The University of New Haven

Adaptive Discounting in Reinforcement Learning

A THESIS

submitted in partial fulfillment of the
requirements for the degree of

Master of Science In Computer Science

by

Milan Zinzuvadiya

UNIVERSITY OF NEW HAVEN
West Haven, Connecticut
December 2020

Adaptive Discounting in Reinforcement Learning

APPROVED BY:

Vahid Behzadan

Vahid Behzadan, Ph.D.

Thesis Advisor

Mohsen Sarraf
Mohsen Sarraf (Dec 21, 2020 14:25 EST)

Mohsen Sarraf, Ph.D.

Committee Member

Muhammad Aminul Islam
Muhammad Aminul Islam (Dec 21, 2020 14:37 EST)

Muhammad Aminul Islam, Ph.D.

Committee Member

Barun Chandra
Barun Chandra (Dec 21, 2020 15:56 EST)

Barun Chandra, Ph.D.

Program Coordinator

Ali Golbazi
Ali Golbazi (Dec 21, 2020 15:57 EST)

Ali Golbazi, Ph.D.

Department Chair

RSHarichandran

Ronald Harichandran, Ph.D.

Dean of the College

Mario Thomas Gabouri, Ph.D.

Provost

Copyright

Acknowledgments

My graduate studies have been a journey filled with adventurous Explorations in Computer Science. I began research in the field of Machine Learning at Secured Assured Intelligent Learning (SAIL) lab, the University of New Haven in 2019. Hence, I would like to begin by thanking my advisor, guide, and supporter Prof. Vahid Behzadan at the University of New Haven for guiding me throughout research, and supporting my work on Reinforcement Learning, furthermore, supporting my work on this dissertation.

I would like to further extend my appreciation of Prof. Vahid Behzadan for his relentless support as well as nurture, for having profound belief in my abilities, for pushing me to become better version of myself, and for advising me on how to build a successful academic career. I am forever indebted to him and will always refer to him as a role model for not only my academic conduct, but also in my personal life.

I gratefully acknowledge the help of my friends and colleagues at Secured Assured Intelligent Learning (SAIL) lab. I considered myself fortunate to work with such talented researchers and will eagerly follow their inevitable accomplishments in coming years.

My Graduate studies would not have been possible without the support of my dear friends and family. I heartfully thank My grandmother - Vanitaben Jadavbhai Zinzuvadiya for being my moral compass and showing me deep-rooted wisdom. My college education cannot be possible without unwavering support of my father – Vipulbhai Zinzuvadiya. I express deepest gratitude towards my mother – Kalpanaben Zinzuvadiya for demonstrating importance of ingenious ideas and inspiring me to be a creative problem solver. I am deeply indebted to my brother- Keval Zinzuvadiya for his constant support and having profound belief in me. I am extremely grateful to Ravi Agheda and Deep Ajudiya for bringing clarity to my ideas by having valuable discussions.

Lastly, I would like to thank the esteemed members of my dissertation committee Prof. Mohsen Sarraf and Prof. Muhammad Aminul Islam for their useful feedback, comments,

and suggestions to improve this dissertation. I would also like to acknowledge guidance of my graduate advisor Prof. Barun Chandra and Dr. Ronald Harichandran for giving me opportunity to study my masters at University of New Haven.

Dedication

To My Father, Vipulbhai Zinzuvadiya, whose constant support makes my studies possible

To My Mother Kalpanaben Zinzuvadiya, who inspired me to be creative in every aspect of my life

Learning is the only thing the mind never exhausts, never fears, and never regrets.

Leonardo da Vinci

Abstract

In Markov Decision Process (MDP) models of sequential decision-making, it is common practice to account for temporal discounting by incorporating a constant discount factor. While the effectiveness of fixed-rate discounting in various Reinforcement Learning (RL) settings is well-established, the efficiency of this scheme has been questioned in recent studies. Another notable shortcoming of fixed-rate discounting stems from abstracting away the experiential information of the agent, which is shown to be a significant component of delay discounting in human cognition. To address this issue, this thesis proposes a novel method for adaptive discounting entitled *State-wise Adaptive Discounting from Experience (SADE)*. This method leverages the experiential observations of state values in episodic trajectories to iteratively adjust state-specific discount rates. We report experimental evaluations of SADE in Q-learning agents, which demonstrate significant improvements in sample complexity and convergence rate compared to fixed-rate discounting. Additionally, this thesis proposes a second adaptive discounting method for deep RL entitled *Batch-wise Adaptive Discounting from Experience (BADE)*, and reports the experimental analyses of Deep Q-Network (DQN) agents with BADE discounting in an Atari game environment. Finally, the thesis concludes with remarks on future direction of research.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Artificial Intelligence (AI) is a branch of Computer Science that is concerned with the study of intelligent agents. Within AI, Reinforcement Learning (RL) is a fundamental machine learning paradigm that enables agents to solve sequential decision-making problems from trials and errors. In RL, the agent's goal is to learn how to behave such that the cumulative rewards obtained from each of its actions is maximized. with intermediate states. Any RL problem can be formulated with Markov Decision Process (MDP) model.

Calculating the expected cumulative reward that can be obtained from each state involves the consideration of future states which are reachable from the present state. However, rational agents may assign a higher preference to rewards that may be gained sooner rather than later. Accordingly, the MDP formulation for calculating the expected return incorporates a *discount factor* ($\gamma \in [0, 1]$), where $\gamma \to 0$ implies a myopic bias towards immediate rewards, and $\gamma \to 1$ represents the long-term bias of the agent towards expected future rewards.

Conventionally, the discount factