivity-to-delay parameter, meaning that this hyperparameter may need to be carefully tuned before it can be applied. In our work, we did not do any hyperparameter tuning and selected 3 representative values from the range $[0, 1]$.

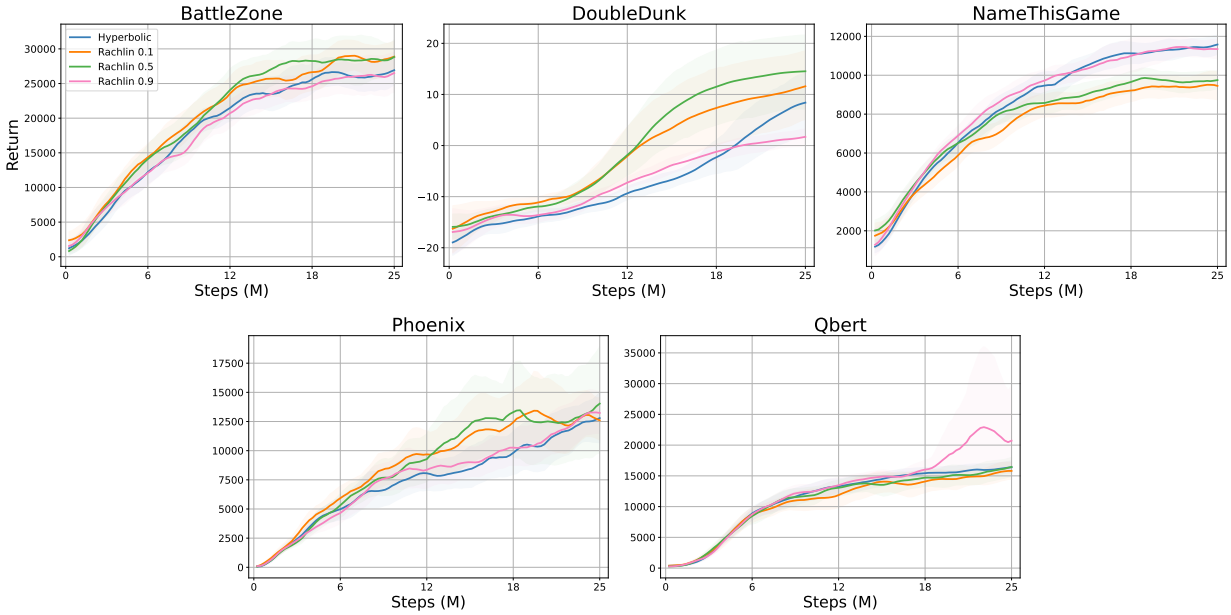


Figure 4.2: Performance of Rachlin hyperbolic discounting for 3 different values of sensitivity-to-delay (σ) on the Atari-5 benchmark (Aitchison et al., 2023). The mean (dark line) and 95% bootstrapped confidence interval (shaded region) are shown, calculated over 5 seeds for each experiment.

4.4.2 Procgen

Figure 4.3 shows the performance of Rachlin hyperbolic discounting for 3 different values of sensitivity-to-delay (σ) against the one-parameter Hyperbolic discounting (baseline) on all Procgen environments for 25M timesteps. The performance of the baseline one-parameter hyperbolic is fairly close to the 3 Rachlin models studied here, except in bigfish, coinrun and starpilot environments.

4.5 Conclusion

The conventional exponential reward discounting model in reinforcement learning inadequately captures the intricacies of human decision-making processes. The subjective value of a reward varies between individuals and is influenced by their perception of the time required to obtain the reward. By integrating sensitivity-to-delay models borrowed from psychological sciences literature into reinforcement learning agents, We have demonstrated

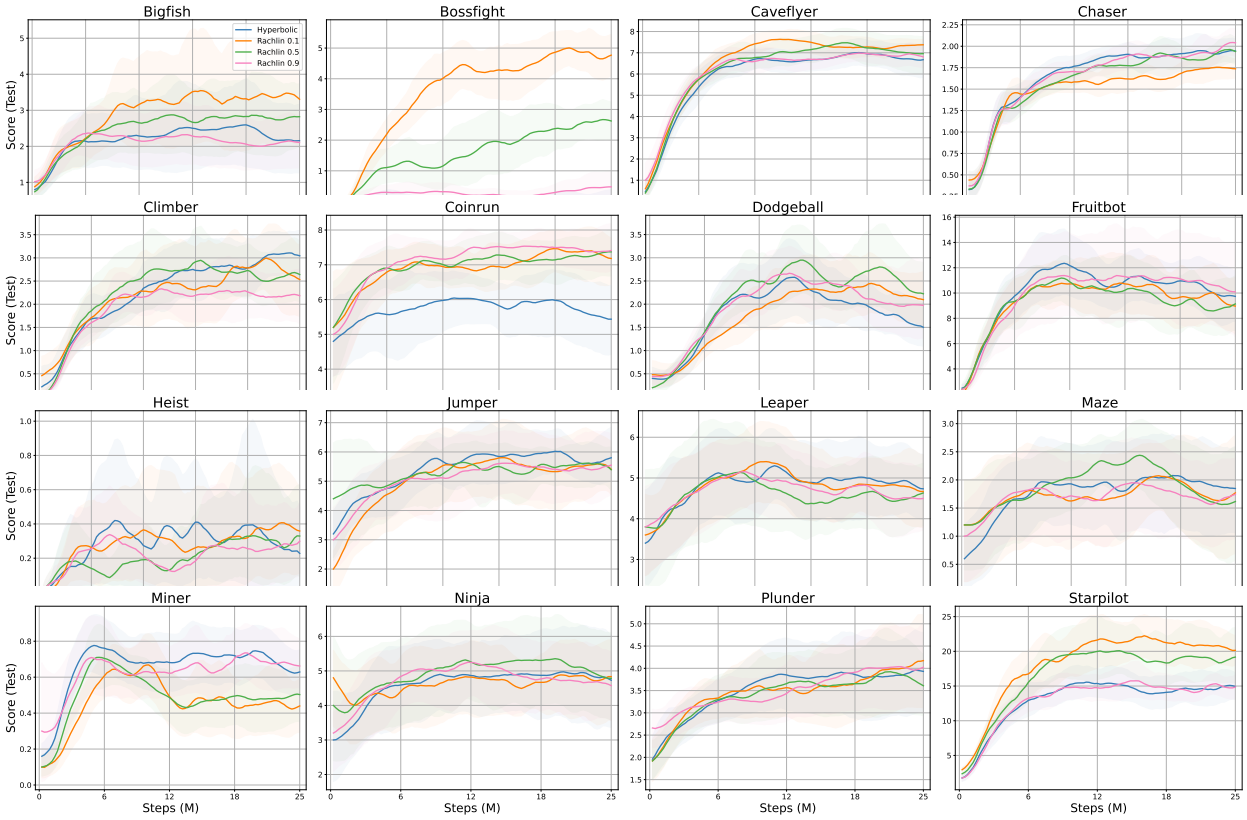


Figure 4.3: Performance of Rachlin hyperbolic discounting for 3 different values of sensitivity-to-delay (σ) against the one-parameter Hyperbolic discounting (baseline) on the Procgen benchmark. Mean and 95% confidence intervals are shown.

a method to incorporate subjective value and accommodate preference reversals. This approach holds promise for the development of AWe agents that emulate human behavior more accurately, enhancing their social acceptability and facilitating smoother collaboration with humans.

In this work, we have introduced a novel approach to integrating the Rachlin hyperbolic discounting model into the deep reinforcement learning framework. By modifying the reward function and the value estimation process to incorporate both the discount factor and the sensitivity-to-delay parameter, our approach enables RL agents to learn policies that better align with human preferences and decision-making patterns in tasks involving intertemporal trade-offs.

Our results demonstrate the potential of this approach to improve decision-making and performance in delayed gratification scenarios and other tasks that require long-term plan-

ning and decision-making. Additionally, we have explored the impact of the sensitivity-to-delay parameter on the agent's behavior, highlighting the importance of carefully tuning this parameter to achieve the desired decision-making patterns. The optimal value of σ appears to exhibit variability across different environments, indicating that environmental dynamics significantly influence the selection of this parameter. Fine-tuning σ during hyperparameter optimization may enhance the performance of Rachlin discounting.

Future work could explore the integration of our approach into more complex RL algorithms, such as multi-agent reinforcement learning or hierarchical reinforcement learning frameworks. Additionally, further investigation into the theoretical properties and convergence guarantees of our approach would be valuable, as well as exploring alternative formulations of the two-parameter delay discounting function.

Chapter 5

Beyond Exponential Discounting in Multi-Agent Reinforcement Learning

Aligning AI systems with human preferences, goals, and values is an important research objective. Research from psychology and neuroscience shows that humans discount future rewards non-exponentially, which also captures preference reversals. Motivated by this theme, we study non-exponential discounting, specifically hyperbolic discounting, in the multi-agent setting, where each agent discounts its future rewards according to the hyperboloid function ($\frac{1}{1+kt}$). This approach allows us to study the role of discounting on agent interactions beyond individual learning, especially where AI agents may be paired with humans. To this effect, we lay the theoretical and practical foundations of using hyperbolic discounting in six contemporary multi-agent deep reinforcement learning (MARL) methods, covering different classes such as independent learning, centralized multi-agent policy gradient, and value decomposition. We evaluate two different weighting schemes within hyperbolic discounting and compare them with exponential discounting (γ^t) on a set of four diverse cooperative multi-agent tasks. Our results show that hyperbolic discounting consistently achieves higher returns across all tasks, especially in sparse reward and coordination-specific settings. Our work can be seen as extending human-preference modeling in multi-agent systems.

5.1 Introduction

Behavioral tendencies in humans and animals often exhibit a preference for immediate rewards over delayed ones. This inclination, observed in various scenarios from foraging behavior to economic decisions, is rooted in the concept of discounting, where the perceived value of rewards diminishes over time. Traditional economic models favor an exponential discount function with a fixed interest rate, aligning with the rationality of consistent decisions over time. However, empirical studies suggest that human behavior more closely follows a hyperbolic discounting curve, characterized by preference reversals and inconsistent choices over time. This discrepancy challenges the conventional rationality of exponential discounting, proposing that hyperbolic discounting might be optimal under certain conditions like uncertain risks.

Silver et al. (2021) propose that maximizing rewards is sufficient for learning intelligent behavior. By prioritizing the singular goal of reward maximization, complex behavior can be generated without requiring specialized problem formulations for individual skills. Thus, the *treatment of the reward signal* plays a central role in our quest for intelligent agents. In the reinforcement learning (RL) problem setting, the objective is to maximize cumulative rewards over time and discounting plays a crucial role, particularly in infinite horizon objectives, to ensure well-defined long-term reward objectives. It not only influences the time-preference for rewards, impacting shortest path strategies for autonomous agents, but also represents the termination probability in various scenarios. However, the standard RL approach with exponential discounting fails to align with observed human and animal value preferences, which tend to discount future returns hyperbolically. While this behavior was previously deemed irrational, it aligns mathematically with an agent's uncertainty over environmental risks (Mazur, 1997), suggesting a need to rethink discounting models in RL to better mirror human decision-making under uncertainty, especially since the functional form of discounting directly shapes the solutions obtained.

AI alignment seeks to harmonize AI systems with human preferences, goals, and values. Insights from psychology and neuroscience reveal that humans naturally apply non-

exponential discounting to future rewards, a phenomenon that accounts for preference reversals and underscores the complexity of human decision-making. Inspired by these findings, our research delves into the application of non-exponential discounting, specifically focusing on hyperbolic discounting, within a multi-agent context. Here, each agent employs the hyperboloid discounting function ($\frac{1}{1+kt}$), a departure from traditional exponential models, to evaluate its future rewards. This novel approach enables an in-depth exploration of reward discounting's impact on agent interactions, particularly in scenarios that involve human-AI collaboration, thereby extending the modeling of human preferences in AI systems.

In this study, we establish both theoretical and empirical frameworks for integrating hyperbolic discounting into six multi-agent deep reinforcement learning (MARL) strategies, spanning independent learning, centralized multi-agent policy gradient, and value decomposition classes of methods. Our investigation includes a comparative analysis of two hyperbolic discounting weighting schemes against the conventional exponential discounting model (γ^t) across four cooperative multi-agent tasks. The findings reveal that hyperbolic discounting consistently outperforms exponential discounting, yielding higher returns, especially in environments characterized by sparse rewards and the need for intricate coordination. These enhancements were evident across various learning modalities and were particularly pronounced in the Centralized Training Decentralized Execution (CTDE) framework.

5.2 Related Work

5.2.1 Psychology and Cognitive Sciences

In controlled studies, discounting future rewards has been mostly studied as a personal preference parameter, where each individual is given a questionnaire to evaluate their valuation of future rewards. These studies show how decisions involving immediate versus long-term benefits are influenced by temporal discounting—where individuals place less value on delayed rewards. Recent studies have expanded this concept to decisions made in group settings, like dyads or small groups, revealing that direct interactions can lead to aligned preferences

among participants, making them more similar in patience level over time. Bixter and Luhmann (2021) study whether such social influences could also be indirect, such as through mutual acquaintances within a group. Focusing on hypothetical monetary rewards, the research involved groups of three where one member's decision preferences before collaboration were linked to another's preferences after collaborating with an intermediary. Findings highlighted that decision-making tendencies regarding time can spread through a social network's connections, showing the presence of indirect social influence in a controlled setting.

5.2.2 Deep Multi-Agent RL

Several studies have explored the use of discount factors and discounting functions in single agent RL to improve the quality of learned policies and value functions. François-Lavet et al. (2015) proposed a dynamic discount rate that increases as the learning steps increase, leading to improved Q-function estimates. Xu et al. (2018) treated the discount factor as a learnable parameter and used meta-learning to optimize the discount rate. Fedus et al. (2019) proposed the first deep RL method to integrate hyperbolic and other non-exponential discounting functions using a value-based method called Rainbow. Schultheis et al. (2022) employed inverse reinforcement learning to recover properties of the discount function given decision data.

While these studies have focused on non-exponential discounting methods in the single-agent domain, there exist no known works that explore hyperbolic discounting in multi-agent RL. Based on existing literature and previous findings in single-agent RL, we feel there is a potential for such approaches to theoretically and empirically improve learning in the multi-agent domain as well. Drawing inspiration from these previous works, this work implements and evaluates two forms of hyperbolic discounting in various multi-agent algorithms.

5.3 Preliminaries

5.3.1 Markov Games

We consider cooperative Markov games for N agents (Littman, 1994) with partial observability defined by the tuple $(N, S, \{O_i\}_{i\in N}, \{A_i\}_{i\in N}, \Omega, P, \{R_i\}_{i\in N})$, with agents $i \in N = \{1, \ldots, N\}$, state space S, joint observation space $O = O_1 \times \ldots \times O_N$, and joint action space $A = A_1 \times \ldots \times A_N$. Each agent i only perceives local observations $o_i \in O_i$, which depend on the state and applied joint action via the observation function $\Omega : S \times A \to \Delta(O)$. The function $P : S \times A \to \Delta(S)$ returns a distribution over successor states given a state and a joint action. $R_i : S \times A \times S \to \mathbb{R}$ is the reward function for agent i, which gives its individual reward r_i. The objective is to find policies $\pi = (\pi_1, \ldots, \pi_N)$ for all agents such that the discounted return of each agent i, $G_i = \sum_{t=0}^{T} \gamma^t r_{i,t}$, is maximized with respect to other policies in π, formally $\forall i : \pi_i \in \arg\max_{\pi_i} \mathbb{E}[G_i|\pi_i, \pi_{-i}]$ where $\pi_{-i} = \pi \backslash \{\pi_i\}$, γ is the discount factor, and T is the total timesteps of an episode.

5.3.2 Hyperbolic Discounting for Value-Based Methods

In line with Fedus et al. (2019), we revisit the technique that utilizes exponentially discounted Q-values to derive hyperbolic discounted Q-values. The Bellman equation (Bellman, 1957a) is written as:

$$Q_\pi^{\gamma^t}(s, a) = \mathbb{E}_{\pi,P}[R(s, a) + \gamma Q_\pi(s', a')] \tag{5.1}$$

We establish a connection between hyperbolic $Q_\pi^{\Gamma_k}$-values and values obtained through standard Q-learning. The hyperbolic discount Γ_k can be represented as the integral of a specific function $f(\gamma, t)$ for $\gamma = [0, 1)$:

$$\int_0^1 \gamma^{kt} d\gamma = \frac{1}{1 + kt} = \Gamma_k(t) \tag{5.2}$$

The integration of the function $f(\gamma, t) = \gamma^{kt}$ across the domain $\gamma \in [0, 1)$ results in the

hyperbolic discount factor $\Gamma_k(t)$. This integration, incorporating an infinite set of exponential discount factors γ, reveals that γ^{kt} functions as the standard exponential discount factor, linking the concept to traditional Q-learning. This approach suggests that by aggregating an infinite collection of γ values, hyperbolic discounts can be derived for each respective time step t. For a hyperbolic discount function $\Gamma_k(t)$, the hyperbolic Q-values can be written as:

$$
\begin{aligned}
Q^{\Gamma_\pi(s,a)} =& \mathbb{E}_\pi \left[\sum_t \Gamma_k(t) R(s_t, a_t) \middle| s, a \right] && (5.3)\\
=& \mathbb{E}_\pi \left[\sum_t \left(\int_{\gamma=0}^{1} \gamma^{k^t} d\gamma \right) R(s_t, a_t) \middle| s, a \right] && (5.4)\\
=& \int_{\gamma=0}^{1} \mathbb{E}_\pi \left[\sum_t R(s_t, a_t)(\gamma^{kt}) \middle| s, a \right] d\gamma && (5.5)\\
=& \int_{\gamma=0}^{1} Q_\pi^{(\gamma^{kt})}(s, a) d\gamma && (5.6)
\end{aligned}
$$

5.3.3 Hyperbolic Discounting for Policy Gradient Methods

In policy gradient methods, particularly Actor-Critic variants utilizing the Advantage function for value estimation, we build upon previous work that introduced hyperbolically discounted advantage (Nafi et al., 2022b) in single-agent settings. Advantage is defined as $A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$. It follows from Eq. 5.6 that the hyperbolic advantage can be written as:

$$
\begin{aligned}
A_\pi^\Gamma(s_t, a_t) &= \int_0^1 A_\pi^{(\gamma^k)^t}(s_t, a_t) d\gamma && (5.7)\\
&= \int_0^1 \left[Q_\pi^{(\gamma^k)^t}(s_t, a_t) - V_\pi^{(\gamma^k)^t}(s_t) \right] d\gamma && (5.8)\\
&= \int_0^1 \left[Q_\pi^{(\gamma^k)^t}(s_t, a_t) \right] d\gamma - \int_0^1 \left[V_\pi^{(\gamma^k)^t}(s_t) \right] d\gamma && (5.9)\\
&= \int_0^1 \left[r_t + \gamma^k V_\pi^{(\gamma^k)^t}(s_{t+1}) \right] d\gamma - \int_0^1 \left[V_\pi^{(\gamma^k)^t}(s_t) \right] d\gamma && (5.10)
\end{aligned}
$$

where we compute $V^{(\gamma^{kt})}$ in a similar manner as $Q^{(\gamma^{kt})}$ from Eq. 5.6 for each of the n_γ heads by using a multi-head architecture where each head corresponds to the value function for each γ^k. By computing the value function over all γ^k where $0 \leq \gamma < 1$, we estimate hyperbolically discounted advantage. Following Fedus et al. (2019), we consider a finite set of γ^k to approximate the advantage through a Riemann sum:

$$A^{\Gamma}_{\pi}(s,a) \approx \sum_{\gamma_i} (\gamma_{i+1} - \gamma_i) w(\gamma) A^{(\gamma^k)_i}_{\pi}(s,a) \tag{5.11}$$

This is done in the Critic part of the Actor-Critic network, which then supplies the Advantage to the Actor for optimizing the objective function. For the Critic's value estimation learning, we minimize the average of the losses calculated for these multiple γ^k such that the loss function corresponding to each γ^k is defined as:

$$L^{\gamma^k}_{v}(\theta) = \hat{\mathbb{E}}_t \left[\left(V^{\gamma^k}_{\theta}(s_t) - \hat{V}^{\gamma^k}_{targ} \right)^2 \right] \tag{5.12}$$

5.4 Hyperbolic Discounting in MARL

5.4.1 Weighting Schemes

Discounting is a fundamental concept in reinforcement learning that allows agents to prioritize between immediate and future rewards during the decision-making process. While exponential discounting has been the standard approach, it falls short in capturing preference reversals, and accurately representing the agent's changing uncertainty regarding its survival in subsequent states. Although various non-exponential discounting techniques have been suggested for single-agent scenarios, such discounting functions in multi-agent environments remain largely unexplored. To address this gap, we introduce two novel discounting methods specifically designed for multi-agent reinforcement learning algorithms: hyperbolic discounting and averaged horizon discounting. The average horizon discounting is a special case of

hyperbolic discounting where the agent learns over multiple discount factors γ and averages the resulting value-estimates for action selection as well as learning (loss calculation and back-propagation). These approaches aim to better capture the complexities and dynamics of multi-agent interactions while optimizing long-term performance.

Hyperbolic Discounting

Hyperbolic discounting was introduced theoretically as a way to model human-time preferences into reinforcement learning agents ((Ainslie, 1975), see Maia (2009) for an overview of RL and the brain). Hyperbolic discounting was implemented for temporal difference learning (Alexander and Brown, 2010). As discussed in Section 5.3.2, Fedus et al. (2019) introduced a practical method for implementing a hyperbolic discounting function. Their approach involved using a multi-headed value output structure where each value estimate was linked to a distinct discount factor. Subsequently, they computed all the outputs and performed an integral approximation using a Riemann sum across the multiple values to estimate a hyperbolic function.

$$Q^{\Gamma}(s,a) \approx \sum_{(\gamma_{i+1})\subseteq\mathcal{G}} (\gamma_{i+1}-\gamma_i) w(\gamma_i) \cdot Q^{\gamma_i}(s,a) \tag{5.13}$$

where $\mathcal{G}$ is a set of discount factors given by:

$$\mathcal{G} = [\gamma_0, \gamma_1, \cdots, \gamma_{n_\gamma}] \tag{5.14}$$

In Equation 5.13, hyperbolic value estimates are represented by Q on the left side. The equation also features individually discounted outputs from the networks, symbolized as $Q^{(\gamma_i)}$, where γ_i represents varying discount rates within the set $\mathcal{G}$ given by Equation 5.14. This work introduces a novel approach to discounting, along with an algorithmic framework designed to facilitate the creation of new discounting functions by employing a multi-headed output strategy. Our previous work ((Ali et al., 2023a; Nafi et al., 2022b)) explored the advantages of a multi-horizon framework. By utilizing the same multi-headed output ap-

proach, we can extend the application of hyperbolic discounting to multi-agent algorithms that depend on a value function for their operation.

Averaged Horizon Discounting

Model averaging is a widely adopted ensemble technique in the field of machine learning. It entails combining the predictions of multiple machine learning models by computing the average of their respective outputs. This approach has been shown to yield more accurate and robust predictions Arpit et al. (2022). Furthermore, a study conducted by Churchill et al. (2023) demonstrated that the error variance in an ensemble model composed of K individual models was found to be inversely proportional to K. This finding indicates that the ensemble model effectively reduces the error by a factor of K when compared to the individual models constituting the ensemble.

Since our model architecture is learning over multiple heads, it already constitutes a partial ensemble model. Moreover, as shown in Equation 5.13, the output Q-values from each head can be combined using a Riemann Sum to approximate a hyperbolic Q-value, where the weights are the difference between successive γ values in the set $\mathcal{G}$. Inspired by the average ensemble method, we devise a new technique that involves computing the average across multiple value predictions, each incorporating a distinct discount factor, thereby integrating diverse discounting horizons into the overall value estimation process. Mathematically, it can be expressed as:

$$Q^{\Gamma}(s,a) \approx \frac{1}{|\mathcal{G}|} \sum_{\gamma_i \subseteq \mathcal{G}} Q^{\gamma_i}(s,a) \tag{5.15}$$

The final value estimate is represented by Q^{Γ} on the left-hand side of the equation. On the right-hand side, a summation is performed over all values of Q^{γ_i}, which is then divided by the number of discount rates denoted by $|\mathcal{G}|$. By leveraging the multi-headed output framework introduced by Fedus et al. (2019), we can partially ensemble our model by utilizing the same feature extraction layers and then branching out to separate output layers for each discount rate. This approach allows us to train multiple output layers for

different discount rates using the same set of feature layers, thereby leveraging the strengths of an ensemble method similar to those proposed by Arpit et al. (2022) and Churchill et al. (2023). During inference, the value estimates for each discount rate are computed using their respective output layers and subsequently averaged using the aforementioned equation to obtain the final value estimate. This technique strikes a balance between the benefits of ensemble methods and the computational cost associated with training and deploying a fully-fledged ensemble model.

5.4.2 Network Architectures

Integrating non-exponential discounting methods into multi-agent reinforcement learning algorithms requires careful consideration of the neural network design, as these networks are trained using values of future rewards that have been discounted to their present value. This work explores the three main types of neural network architectures found in MARL algorithms:

1. **Value-based Q-networks:** These networks predict the value of taking certain actions in given situations, and usually there is a separate network for each agent, or in case there is a joint value network for all agents, some type of agent indication is used an input to the network to distinguish the network for each agent.

2. **Value decomposition networks:** This network learns to optimize a joint value estimate and helps in understanding how individual agents contribute to the collective goal, facilitating better coordination among them.

3. **Centralized policy gradient:** This network incorporates two key components – an actor that proposes actions and a centralized critic that evaluates these actions. This setup helps in refining the learning process by continuously adjusting based on the critic's feedback.

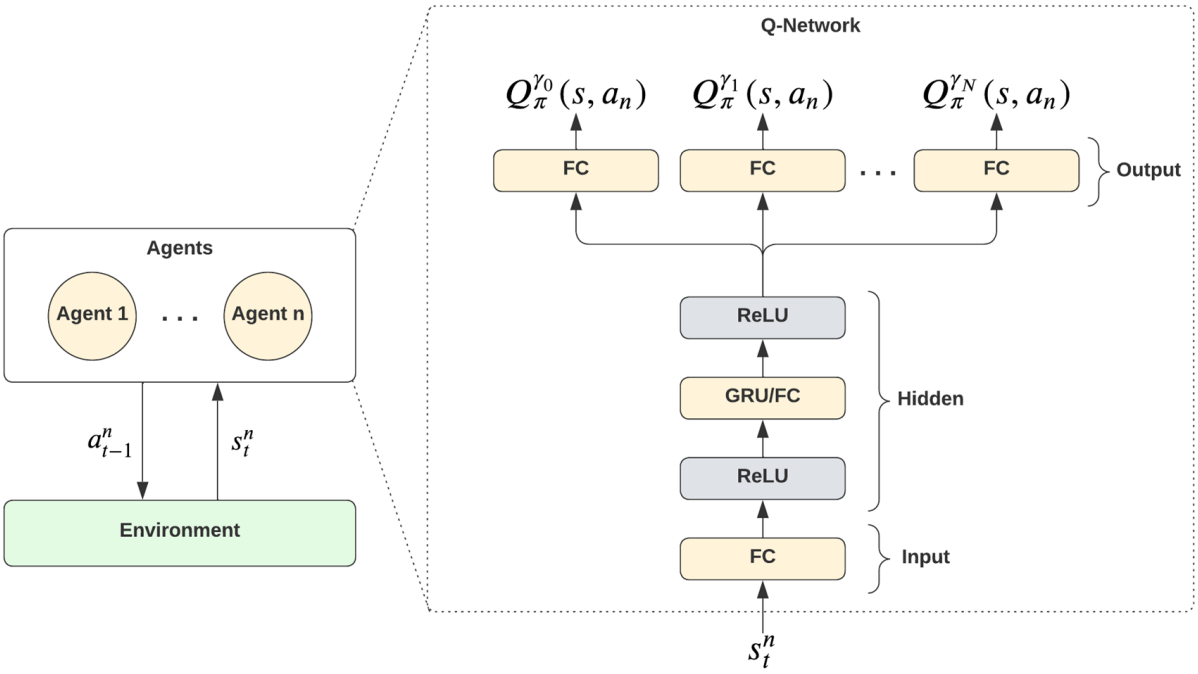


Figure 5.1: Network architecture for hyperbolic discounting in IQL.

Value-Based Q-Networks

Value-based networks approximate the value function of a state or action-value function of a state action pair. The value function represents the expected cumulative reward an agent can obtain by following a certain policy from a given state and forms the basis of most temporal difference learning methods, such as Q-learning. Such networks usually learn the optimal Q-values by iteratively updating the Q-value estimates based on the Bellman equation (Watkins and Dayan, 1992). In deep MARL, Q-networks were the first to be employed to study the interaction and learning of multiple agents simultaneously. We explain the construction of the basic Q-Network and how it can be modified to incorporate non-exponential discounting.

We use a Multi-Layer Perceptron (MLP) network for this work instead of using a CNN since the observation space is non-image based. The network's architecture consists of fully connected (FC) layers, segmented into the input layer for agent observations, hidden layers for representation learning, and the output layer for Q-value generation from the learned features. ReLU activation function to used to introduce non-linearity. The hidden layers

can either be an FC layer or Gated Recurrent Units (GRU) layer, which is a type of recurrent neural network architecture that effectively captures temporal dependencies in sequence data.

The use of multiple agents necessitates a change in how each agent's behavior and learning are modeled through the network, leading to two distinct configurations, with and without parameter sharing. In setups without parameter sharing, each agent operates with its unique set of network parameters. Conversely, with parameter sharing, all agents utilize a common set of network parameters, with the network receiving additional inputs identifying each agent through a one-hot vector (Gupta et al., 2017). This approach enables agents to develop distinct behaviors despite shared parameters. The collective loss from all agents is used to optimize these shared parameters.

While standard Q-networks output a singular set of Q-values, our design introduces modifications to the output layer to accommodate non-exponential discounting based on the works of Fedus et al. (2019). As shown in Figure 5.1, the input state/observation of individual agents is represented as s_t^n, where n indicates a unique agent. The output layer produces a set of Q-values for each agent n instead of a single Q-value, represented as $Q_\pi^{\gamma_i}$. The modified network's output layer features N fully connected (FC) layers, corresponding to the number of discount factors. For each discount factor, a distinct set of Q-values, $Q_\pi^{\gamma_i}$, is produced by its respective layer, where γ_i is a unique discount factor.

It is important to note here that these value-based Q-networks incorporate learning for multiple agents, where each agent is learning its own set of multi-headed Q-values. However, there is no joint Q-value that is being learned, something which we discuss next in a category of methods that learn a joint Q-value and then decompose it into individual agent Q-values.

Joint-Value Based Networks

Value decomposition techniques extend value-based methods for multi-agent cooperation and coordination. The core idea is to facilitate cooperation among agents by decomposing the collective value function into individual components corresponding to each agent. This decomposition allows for the optimization of joint actions while maintaining the autonomy

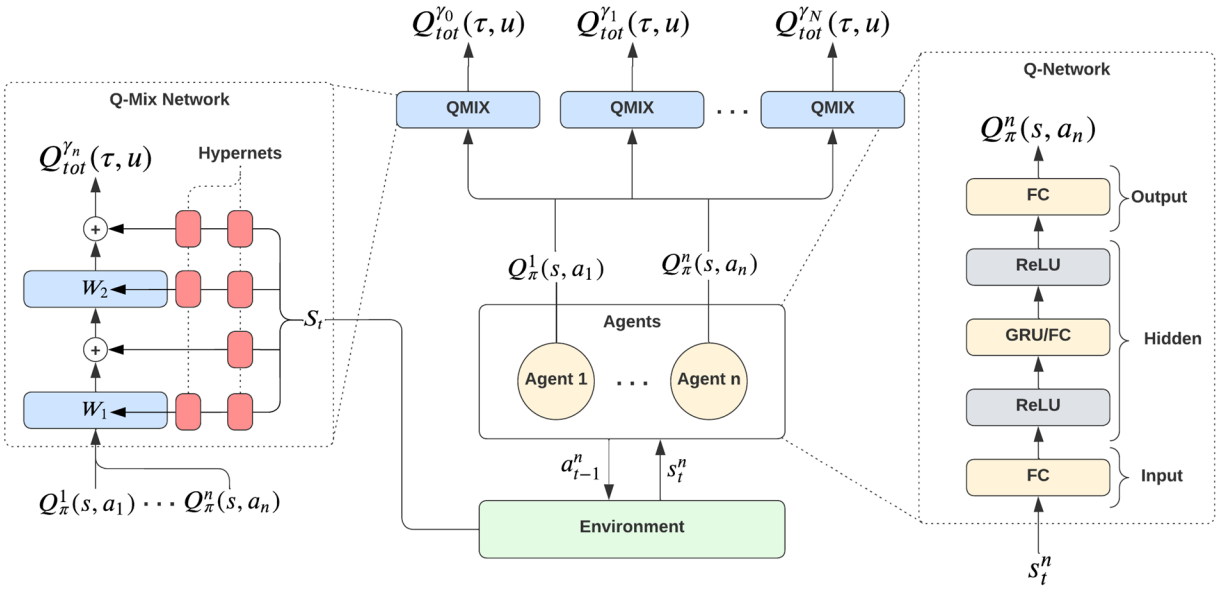


Figure 5.2: Network architecture for hyperbolic discounting in QMIX.

of individual agent decision-making processes. This idea is based on factored value functions (Koller and Parr, 1999), which approximates a joint value function as a linear combination of local value functions.

One of the first methods to introduce this technique in Deep MARL was the Value Decomposition Network (VDN), where each agent learns its own value function, which is then combined to form a global joint-value (Sunehag et al., 2017). During training, the joint Q-value is learned in a centralized manner by summing up the individual Q-values. However, during execution, each agent can act independently using just its individual Q^i, allowing for decentralized execution. This decomposition simplifies the learning problem in several ways. It avoids the exponential blow-up in the joint action space by conditioning each Q^i only on the local action. It is able to address the multi-agent credit assignment problem, as each Q^i is explicitly learned for the contribution of agent i. Finally, it handles partial observability better since each Q^i depends only on the global state s, which can be estimated from local observations.

Another method called the QMIX (Rashid et al., 2018) was proposed as an extension of VDN to represent more complex decomposition. QMIX employs a non-linear mixing network

to combine the individual agent value functions. The mixing network incorporates hypernetworks (a "hypernetwork" is a network to generate the weights for another larger network) that allow for the nonlinear combination of individual Q-values. The mixing network takes the individual agent Q-values as input and produces a global value, called Q-total Q_{total}. This Q-total value is then used to guide the action selection process for each agent. By incorporating a non-linear mixing network, QMIX enables more sophisticated modeling of agent interactions, leading to improved coordination in multi-agent tasks.

In this work, we employ QMIX as a model for incorporating non-exponential discounting within the framework of value decomposition methods, also known as value factorization approaches. Within QMIX, individual agents generate Q-values utilizing the foundational Deep Q-Network architecture (Mnih et al., 2013). These Q-values, produced by each agent's Q-network, are forwarded to a mixing network tasked with producing a combined action value. This mixing network is constructed from hypernetworks that create weight matrices and biases, enabling the nonlinear integration of individual Q-values. Illustrated in Figure 5.2, the individual Q-values, represented as Q_π^n, are inputs for the mixing network. This network utilizes the created weights and biases to merge these Q-values into a collective value, termed Q_{total}. This collective value, Q_{total}, subsequently informs the action choices of each agent, fostering collaboration while preserving the autonomy of each agent.

To implement variable discounting within QMIX, we adopted the methodology introduced by Fedus et al. (2019), establishing several output layers for distinct discount rates, γ. Illustrated in Figure 5.2, the Q-values generated by each agent's Q-network, denoted as Q_π^n, are fed into the mixing network. This network leverages devised weights and biases to aggregate these Q-values into a unified value, Q_{total}. Given the necessity to channel Q-values through a mixing network, we preserved the foundational network design by Mnih et al. (2013) while integrating a distinct mixing network for every different discount rate. Consequently, once agents determine their individual Q-values, marked as Q_π^n in Figure 5.2, these values are input into their respective mixing networks, producing individual Q_{total} values. These are indicated as $Q_{\text{tot}}^{\gamma_i}$ in Figure 5.2, with γ_i indicating a specific discount factor.

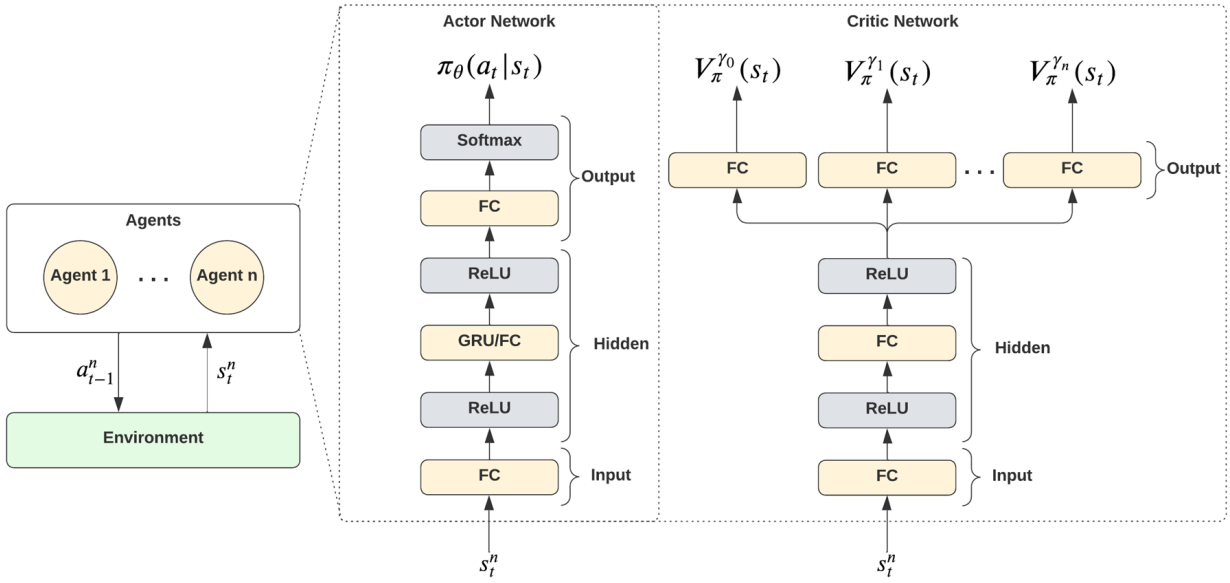


Figure 5.3: Network architecture for hyperbolic discounting variants for Actor-Critic methods used in IA2C, IPPO, MAA2C, MAPPO.

Policy Gradient Actor-Critic Networks

Unlike value-based approaches, policy-based strategies optimize an action's probability distribution rather than a value function, facing challenges like high variance and slow progress in convergence. Actor-critic methods emerge as a solution, merging value-based and policy-based methods' benefits. They incorporate a critic network for value function estimation and an actor network for action selection based on the evaluated policy, as influenced by the critic's assessments. Among the notable actor-critic frameworks are Proximal Policy Optimization (PPO) (Schulman et al., 2017) and Advantage Actor-Critic (A2C) (Mnih et al., 2016). PPO employs a surrogate objective to facilitate minor policy adjustments, enhancing sample efficiency. Conversely, A2C leverages the estimated value function to calculate the relative advantage of actions, enabling more effective updates for both actor and critic networks. Both PPO and A2C advance beyond traditional policy gradient techniques by utilizing actor and critic networks for improved performance.

In this framework, the actor network is tasked with learning a policy that maps the current state to a probability distribution over actions. This begins with an input layer

that interprets the agent's observations, followed by several hidden layers aimed at learning the underlying representations of the input state. The final output layer is responsible for producing a probability distribution over actions, typically employing a Softmax activation function for discrete action scenarios or a Gaussian distribution for continuous action environments. The critic network's role is to approximate the value function, thereby evaluating the quality of a given state. This network also starts with an input layer, builds through several hidden layers, and concludes with an output layer that issues a singular scalar value indicative of the current state's estimated value. Throughout the training process, the actor network's adjustments are informed by the policy gradient, which is calculated using the critic's value estimates, while the critic network is refined through temporal-difference learning for value function approximation.

Given that the actor network outputs a probability distribution across actions rather than discrete values, we introduce a multi-headed critic network designed to output a separate value function for each discount factor. The critic network architecture processes an agent's observed state, s_t^n, and generates individual value functions, represented as $V_\pi^{\gamma_N}$, as shown in Figure 5.3. These value functions are then used to compute the advantage function of state, action pairs. The advantage function is then used in the optimization objective of the actor network. This technique of using the hyperbolic advantage function first proposed in (Nafi et al., 2022b) for use in PPO is equally applicable to A2C and can be extended to any actor-critic method given the common architectural foundation in use with actor-critic algorithms.

5.4.3 Multi-Agent Algorithms

To assess a fair evaluation of non-exponential discounting, a wide range of algorithms was selected to cover multiple approaches in multi-agent reinforcement learning. Using the network modifications and mathematical formulations presented in the previous sections, our algorithms are now suited for non-exponential discounting strategies. In order to evaluate the effects of non-exponential discounting in multi-agent reinforcement learning, a diverse

range of algorithms was selected for this work. These algorithms can be grouped into two primary categories: Independent Learning and Centralized Learning Decentralized Execution (CTDE). The IL category contains IQL, IA2C, and IPPO, while the CTDE category contains QMIX, MAA2C, and MAPPO, in which QMIX is a value factorization/decomposition method, and MAA2C and MAAPO are centralized policy gradient methods. By modifying the network structures and using the proposed mathematical formulations mentioned in the previous discussion, these algorithms have been adapted to incorporate non-exponential discounting functions. In this discussion, we emphasize the key differences and similarities between the algorithms and demonstrate how non-exponential discounting strategies are integrated into their learning and optimization processes.

Independent Learning (IL)

In independent learning, single-agent reinforcement learning algorithms are applied to individual agents without the consideration of a multi-agent structure. Each agent perceives other agents as part of the environment, treating them as dynamic and unpredictable elements rather than explicitly acknowledging their presence as separate learning entities (Papoudakis et al., 2021). This leads to a lack of communication and coordination among agents during both the training and execution phases (Zhang et al., 2021). While independent learning allows for simpler and more efficient training processes, this approach may result in suboptimal and less coordinated behaviors, especially in tasks where agent collaboration is crucial for success. Moreover, the agents may not adapt well to changes in the strategies of their counterparts or be able to handle emergent behaviors that arise from interactions within the multi-agent system. Despite these limitations, independent learning serves as a baseline for evaluating more advanced multi-agent reinforcement learning methods. For this work, We have selected and documented implementation details for these three independent learning algorithms:

Independent Q-Learning (IQL). IQL utilizes a decentralized value-based Q-network to approximate Q-values for the individual observations of each agent (Tan, 1993). Each agent

then updates the Q-network by minimizing the Q-learning loss through the calculation of a temporal difference (TD) error. In traditional methods of discounting, a single discount rate γ is used during the TD calculation. With our modification, the Q-network outputs a set of Q-values for each unique discount rate. These sets of values are then used to calculate separate TD targets, each using its corresponding discount rate Ali et al. (2022); Fedus et al. (2019). During action selection, we use the discounting functions, as presented earlier, to combine a set of Q-values values for different discount factors into a single set of Q-values, using either hyperbolic or averaged horizon discounting. The final set of Q-values is then used for action selection through an *argmax*.

Independent Advantage Actor-Critic (IA2C) This method uses a decentralized actor-critic network to optimize its policy and estimate state values based on individual observations. The actor network in IA2C maps the agent's state to a probability distribution over actions, while the critic network estimates the state-value function. To update both networks, IA2C calculates the advantage function using the values of a critic network. This advantage function helps in determining how much better a specific action is compared to the average actions taken in the current state. Each agent independently updates its actor and critic networks by minimizing the policy gradient loss and the TD error. Similar to the modifications made to IQL, IA2C utilizes multiple discount rates for value estimations. However, due to the action selection relying on a probability distribution instead of values, the advantage function is used in the optimization objective, as opposed to a state-action function.

Independent Proximal Policy Optimization (IPPO) This method follows a similar decentralized approach as IA2C, with each agent using its own actor-critic network for policy optimization and state value estimation (De Witt et al., 2020). The key difference between IPPO and IA2C lies in the optimization process. IPPO employs a trust region optimization approach, which uses a ratio of new and old policies in the surrogate objective function to ensure small, stable updates to the policy and improve sample efficiency (Schulman et al., 2017).

Similar to the modifications made in IA2C, adapting IPPO to utilize a non-exponential discounting function requires the calculations of independently discounted advantages. These advantages are then passed through a non-exponential discounting function and then used for optimization.

Centralized Learning Decentralized Execution (CTDE)

In contrast to independent learning, centralized training decentralized execution (CTDE) methods address the challenges in multi-agent reinforcement learning by allowing agents to share information during the training phase while maintaining decentralized executions (Papoudakis et al., 2021). This approach enables agents to learn coordinated policies by utilizing a global perspective and accounting for the actions and observations of other agents during training. However, during the execution phase, each agent makes decisions independently based on its own observations and learning policy, without relying on any direct communication or information sharing with other agents. This balance between centralized learning and decentralized execution enables CTDE methods to improve coordination and performance in multi-agent tasks while retaining the scalability and robustness associated with decentralized systems. We have selected and documented implementation details for these three CTDE algorithms:

Q-Value Mixing Network (QMIX) is a cooperative multi-agent reinforcement learning algorithm that combines the advantages of centralized learning with decentralized execution to tackle coordination challenges in multi-agent settings (Rashid et al., 2018, 2020)). In QMIX, the agents employ decentralized Q-networks to approximate individual action-value functions, while a centralized mixing network is utilized to aggregate the agents' Q-values into a joint action-value function. The mixing network is designed to be a monotonic function, ensuring that the optimal joint-action can be derived from the optimal individual actions. Since individual Q-values of each agent are combined to compute the joint-action values, the discounted estimations are based on the mixing network outputs instead of the individual Q-values. In order to implement a non-exponential discounting function, additional mixing

networks are introduced to compute separate Q-totals corresponding to individual discount factors. These separate Q-totals are then aggregated through the discounting functions proposed to compute a total loss that is then backpropagated to the individual agents.

Multi-Agent Advantage Actor-Critic (MAA2C) is a cooperative multi-agent reinforcement learning algorithm that builds upon the actor-critic framework by incorporating centralized learning to enhance coordination between agents. Similar to IA2C, MAA2C employs decentralized actor networks for each agent, mapping individual observations to probability distributions over actions. However, unlike IA2C, MAA2C utilizes a centralized critic network, which considers the joint observations of all agents to estimate the state-value function (Lyu et al., 2021). This centralized learning aspect enables MAA2C to capture the interactions between agents, which improves coordination among agents. For incorporating non-exponential discounting, we follow a similar setup to IA2C and utilize multiple discount rates for value estimations. Since MAA2C relies on a probability distribution for action selection, similar to IA2C, the discounting function is applied to the advantage estimations. The difference between MAA2C and IA2C is the inputs to the critic networks. MAA2C aggregates the observations of all agents to combine a joint-observation input, whereas IA2C keeps these observations separated. The actual learning algorithm implementation of non-exponential discounting itself remains the same as IA2C.

Multi-Agent Proximal Policy Optimization (MAPPO) MAPPO (Yu et al., 2022) is a cooperative multi-agent reinforcement learning algorithm that extends the Proximal Policy Optimization (PPO) method (Schulman et al., 2017) to accommodate multiple agents, focusing on enhanced coordination and stable policy updates in complex environments. Similar to MAA2C, MAPPO utilizes a centralized critic network and a decentralized actor network. The key difference between MAPPO and MAA2C lies in the optimization process, similar to the differences between IA2C and IPPO. MAPPO employs a trust region optimization approach, which uses a ratio of new and old policies in the surrogate objective function to ensure small, stable updates to the policy and improve sample efficiency. However, similar to

the modifications made in MAA2C, adapting MAPPO to utilize a non-exponential discounting function requires the calculations of independently discounted advantages corresponding to unique discount rates. These advantages are then used in the optimization objective $J(\theta)$.

5.5 Experimental Setup

This study investigates the effectiveness of two non-exponential discounting methods, namely hyperbolic-weighted and hyperbolic-average discounting, across six MARL methods. These methods are categorized into two groups: three from independent learning (IQL, IPPO, IA2C) and three from centralized training with decentralized execution (CTDE) frameworks (QMIX, MAPPO, MAA2C). We evaluate these approaches in four distinct MARL environments, each offering unique challenges and opportunities to assess the scalability, generalization, and coordination capabilities of the algorithms.

5.5.1 MARL Environments

The following multi-agent environments were used in this work, an overview of which is given in Table 5.1.

Level-Based Foraging (LBF) (Albrecht and Ramamoorthy, 2013) is a grid-world scenario where agents must collect food items scattered randomly. Agents and items are assigned levels, and a group of one or more agents can collect an item if the sum of their levels is greater than or equal to the item's level. Agents can move in four directions and have an action to attempt loading an adjacent item, which succeeds based on the agents' levels. LBF allows for configuring various tasks, including partial observability or highly cooperative scenarios where all agents must participate simultaneously to collect items. Seven distinct tasks with varying world sizes, agent numbers, observability, and cooperation settings are defined to test the multi-agent reinforcement learning (MARL) algorithms' ability to cooperate in collecting resources within the grid world. We use 3 of the available 7 tasks in this work.

Multi-Robot Warehouse (RWARE) (Papoudakis et al., 2021) simulates a dynamic

warehouse scenario where robotic agents must collaborate to transport and sort packages efficiently. It is a cooperative, partially observable scenario with sparse rewards. In this grid-world warehouse simulation, agents (robots) must locate and deliver requested shelves to workstations and return them after delivery. Agents receive rewards only upon completely delivering the requested shelves. The environment features sparse and high-dimensional observations, consisting of a 3×3 grid containing information about surrounding agents and shelves. Agents can move forward, rotate, and load/unload shelves. Three tasks are defined with varying world sizes, agent numbers, and shelf requests. The sparsity of rewards and partial observability make RWARE challenging, as agents must coordinate their actions effectively to complete a series of steps before receiving any reward signal. RWARE serves as a benchmark for evaluating algorithm performance in cooperative, partially observable scenarios with sparse rewards, where effective coordination is crucial. We use all the 3 available tasks in this study.

Multi-Agent Particle Environments (MPE) (Lowe et al., 2017; Mordatch and Abbeel, 2017) consist of several two-dimensional navigation tasks designed to evaluate agent coordination and the ability to handle non-stationarity. In this study, we investigate four tasks from MPE that emphasize coordination: Speaker-Listener, Spread, Adversary, and Predator-Prey. In these tasks, agents receive high-level feature vectors as observations, including relative agent and landmark locations, and agents are required to navigate to fulfill environment-defined tasks. While the Speaker-Listener task requires binary communication and is partially observable, the remaining tasks are fully observable. The MPE tasks serve as a benchmark for assessing agent coordination and their capability to deal with non-stationarity. The rewards in these tasks are highly dependent on the joint actions of the agents, and a lack of effective coordination among individual agents can severely reduce the received rewards (Papoudakis et al., 2019). As mentioned above, we use 4 of the available 6 tasks in MPE.

The StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019) simulates battle scenarios in the real-time strategy game StarCraft, where a team of controlled agents must destroy an enemy team using fixed policies. Agents have limited observation radii and

can move and attack enemies. SMAC offers many tasks, known commonly as maps, varying in the number and types of controlled units. The key challenges include accurately estimating state values under partial observability, coordinating an increasing number of diverse agent types across tasks, and handling large action spaces as agents can select targets for healing or attacking based on their unit type. This environment serves as a complex benchmark for multi-agent reinforcement learning algorithms, demanding effective coordination, decision-making under partial observability, strategic planning, adaptability, and teamwork in realistic scenarios with large action spaces and diverse agents. We study three maps for SMAC in our work. Due to limited computing, we restrict our analysis of SMAC to QMIX, the most widely used and benchmarked method on SMAC.

Table 5.1: Overview of MARL environments (from Papoudakis et al. (2021)).

Environment	**Observability**	**Reward Sparsity**	**Agents**	**Main Difficulty**
LBF	Partial / Full	Sparse	2-4	Coordination
RWARE	Partial	Sparse	2-4	Sparse reward
MPE	Partial / Full	Dense	2-3	Non-stationarity
SMAC	Partial	Dense	2-10	Large action space

5.5.2 Evaluation Criteria

The performance of each algorithm is assessed based on several criteria, including the efficiency of learning (speed of convergence), the robustness of the learned policies, scalability to larger and more complex scenarios, and the ability to generalize to unseen environments. Additional metrics, specific to each environment, such as resource collection efficiency in LBF and RWARE, task completion time in MPE, and win rate in SMAC, are also considered. To ensure fair comparison and compensate for the higher sample efficiency of off-policy algorithms relative to on-policy ones, we follow the procedure recommended by Papoudakis et al. (2021) and adjust training steps accordingly. For MPE and LBF, on-policy algorithms (IA2C, IPPO, MAA2C, MAPPO) are trained for 20 million timesteps, and off-policy algorithms (IQL, QMIX) are trained for 2 million timesteps. In SMAC and RWARE environments, the on-policy and off-policy training is set to 40 and four million timesteps,

respectively. Despite on-policy algorithms not reusing samples via experience replay and hence resulting in lower sample efficiency, they are not considered slower. Evaluations are carried out at every 10k steps for 2M train steps, every 100k steps for 20M train steps, every 20k steps for 4M train steps, and every 200k steps for 40M train steps, maintaining the total number of evaluations to be constant (41 evaluations), are done at regular intervals throughout the training and are done for 100 episodes per evaluation.

5.5.3 Performance Metrics

To evaluate the performance of the algorithms, we consider two metrics, following the recommendations of Papoudakis et al. (2021): maximum returns and average returns. By using both, we can assess the algorithms' performance in terms of their peak capabilities and their overall learning efficiency throughout the training process.

Maximum Returns For each algorithm, we identify the evaluation timestep during training where the algorithm achieves the highest average evaluation returns across five random seeds. We report the average returns and the 95% confidence interval across these five seeds from this evaluation timestep. This metric represents the peak performance achieved by the algorithm.

Average Returns We also report the average returns achieved throughout all evaluations during training. This metric is computed over all evaluations executed during the training process, considering not only the final achieved returns but also the algorithm's learning speed.

5.5.4 Implementation Details

For each environment, experiments were done for both hyperbolic-weighted and hyperbolic-average discounting methods applied to the six selected MARL algorithms. Agents are trained for steps specified for each method and environment combination described previ-

ously, with results averaged over 5 seeds to account for variability. Parameter sharing is employed which allows for each agent to have a different behavior while sharing the same base network, and environment-specific adaptations, such as action selection probability adjustments for invalid actions, are implemented to ensure fairness and comparability across tests.

5.6 Tabular Results

In this section, we first present the tabular results for hyperbolic-weighted and hyperbolic-average methods, along with the baseline of exponential discounting. We then present Max Returns and Average Returns separately for Independent Learning and CTDE methods.

Table 5.2: Average returns and 95% confidence interval over five seeds for the three discounting policies across all tasks for IL methods.

Algs	Envs. /Disc. Policy	Exponential	Average	Hyperbolic
IA2C	Foraging-15x15-3p-5f-v2	0.32 ± 0.06*	0.33 ± 0.03*	**0.35 ± 0.05**
	Foraging-15x15-4p-3f-v2	0.72 ± 0.04*	**0.73 ± 0.04**	0.72 ± 0.03*
	Foraging-15x15-4p-5f-v2	0.36 ± 0.04	0.40 ± 0.06*	**0.45 ± 0.04**
	Simpleadversary-v0	13.68 ± 0.30*	**13.86 ± 0.29**	13.78 ± 0.28*
	Simplespeakerlistener-v0	-34.88 ± 6.71*	-32.27 ± 4.25*	**-30.87 ± 1.92**
	Simpletag-v0	405.60 ± 22.42	429.43 ± 24.97	**485.93 ± 21.78**
	Rware-small-4ag-v1	1.91 ± 0.40	1.76 ± 0.28	**2.45 ± 0.15**
	Rware-tiny-2ag-v1	1.92 ± 0.69*	1.75 ± 0.27*	**2.85 ± 1.02**
	Rware-tiny-4ag-v1	5.11 ± 3.50*	5.68 ± 2.90*	**6.97 ± 0.71**
IPPO	Foraging-15x15-3p-5f-v2	0.13 ± 0.07*	**0.20 ± 0.06**	0.17 ± 0.05*
	Foraging-15x15-4p-3f-v2	0.58 ± 0.03	0.60 ± 0.03*	**0.65 ± 0.04**
	Foraging-15x15-4p-5f-v2	0.30 ± 0.09*	**0.35 ± 0.04**	0.25 ± 0.04
	Simpleadversary-v0	14.16 ± 0.28*	**14.33 ± 0.28**	14.21 ± 0.28*
	Simplespeakerlistener-v0	-37.05 ± 7.86*	-33.78 ± 6.66*	**-29.78 ± 1.98**
	Simpletag-v0	446.13 ± 19.62	510.58 ± 37.86*	**516.53 ± 24.54**
	Rware-small-4ag-v1	**6.77 ± 3.86**	4.43 ± 1.76*	5.17 ± 1.91*
	Rware-tiny-2ag-v1	11.23 ± 4.75*	10.44 ± 4.43*	**12.08 ± 4.66**
	Rware-tiny-4ag-v1	20.96 ± 14.08*	23.71 ± 6.22*	**25.46 ± 6.52**
IQL	Foraging-15x15-3p-5f-v2	0.04 ± 0.01*	**0.06 ± 0.01**	0.05 ± 0.01*
	Foraging-15x15-4p-3f-v2	0.12 ± 0.03*	0.17 ± 0.02*	**0.17 ± 0.06**
	Foraging-15x15-4p-5f-v2	0.08 ± 0.01	**0.11 ± 0.01**	0.10 ± 0.01*
	Simpleadversary-v0	4.14 ± 1.27*	5.11 ± 1.02*	**5.75 ± 1.24**
	Simplespeakerlistener-v0	-50.38 ± 8.32*	-50.23 ± 8.13*	**-49.67 ± 7.58**
	Simpletag-v0	318.72 ± 17.06	296.17 ± 24.84	**362.06 ± 15.69**
	Rware-small-4ag-v1	**0.11 ± 0.11**	0.01 ± 0.01*	0.04 ± 0.04*
	Rware-tiny-2ag-v1	**0.03 ± 0.05**	0.01 ± 0.01*	0.01 ± 0.01*
	Rware-tiny-4ag-v1	0.29 ± 0.17*	0.09 ± 0.04	**0.29 ± 0.10**

Table 5.3: Average returns and 95% confidence interval over five seeds for the three discounting policies across all tasks for CTDE methods.

Algs	Envs. /Disc. Policy	Exponential	Average	Hyperbolic
MAA2C	Foraging-15x15-3p-5f-v2	0.27 ± 0.05	0.38 ± 0.03*	**0.38 ± 0.03**
	Foraging-15x15-4p-3f-v2	0.65 ± 0.03	**0.72 ± 0.04**	0.70 ± 0.02*
	Foraging-15x15-4p-5f-v2	0.36 ± 0.05	**0.49 ± 0.04**	0.49 ± 0.04*
	Simpleadversary-v0	14.44 ± 0.27*	**14.67 ± 0.28**	14.50 ± 0.28*
	Simplespeakerlistener-v0	-29.02 ± 1.71*	**-28.61 ± 1.17**	-28.82 ± 1.27*
	Simpletag-v0	**452.36 ± 15.49**	385.21 ± 197.41*	434.42 ± 42.02*
	Rware-small-4ag-v1	1.23 ± 0.89	1.95 ± 0.41	**2.71 ± 0.19**
	Rware-tiny-2ag-v1	1.62 ± 0.47	1.56 ± 0.32	**2.89 ± 0.47**
	Rware-tiny-4ag-v1	5.82 ± 3.53*	5.65 ± 3.60*	**7.55 ± 3.74**
MAPPO	Foraging-15x15-3p-5f-v2	0.15 ± 0.06*	**0.19 ± 0.04**	0.11 ± 0.05
	Foraging-15x15-4p-3f-v2	0.48 ± 0.04	0.57 ± 0.04*	**0.61 ± 0.04**
	Foraging-15x15-4p-5f-v2	0.26 ± 0.06*	**0.31 ± 0.04**	0.23 ± 0.06
	Simpleadversary-v0	13.36 ± 0.27*	**13.61 ± 0.26**	13.53 ± 0.27*
	Simplespeakerlistener-v0	-29.43 ± 2.97*	-28.39 ± 1.71*	**-28.24 ± 1.30**
	Simpletag-v0	426.19 ± 24.48	472.18 ± 15.47*	**481.18 ± 20.32**
	Rware-small-4ag-v1	**17.82 ± 1.04**	16.97 ± 1.00*	15.30 ± 0.85
	Rware-tiny-2ag-v1	13.96 ± 3.50*	16.99 ± 1.18*	**17.03 ± 1.17**
	Rware-tiny-4ag-v1	40.41 ± 2.82*	24.94 ± 19.71*	**41.25 ± 2.11**
QMIX	Foraging-15x15-3p-5f-v2	0.05 ± 0.02	**0.08 ± 0.01**	0.07 ± 0.02*
	Foraging-15x15-4p-3f-v2	0.06 ± 0.01	**0.17 ± 0.07**	0.13 ± 0.06*
	Foraging-15x15-4p-5f-v2	0.08 ± 0.04*	0.11 ± 0.02*	**0.11 ± 0.03**
	Simpleadversary-v0	10.52 ± 1.09*	**10.63 ± 1.46**	10.45 ± 1.27*
	Simplespeakerlistener-v0	-37.60 ± 5.48*	-42.29 ± 7.68*	**-37.12 ± 5.12**
	Simpletag-v0	368.15 ± 10.75*	346.31 ± 18.55	**383.42 ± 10.88**
	Rware-small-4ag-v1	**0.02 ± 0.01**	0.00 ± 0.00*	0.00 ± 0.01*
	Rware-tiny-2ag-v1	**0.02 ± 0.01**	0.01 ± 0.01*	0.01 ± 0.02*
	Rware-tiny-4ag-v1	**0.20 ± 0.13**	0.06 ± 0.03*	0.16 ± 0.18*
	Smac-2s-vs-1sc	**14.51 ± 1.58**	10.04 ± 2.36	14.39 ± 1.35*
	Smac-3s5z	13.41 ± 0.94	13.81 ± 1.06	**15.66 ± 0.74**
	Smac-mmm2	9.96 ± 0.69*	7.30 ± 0.86	**10.32 ± 0.83**

Table 5.4: Maximum returns and 95% confidence interval over five seeds for the three discounting policies across all tasks for IL Methods.

Algs	Envs. /Disc. Policy	Exponential	Average	Hyperbolic
IA2C	Foraging-15x15-3p-5f-v2	0.44 ± 0.03*	0.43 ± 0.05*	**0.48 ± 0.01**
	Foraging-15x15-4p-3f-v2	**0.85 ± 0.03**	0.84 ± 0.02*	0.84 ± 0.01*
	Foraging-15x15-4p-5f-v2	0.55 ± 0.04*	0.55 ± 0.03*	**0.60 ± 0.05**
	Simpleadversary-v0	15.23 ± 0.16*	**15.36 ± 0.22**	15.25 ± 0.24*
	Simplespeakerlistener-v0	-27.04 ± 6.01*	**-23.51 ± 1.41**	-23.80 ± 1.38*
	Simpletag-v0	443.64 ± 17.92	502.55 ± 18.65	**567.29 ± 34.67**
	Rware-small-4ag-v1	3.98 ± 0.44	4.06 ± 0.42*	**4.58 ± 0.24**
	Rware-tiny-2ag-v1	5.53 ± 3.14*	4.70 ± 1.16*	**6.64 ± 2.99**
	Rware-tiny-4ag-v1	10.33 ± 7.40*	11.56 ± 6.46*	**13.02 ± 1.67**
IPPO	Foraging-15x15-3p-5f-v2	0.23 ± 0.17*	0.41 ± 0.05*	**0.41 ± 0.03**
	Foraging-15x15-4p-3f-v2	0.68 ± 0.02	0.74 ± 0.03	**0.83 ± 0.01**
	Foraging-15x15-4p-5f-v2	0.45 ± 0.13*	0.54 ± 0.04*	**0.56 ± 0.03**
	Simpleadversary-v0	15.41 ± 0.25*	**15.50 ± 0.19**	15.47 ± 0.26*
	Simplespeakerlistener-v0	-30.80 ± 7.73*	-26.51 ± 3.10*	**-23.90 ± 1.46**
	Simpletag-v0	495.76 ± 34.47	**624.45 ± 25.76**	611.43 ± 24.86*
	Rware-small-4ag-v1	**16.04 ± 4.52**	8.90 ± 3.73	10.60 ± 4.55*
	Rware-tiny-2ag-v1	**18.36 ± 5.88**	15.53 ± 6.18*	18.31 ± 6.89*
	Rware-tiny-4ag-v1	28.98 ± 18.92*	**33.22 ± 8.71**	32.90 ± 7.67*
IQL	Foraging-15x15-3p-5f-v2	0.05 ± 0.02	**0.08 ± 0.01**	0.06 ± 0.02*
	Foraging-15x15-4p-3f-v2	0.20 ± 0.09*	0.28 ± 0.05*	**0.32 ± 0.12**
	Foraging-15x15-4p-5f-v2	0.11 ± 0.01	**0.14 ± 0.01**	0.13 ± 0.02*
	Simpleadversary-v0	7.74 ± 0.48	**9.20 ± 0.22**	9.15 ± 0.49*
	Simplespeakerlistener-v0	-37.07 ± 7.67*	-35.48 ± 6.67*	**-35.45 ± 6.48**
	Simpletag-v0	375.73 ± 14.91	355.77 ± 39.68	**413.09 ± 13.24**
	Rware-small-4ag-v1	**0.42 ± 0.47**	0.03 ± 0.02*	0.15 ± 0.18*
	Rware-tiny-2ag-v1	**0.12 ± 0.22**	0.02 ± 0.01*	0.03 ± 0.02*
	Rware-tiny-4ag-v1	0.75 ± 0.64*	0.25 ± 0.17	**1.20 ± 0.60**

Table 5.5: Maximum returns and 95% confidence interval over five seeds for the three discounting policies across all tasks for CTDE Methods.

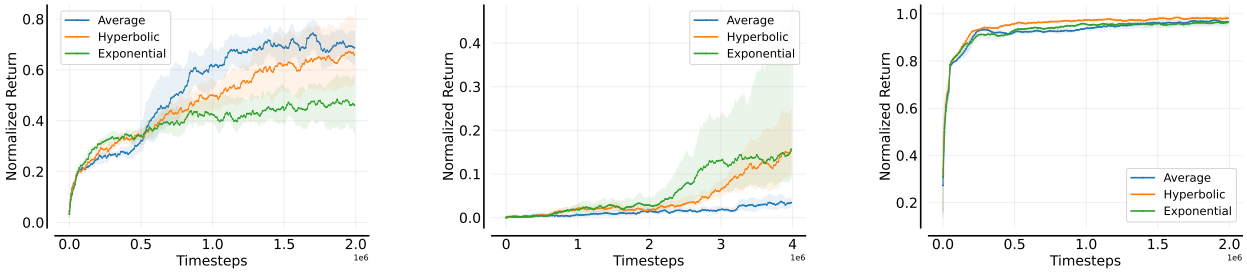
Algs	Envs. /Disc. Policy	Exponential	Average	Hyperbolic
MAA2C	Foraging-15x15-3p-5f-v2	0.44 ± 0.04	0.47 ± 0.02*	**0.50 ± 0.02**
	Foraging-15x15-4p-3f-v2	0.74 ± 0.04	**0.84 ± 0.02**	0.82 ± 0.01*
	Foraging-15x15-4p-5f-v2	0.52 ± 0.02	0.59 ± 0.05*	**0.61 ± 0.04**
	Simpleadversary-v0	15.75 ± 0.24*	**16.09 ± 0.22**	15.91 ± 0.24*
	Simplespeakerlistener-v0	**-23.13 ± 1.30**	-23.21 ± 1.35*	-23.43 ± 1.31*
	Simpletag-v0	502.37 ± 32.86*	453.35 ± 271.68*	**523.43 ± 40.35**
	Rware-small-4ag-v1	3.56 ± 0.17	5.73 ± 0.86*	**6.65 ± 0.50**
	Rware-tiny-2ag-v1	3.78 ± 1.44	3.57 ± 0.48	**8.07 ± 3.03**
	Rware-tiny-4ag-v1	13.46 ± 5.23*	11.17 ± 7.67*	**18.27 ± 9.78**
MAPPO	Foraging-15x15-3p-5f-v2	0.37 ± 0.05*	**0.40 ± 0.02**	0.30 ± 0.12*
	Foraging-15x15-4p-3f-v2	0.64 ± 0.05	0.70 ± 0.03	**0.81 ± 0.01**
	Foraging-15x15-4p-5f-v2	0.50 ± 0.02*	**0.52 ± 0.03**	0.51 ± 0.04*
	Simpleadversary-v0	14.52 ± 0.28*	**14.75 ± 0.23**	14.66 ± 0.22*
	Simplespeakerlistener-v0	**-23.14 ± 1.36**	-23.23 ± 1.31*	-23.35 ± 1.35*
	Simpletag-v0	470.61 ± 24.88	**552.06 ± 19.96**	541.54 ± 14.49*
	Rware-small-4ag-v1	26.20 ± 0.76*	**26.86 ± 0.71**	25.05 ± 0.52
	Rware-tiny-2ag-v1	18.77 ± 4.26*	22.06 ± 1.20*	**22.13 ± 1.37**
	Rware-tiny-4ag-v1	50.71 ± 1.97*	31.28 ± 24.69*	**51.50 ± 2.50**
QMIX	Foraging-15x15-3p-5f-v2	0.08 ± 0.04	**0.14 ± 0.02**	0.11 ± 0.02*
	Foraging-15x15-4p-3f-v2	0.08 ± 0.03	**0.51 ± 0.22**	0.37 ± 0.20*
	Foraging-15x15-4p-5f-v2	0.15 ± 0.06*	0.20 ± 0.01*	**0.21 ± 0.10**
	Simpleadversary-v0	**14.42 ± 0.15**	14.27 ± 0.72*	14.17 ± 0.65*
	Simplespeakerlistener-v0	-24.15 ± 1.47*	-25.51 ± 1.95*	**-24.00 ± 1.59**
	Simpletag-v0	**511.80 ± 13.05**	435.48 ± 6.02	509.49 ± 12.30*
	Rware-small-4ag-v1	**0.05 ± 0.04**	0.00 ± 0.01*	0.01 ± 0.02*
	Rware-tiny-2ag-v1	0.04 ± 0.02*	0.01 ± 0.02*	**0.05 ± 0.05**
	Rware-tiny-4ag-v1	0.62 ± 0.52*	0.12 ± 0.07*	**0.65 ± 0.86**
	Smac-2s-vs-1sc	17.81 ± 0.31*	13.21 ± 1.69	**17.97 ± 0.28**
	Smac-3s5z	15.30 ± 0.86	15.21 ± 1.83	**18.10 ± 0.30**
	Smac-mmm2	11.19 ± 0.34*	8.35 ± 0.32	**11.86 ± 1.29**

5.7 Method-Wise Results

We now present results for the two proposed discounting methods along with the baseline for each of the 6 methods, comparing the performance of each method across benchmarks. As a general trend, the performance difference is visible in LBF and RWARE (and for QMIX in SMAC), with one of the two proposed variants performing better. However, the performance difference is minimal in MPE.

5.7.1 IQL

Figure 5.4 shows the performance of the proposed hyperbolic-weighted and hyperbolic-average discounting methods for IQL.



(a) LBF (b) RWARE (c) MPE

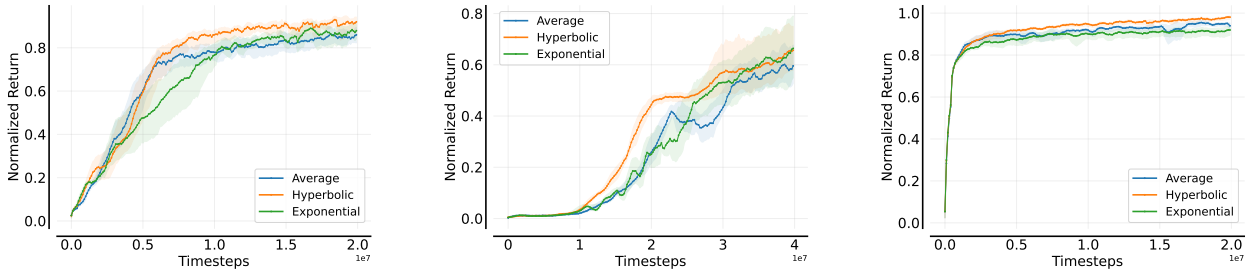
Figure 5.4: Aggregate performance of the three discounting methods implemented in IQL on the cooperative MARL benchmarks studied.

5.7.2 IA2C

Figure 5.5 shows the performance of the proposed hyperbolic-weighted and hyperbolic-average discounting methods for IA2C.

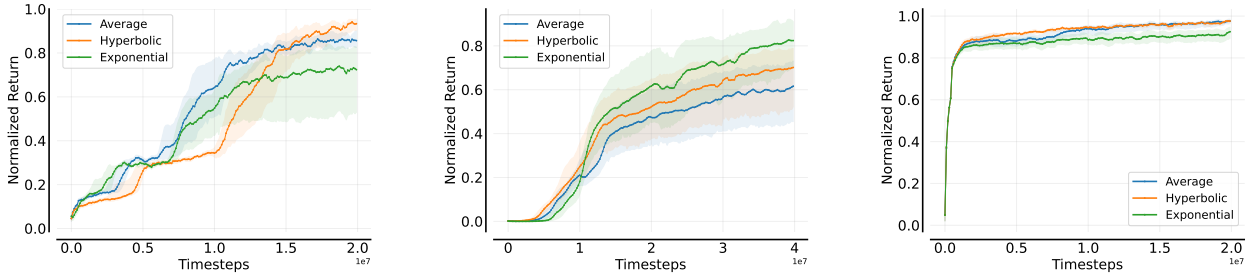
5.7.3 IPPO

Figure 5.6 shows the performance of the proposed hyperbolic-weighted and hyperbolic-average discounting methods for IPPO.



(a) LBF (b) RWARE (c) MPE

Figure 5.5: Aggregate performance of the three discounting methods implemented in IA2C on the cooperative MARL benchmarks studied.

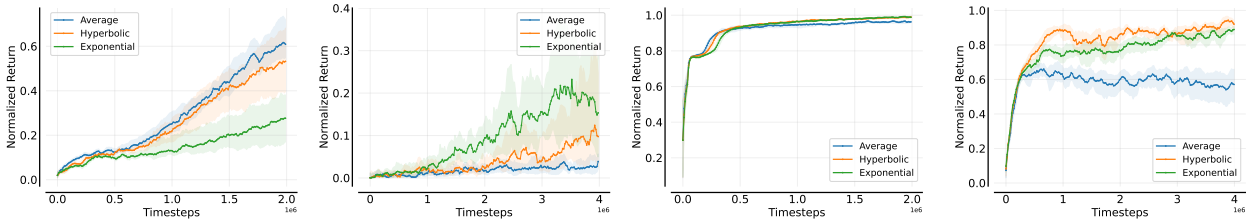


(a) LBF (b) RWARE (c) MPE

Figure 5.6: Aggregate performance of the three discounting methods implemented in IPPO on the cooperative MARL benchmarks studied.

5.7.4 QMIX

Figure 5.7 shows the performance of the proposed hyperbolic-weighted and hyperbolic-average discounting methods for QMIX.

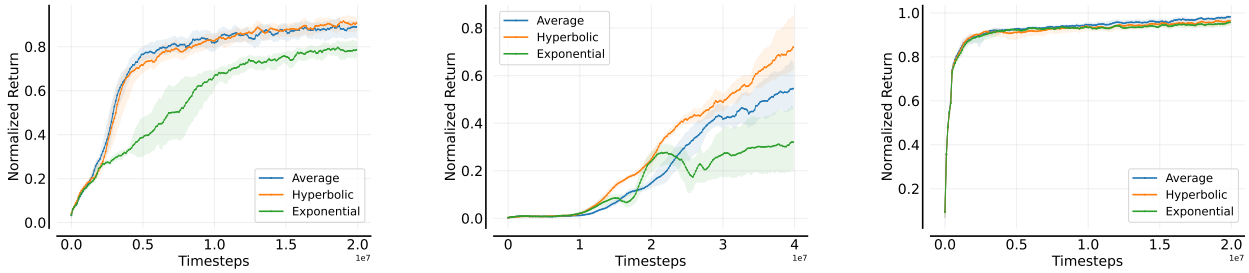


(a) LBF (b) RWARE (c) MPE (d) SMAC

Figure 5.7: Aggregate performance of the three discounting methods implemented in QMIX on the cooperative MARL benchmarks studied.

5.7.5 MAA2C

Figure 5.8 shows the performance of the proposed hyperbolic-weighted and hyperbolic-average discounting methods for MAA2C.

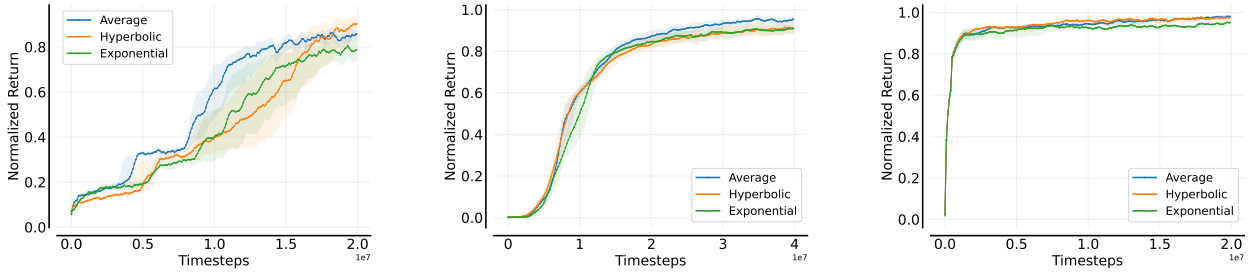


(a) LBF (b) RWARE (c) MPE

Figure 5.8: Aggregate performance of the three discounting methods implemented in MAA2C on the cooperative MARL benchmarks studied.

5.7.6 MAPPO

Figure 5.9 shows the performance of the proposed hyperbolic-weighted and hyperbolic-average discounting methods for MAPPO.



(a) LBF (b) RWARE (c) MPE

Figure 5.9: Aggregate performance of the three discounting methods implemented in MAPPO on the cooperative MARL benchmarks studied.

5.8 Environment-Wise Results

5.8.1 LBF

Figure 5.10 presents a comparison of the proposed hyperbolic discounting and average discounting formulations against the baseline of exponential discounting for the 6 methods on the set of three environments from the LBF (Papoudakis et al., 2021) benchmark.

5.8.2 RWARE

Figure 5.11 presents a comparison of the proposed hyperbolic discounting and average discounting formulations against the baseline of exponential discounting for the 6 methods on the set of three environments from the Robot Warehouse (RWARE) (Papoudakis et al., 2021) benchmark.

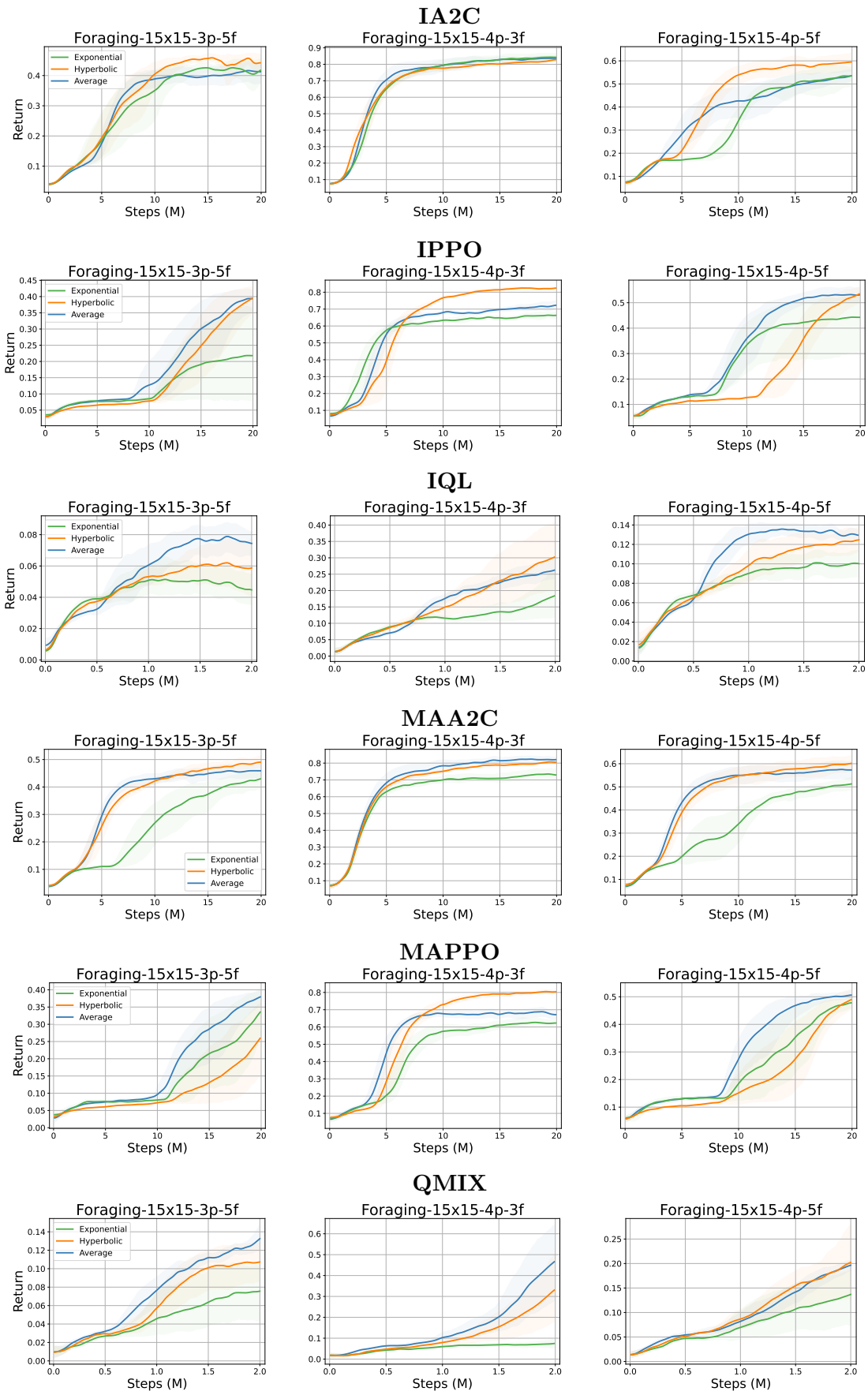


Figure 5.10: Individual results of each of the 3 discounting policies on all 6 methods in 3 LBF environments. The hyperbolic variants (blue and orange) perform better than exponential discounting (green) in most of the environments.

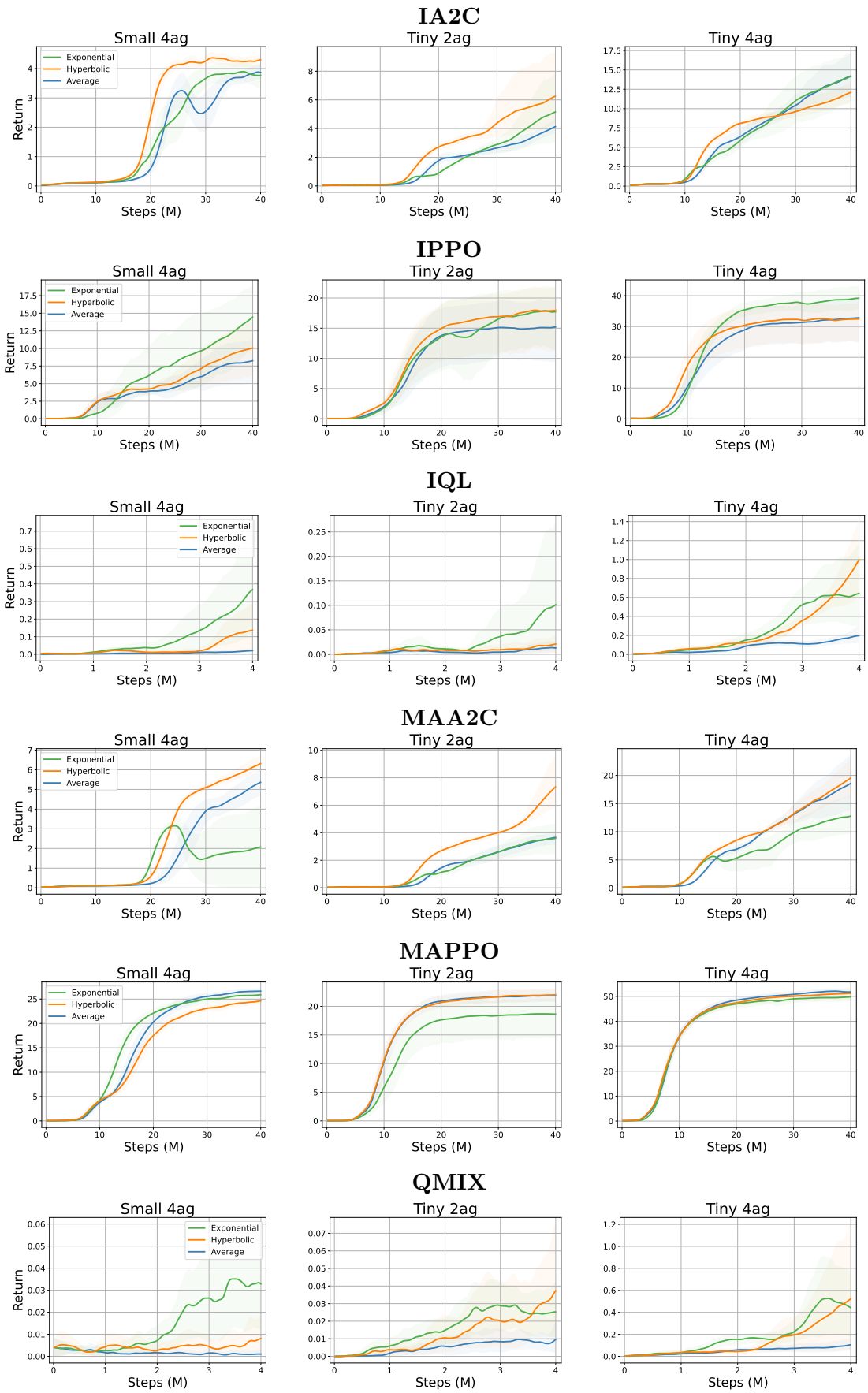


Figure 5.11: Individual results of each of the 3 discounting policies on all 6 methods in 3 RWARE environments.

5.8.3 MPE

Figure 5.12 presents a comparison of the proposed hyperbolic discounting and average discounting formulations against the baseline of exponential discounting for the 6 methods on the set of three environments from the Multi Particle Environment (MPE) (Lowe et al., 2017) benchmark.

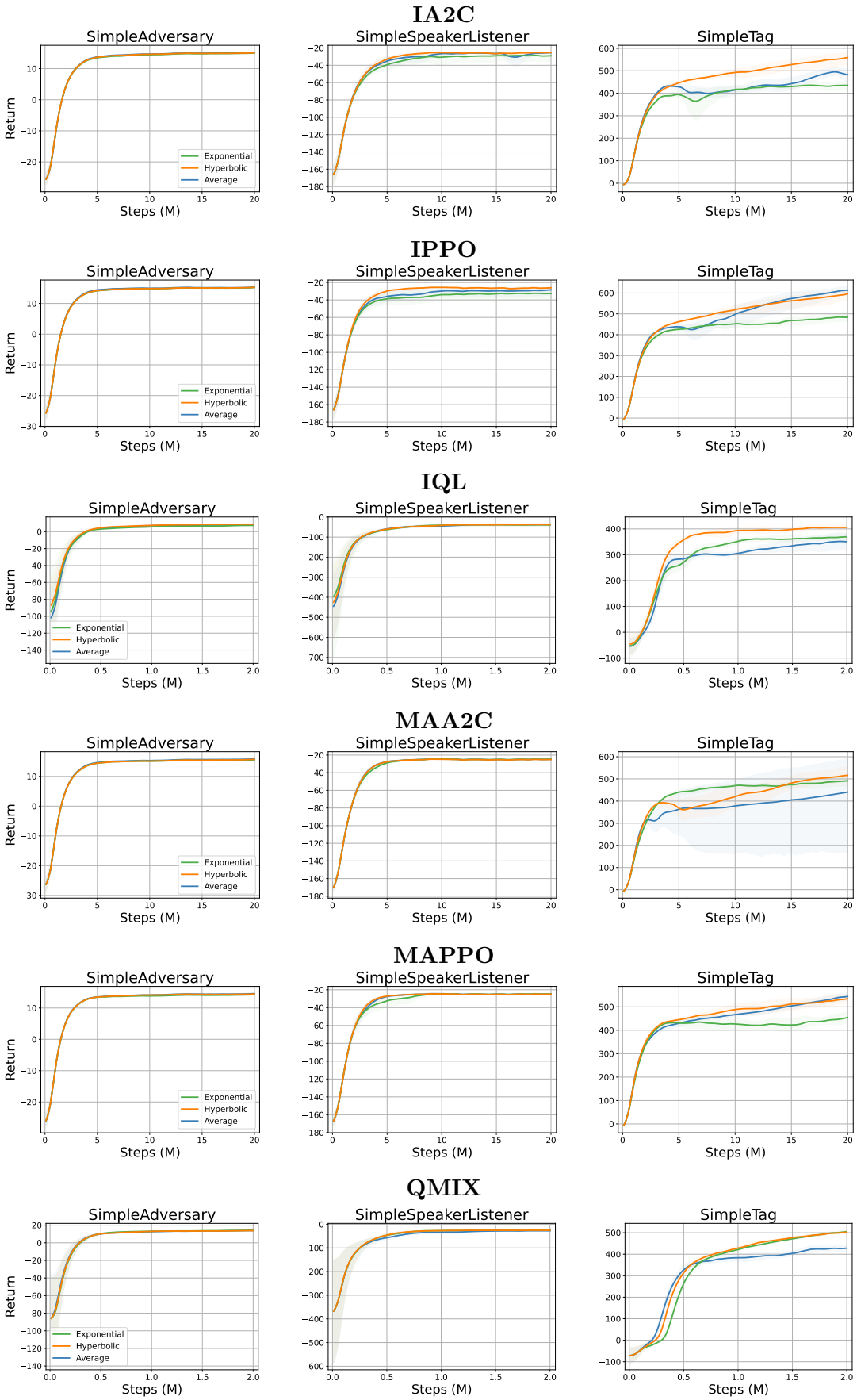


Figure 5.12: Individual results of each of the 3 discounting policies on all 6 methods in 3 MPE environments. There is no notable performance difference between discounting methods except in the SimpleTag environment.

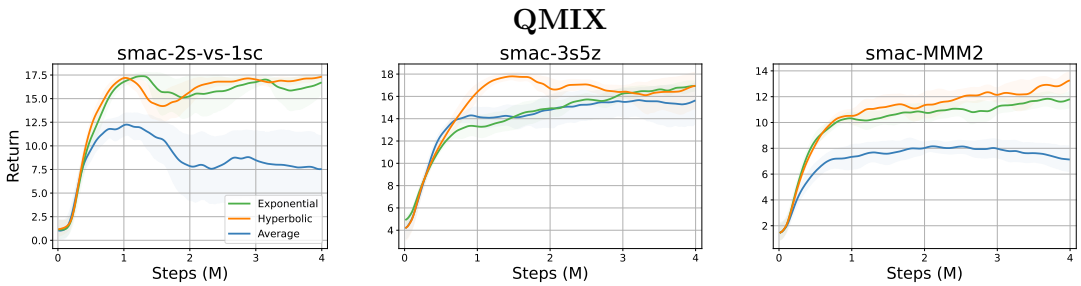


Figure 5.13: Individual results of each of the 3 discounting policies on only QMIX for 3 SMAC maps.

5.8.4 SMAC

Figure 5.13 presents a comparison of the proposed hyperbolic discounting and average discounting formulations against the baseline of exponential discounting for the QMIX method on a set of three maps from the StarCraft MultiAgent Challenge (Samvelyan et al., 2019) benchmark. Due to the increased runtimes of SMAC, we only present results here for QMIX, which is the most widely studied MARL algorithm, especially in the SMAC environment.

5.9 Conclusion

These experiments revealed significant improvements in performance, stability, and sample efficiency. While the impact varied across the algorithms and environments tested, non-exponential discounting methods consistently outperformed the traditional exponential discounting method. The averaged horizon discounting method emerged as the most reliable choice in terms of stability and real-world applications, as it consistently demonstrated smaller standard deviations and improved performance in several algorithms. The study also suggests that structural differences in algorithms may affect the impact of non-exponential discounting methods, with some algorithms benefiting more than others. Future studies can explore the use of ensemble methods to further enhance the incorporation of multi-horizon discounting functions. Overall, the findings highlight the potential benefits of non-exponential discounting methods in reinforcement learning, which can lead to more efficient and effective decision-making processes in real-world scenarios.

Our results not only advocate for the adoption of hyperbolic discounting in MARL settings but also pave the way for future research into preference-based reinforcement learning. By aligning AI discounting behaviors more closely with human cognitive processes, this work promises to enhance human-AI cooperation, foster greater societal trust in AI technologies, and contribute to the ethical development of AI systems. Through our approach, we invite further exploration into how preference-based modeling can inform the design and implementation of multi-agent systems, with the potential to revolutionize our interaction with AI in complex decision-making scenarios.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This dissertation explored the integration of non-exponential discounting functions, particularly hyperbolic discounting, into deep reinforcement learning (RL) frameworks. The overarching goal was to develop RL agents that can better model and mimic human-like decision-making processes, which often exhibit preference reversals and inconsistent choices over time. By incorporating hyperbolic discounting, the aim was to enhance the agents' ability to make decisions involving intertemporal trade-offs, aligning their behavior more closely with empirical observations from psychology and behavioral economics.

The first contribution of this dissertation revisited the idea of hyperbolic discounting and multi-horizon learning in the context of off-policy, value-based RL methods. Through extensive experiments on procedurally generated environments, such as Procgen and Crafter, the impact of hyperbolic discounting and learning over multiple horizons was evaluated across various performance dimensions, including sample complexity, generalization, and architectural improvements. The results demonstrated that agents employing hyperbolic discounting and multi-horizon learning generally outperformed their single-horizon counterparts, exhibiting improved generalization capabilities and robustness in diverse and challenging environments.

The second contribution introduced a novel two-parameter hyperboloid model, which extended the Rachlin hyperbolic discounting framework by incorporating a sensitivity-to-delay parameter alongside the base hyperbolic discount factor. This model aimed to capture individual variations in the perception of reward delays, a crucial aspect of human decision-making that is often overlooked in traditional RL approaches. The empirical evaluation of the proposed model revealed its potential for enhancing temporal decision-making in RL agents, paving the way for more nuanced and human-like behavior.

This dissertation's third and most significant contribution was exploring non-exponential discounting in multi-agent reinforcement learning (MARL) settings. By introducing hyperbolic discounting and averaged horizon discounting techniques, this work investigated the impact of these discounting methods on agent decision-making and collective performance in cooperative and competitive scenarios. A comprehensive analysis was conducted across various MARL algorithms, spanning different classes and architectures, including Independent Learning (IL), Centralized Training with Decentralized Execution (CTDE), Value Factorization, and Centralized Policy Gradient methods. The results demonstrated that the benefits of non-exponential discounting were evident across multiple learning modalities, with particularly pronounced enhancements observed in the CTDE framework. The findings also highlighted the potential of hyperbolic discounting to improve cooperation among agents and facilitate more effective coordination in multi-agent tasks.

Overall, this dissertation contributes to the growing body of research aimed at bridging the gap between traditional exponential discounting methods and non-exponential forms of discounting. This work paves the way for more intelligent, adaptable, and human-centric AI systems by incorporating hyperbolic discounting and related discount functions into deep RL frameworks. The findings highlight the potential of non-exponential discounting to enhance agent performance, facilitate human-like decision-making, and foster improved cooperation and coordination in multi-agent scenarios.

6.2 Future Work

While this dissertation has made significant strides in integrating non-exponential discounting into deep RL frameworks, several avenues for future research remain unexplored. Here are some potential directions for further investigation:

6.2.1 Other Non-Exponential Discounting Functions

Although this dissertation focused primarily on hyperbolic discounting, other non-exponential discounting functions, such as quasi-hyperbolic discounting or generalized hyperbolic discounting, could be explored. These alternative functions may capture different aspects of human decision-making behavior and could potentially lead to further improvements in agent performance and human likeness.

6.2.2 Adaptive and Dynamic Discounting

The current work employed fixed discounting parameters (e.g., the hyperbolic discount factor k and the sensitivity-to-delay parameter σ) across all environments and tasks. However, human decision-making is known to be context-dependent, with individuals exhibiting varying degrees of patience and risk aversion based on the specific situation. Future research could investigate adaptive or dynamic discounting mechanisms that can adjust the discounting parameters based on the agent's experiences, environmental cues, or task characteristics.

6.2.3 Survival Analysis

One promising avenue for future research involves leveraging survival analysis to bridge the gap between reward discounting and the underlying hazard rate of the environment. By developing a learning representation that incorporates the empirical hazard rate over time, we can enable agents to adapt their discounting strategies accordingly. Human decision-making is profoundly influenced by the perceived level of hazard in a given situation, be it a wartime scenario or a terminal medical diagnosis. Consequently, our plans and preferences can shift

drastically based on our perceived life expectancy. Prior work (Fedus et al., 2019) has shown that a known, constant hazard rate implies the suitability of exponential discounting, while an unknown or variable hazard rate necessitates the adoption of non-exponential discounting functions, such as hyperbolic discounting. Hyperbolic discounting, with its robustness in scenarios of uncertain hazard, can facilitate learning over multiple horizons. However, if the agent can accurately approximate the environment's underlying hazard rate, it can equivalently select a single, hazard-appropriate exponential discount factor. The overarching goal of this research direction is to harness survival analysis techniques, including statistical methods like Kaplan-Meier and Cox regression and machine learning approaches, to estimate the hazard rate. Future work in this domain may explore the integration of other non-exponential discounting functions, with a particular emphasis on studying priors for the gamma distribution of the hazard rate, such as Erlang distributions. Additionally, elucidating practical applications where certain discounting functions may be undesirable could further refine our understanding of this intricate relationship between discounting and hazard rates.

6.2.4 Multi-Agent Coordination

While this dissertation explored the impact of non-exponential discounting on agent decision-making and performance in multi-agent settings, the focus was primarily on cooperative and competitive scenarios. Future work could delve deeper into the implications of non-exponential discounting for multi-agent coordination, particularly in scenarios involving complex communication protocols, dynamic team formations, or hierarchical decision-making structures. Another avenue of future work could be to study the effects of discounting in competitive, zero-sum, and general-sum games.

6.2.5 Theoretical Foundations and Convergence Analysis

There exists a body of work that disagrees with hyperbolic discounting accurately capturing the discounting model in humans (Sopher and Sheth, 2006). However, a predominant

part of research in psychology suggests that observed empirical data does not fit exponential function, and is more hyperbolic in nature. To circumvent the issue of whether humans discount hyperbolically or not, one can study non-exponential discounting on its own merits. Although the empirical results presented in this dissertation demonstrate the potential benefits of non-exponential discounting, a more rigorous theoretical analysis of the proposed methods' convergence properties and optimality guarantees is warranted. Such theoretical foundations could provide valuable insights into the conditions under which non-exponential discounting is advantageous and could guide the development of more principled algorithms.

6.2.6 Integration with Other RL Advancements

This dissertation focused on the integration of non-exponential discounting into existing deep RL frameworks. However, the field of reinforcement learning is rapidly evolving, with new advancements in areas such as meta-learning, hierarchical reinforcement learning, and multi-task learning. Future research could explore the synergies between non-exponential discounting and these emerging techniques, potentially leading to more powerful and versatile RL agents.

6.2.7 Real-World Applications and Deployment

While this dissertation primarily focused on evaluating the proposed methods in simulated environments, the ultimate goal is to develop AI systems that can be deployed in real-world scenarios. Future work could involve the application of non-exponential discounting techniques to real-world problems, such as robotics, autonomous systems, or decision support systems in domains where human-like decision-making is crucial. The application of discounting in Large Language Models would be an interesting experiment.

6.2.8 Interpretability and Explainability

As AI systems become more complex and integrated into decision-making processes, there is a growing need for interpretability and explainability. Future research could explore techniques

for interpreting and explaining the decision-making processes of RL agents that employ non-exponential discounting, potentially enhancing trust and transparency in these systems.

6.2.9 Ethical Considerations and Societal Impact

The development of AI systems that can mimic human-like decision-making raises important ethical considerations and potential societal impacts. Future work should carefully examine the ethical implications of non-exponential discounting in RL agents, particularly in domains where decisions can have significant consequences, such as healthcare, finance, or public policy.

By addressing these future research directions, the field of reinforcement learning can continue to advance toward the development of more intelligent, adaptable, and human-centric AI systems capable of making decisions that align with human preferences and values.

Bibliography

Rishab Agarwal. rliable. https://github.com/google-research/rliable, 2021. 41

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *NeurIPS*, 2021. 40

George Ainslie. Specious reward: A behavioral theory of impulsiveness and impulse control. *Psychological Bulletin*, 82(4):463–496, 1975. doi: 10.1037/h0076860. 26, 51, 66

Matthew Aitchison, Penny Sweetser, and Marcus Hutter. Atari-5: Distilling the arcade learning environment down to five games. In *International Conference on Machine Learning*, pages 421–438. PMLR, 2023. 41, 43, 51, 55, 56

Stefano V Albrecht and Subramanian Ramamoorthy. A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems. In *AAMAS'13 Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1155–1156. International Foundation for Autonomous Agents and Multiagent Systems, 2013. 79

William H Alexander and Joshua W Brown. Hyperbolically discounted temporal difference learning. *Neural computation*, 22(6):1511–1527, 2010. 66

Raja Farrukh Ali. Non-exponential reward discounting in reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(13):16111–16112, 2023. 4

Raja Farrukh Ali, Nasik Muhammad Nafi, Kevin Duong, and William Hsu. Efficient multi-horizon learning for off-policy reinforcement learning. In *NeurIPS Deep Reinforcement Learning Workshop*, 2022. 5, 76

Raja Farrukh Ali, Kevin Duong, Nasik Muhammad Nafi, and William Hsu. Multi-horizon learning in procedurally-generated environments for off-policy reinforcement learning (student abstract). *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(13): 16150–16151, 2023a. 5, 66

Raja Farrukh Ali, Ayesha Farooq, Emmanuel Adeniji, John Woods, Vinny Sun, and William Hsu. Explainable reinforcement learning for alzheimer's disease progression prediction. In *NeurIPS XAI in Action Workshop*, 2023b. 5

Devansh Arpit, Huan Wang, Yingbo Zhou, and Caiming Xiong. Ensemble of averages: Improving model selection and boosting performance in domain generalization. *Advances in Neural Information Processing Systems*, 35:8265–8277, 2022. 67, 68

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. 31, 35, 41, 55

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pages 449–458. PMLR, 2017. 37

Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1957a. 15, 53, 63

Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957b. 12, 49

Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002. 14

Dimitri P Bertsekas and John N Tsitsiklis. Neuro-dynamic programming: an overview. In *Proceedings of 34th IEEE conference on decision and control*. IEEE, 1995. 34

Michael T Bixter and Christian C Luhmann. The social contagion of temporal discounting in small social networks. *Cognitive Research: Principles and Implications*, 6(1):13, 2021. 62

Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G. Bellemare. Dopamine: A Research Framework for Deep Reinforcement Learning, 2018. URL http://arxiv.org/abs/1812.06110. 39

Victor Churchill, Steve Manns, Zhen Chen, and Dongbin Xiu. Robust modeling of unknown dynamical systems via ensemble averaged learning. *Journal of Computational Physics*, 474:111842, 2023. 67, 68

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *ICML*, pages 2048–2056. PMLR, 2020. 27, 31, 35, 39, 41, 51, 55

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989. 10

Christian Schroeder De Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020. 76

Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009. 9

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, pages 1407–1416. PMLR, 2018. 37

William Fedus, Carles Gelada, Yoshua Bengio, Marc G Bellemare, and Hugo Larochelle. Hyperbolic discounting and learning over multiple horizons. *arXiv preprint arXiv:1902.06865*,

2019. 3, 25, 33, 35, 36, 37, 38, 39, 41, 42, 43, 47, 50, 52, 53, 62, 63, 65, 66, 67, 70, 72, 76, 101

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. *Proceedings of the AAAI conference on artificial intelligence*, 32(1), 2018. 23

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *Proceedings of the International Conference on Representation Learning (ICLR)*, 2018. 35, 37

Christopher T Franck, Haily K Traxler, Brent A Kaplan, Mikhail N Koffarnus, and Mark J Rzeszutek. A tribute to howard rachlin and his two-parameter discounting model: Reliable and flexible model fitting. *Journal of the Experimental Analysis of Behavior*, 119(1):156–168, 2023. 49

Vincent François-Lavet, Raphael Fonteneau, and Damien Ernst. How to discount deep reinforcement learning: Towards new dynamic strategies. *arXiv preprint arXiv:1512.02011*, 2015. 62

Leonard Green and Joel Myerson. A discounting framework for choice with delayed and probabilistic rewards. *Psychological bulletin*, 130(5):769, 2004. 49

Leonard Green, Nathanael Fristoe, and Joel Myerson. Temporal discounting and preference reversals in choice between delayed outcomes. *Psychonomic Bulletin & Review*, 1(3):383–389, 1994. 36, 52

Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers 16*, pages 66–83. Springer, 2017. 70

Danijar Hafner. Benchmarking the spectrum of agent capabilities. *arXiv preprint arXiv:2109.06780*, 2021. 27, 31, 35, 41, 51, 55

Eric A Hansen, Daniel S Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, pages 709–715, 2004. 14

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 37

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI conference on artificial intelligence*, 2018. 27, 35, 37, 38, 39, 51

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 9

Hao Hu, Yiqin Yang, Qianchuan Zhao, and Chongjie Zhang. On the role of discount factor in offline reinforcement learning. In *International Conference on Machine Learning*, pages 9072–9098. PMLR, 2022. 25

Bernhard Jaeger and Andreas Geiger. An invitation to deep reinforcement learning. *arXiv preprint arXiv:2312.08365*, 2023. 7

Natasha Jaques. *Social and affective machine learning*. PhD thesis, Massachusetts Institute of Technology, 2019. 7

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. 12

Herman Kahn and Theodore E Harris. Estimation of particle transmission by random sampling. *National Bureau of Standards applied mathematics series*, 12:27–30, 1951. 20

Kris N. Kirby. Bidding on the future: Evidence against normative discounting of delayed rewards. *Journal of Experimental Psychology: General*, 126(1):54–70, 1997. 52

Daphne Koller and Ronald Parr. Computing factored value functions for policies in structured mdps. In *IJCAI*, volume 99, pages 1332–1339, 1999. 71

Zeb Kurth-Nelson and A David Redish. Temporal-difference reinforcement learning with distributed representations. *PLoS One*, 4(10):e7362, 2009. 36

Ariel Kwiatkowski, Eduardo Alvarado, Vicky Kalogeiton, C Karen Liu, Julien Pettré, Michiel van de Panne, and Marie-Paule Cani. A survey on reinforcement learning methods in character animation. *Computer Graphics Forum*, 41(2):613–639, 2022. 7

Ariel Kwiatkowski, Vicky Kalogeiton, Julien Pettré, and Marie-Paule Cani. Ugae: A novel approach to non-exponential discounting. *arXiv preprint arXiv:2302.05740*, 2023. 52

Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995. 9

Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994. 13, 63

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*, 2017. 23, 29, 32, 80, 94

Xueguang Lyu, Yuchen Xiao, Brett Daley, and Christopher Amato. Contrasting centralized and decentralized critics in multi-agent reinforcement learning. *arXiv preprint arXiv:2102.04402*, 2021. 78

Tiago V Maia. Reinforcement learning, conditioning, and the brain: Successes and challenges. *Cognitive, Affective, & Behavioral Neuroscience*, 9(4):343–364, 2009. 66

James E. Mazur. An adjusting procedure for studying delayed reinforcement. In *Quantitative analyses of behavior*, volume 5, pages 55–73. Lawrence Erlbaum Associates, Inc., 1987. 25, 49, 50, 51

James E Mazur. Choice, delay, probability, and conditioned reinforcement. *Animal Learning & Behavior*, 25(2):131–147, 1997. 36, 60

Todd L McKerchar, Leonard Green, Joel Myerson, T Stephen Pickford, Jade C Hill, and Steven C Stout. A comparison of four models of delay discounting in humans. *Behavioural processes*, 81(2):256–259, 2009. 49, 51, 52

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018. 35

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *NeurIPS Deep Learning Workshop*, 2013. 72

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015. 22, 27, 36

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016. 73

Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*, 2017. 32, 80

Joel Myerson and Leonard Green. Discounting of delayed rewards: Models of individual choice. *Journal of the experimental analysis of behavior*, 64(3):263–276, 1995. 25, 49, 52

Nasik Muhammad Nafi, Raja Farrukh Ali, and William Hsu. Analyzing the sensitivity to policy-value decoupling in deep reinforcement learning generalization. In *Deep Reinforcement Learning Workshop NeurIPS*, 2022a. 5

Nasik Muhammad Nafi, Raja Farrukh Ali, and William Hsu. Hyperbolically discounted advantage estimation for generalization in reinforcement learning. In *ICML Workshop on Decision Awareness in Reinforcement Learning*, 2022b. 5, 27, 52, 64, 66, 74

Nasik Muhammad Nafi, Raja Farrukh Ali, and William Hsu. Analyzing the sensitivity to policy-value decoupling in deep reinforcement learning generalization. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pages 2625–2627, 2023. 5

Guillermo Owen. *Game theory*. Academic Press, 1982. 13

Georgios Papoudakis, Filippos Christianos, Arrasy Rahman, and Stefano V Albrecht. Dealing with non-stationarity in multi-agent deep reinforcement learning. *arXiv preprint arXiv:1906.04737*, 2019. 80

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*, 2021. 31, 32, 75, 77, 79, 81, 82, 91

Howard Rachlin. *Judgment, Decision, and Choice: A Cognitive/Behavioral Synthesis*. W.H. Freeman and Company, 1989. 25, 49, 52

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4295–4304. PMLR, 2018. 29, 71, 77

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-

agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020. 29, 77

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986. 9

G. A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, University of Cambridge, Department of Engineering, 1994. 21

Paul A Samuelson. A note on measurement of utility. *The review of economic studies*, 4(2): 155–161, 1937. 51

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft Multi-Agent Challenge. *CoRR*, abs/1902.04043, 2019. 32, 80, 96

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *ICLR*, 2016. 37

Dominik Schmidt and Thomas Schmied. Fast and data-efficient training of rainbow: an experimental study on atari. *Deep RL Workshop NeurIPS*, 2021. URL https://arxiv.org/abs/2111.10247. 35, 38, 39

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 27, 73, 76, 78

Matthias Schultheis, Constantin A Rothkopf, and Heinz Koeppl. Reinforcement learning with non-exponential discounting. *Advances in neural information processing systems*, 35: 3649–3662, 2022. 62

Craig Sherstan, Shibhansh Dohare, James MacGlashan, Johannes Günther, and Patrick M

Pilarski. Gamma-nets: Generalizing value estimation over timescale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5717–5725, 2020. 36

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. Pmlr, 2014. 29

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018. 22

David Silver, Satinder Singh, Doina Precup, and Richard S Sutton. Reward is enough. *Artificial Intelligence*, 299:103535, 2021. 2, 23, 60

Travis R Smith, Robert Southern, and Kimberly Kirkpatrick. Mechanisms of impulsive choice: Experiments to explore and models to map the empirical terrain. *Learning & behavior*, 51(4):355–391, 2023. 49, 51

Barry Sopher and Arnav Sheth. A deeper look at hyperbolic discounting. *Theory and Decision*, 60:219–255, 2006. 101

Peter D Sozou. On hyperbolic discounting and uncertain hazard rates. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 265(1409):2015–2020, 1998. 3

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017. 29, 71

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988. 37

Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991. 22

Richard S Sutton. Td models: Modeling the world at a mixture of time scales. In *Machine Learning Proceedings 1995*, pages 531–539. Elsevier, 1995. 36

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT press Cambridge, 1998. 11, 21, 24, 49, 52

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction, Second Edition*. MIT press Cambridge, 2018. 23

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *Proceedings of the AAAI conference on artificial intelligence*, 31(1), 2017. 10

Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993. 75

Saori C Tanaka, Kenji Doya, Go Okada, Kazutaka Ueda, Yasumasa Okamoto, and Shigeto Yamawaki. Prediction of immediate and future rewards differentially recruits cortico-basal ganglia loops. In *Behavioral economics of preferences, choices, and happiness*, pages 593–616. Springer, 2016. 36

Peter Vamplew, Benjamin J Smith, Johan Källström, Gabriel Ramos, Roxana Rădulescu, Diederik M Roijers, Conor F Hayes, Fredrik Heintz, Patrick Mannion, Pieter JK Libin, et al. Scalar reward is not enough: A response to silver, singh, precup and sutton (2021). *Autonomous Agents and Multi-Agent Systems*, 36(2):41, 2022. 23

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016. 37

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*. PMLR, 2016. 35, 36, 37, 39

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992. 21, 69

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992. 18

Zhongwen Xu, Hado P van Hasselt, and David Silver. Meta-gradient reinforcement learning. *Advances in neural information processing systems*, 31, 2018. 36, 62

Michael E Young. Discounting: A practical guide to multilevel analysis of indifference data. *Journal of the Experimental Analysis of Behavior*, 108(1):97–112, 2017. 51

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022. 29, 78

Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of reinforcement learning and control*, pages 321–384, 2021. 75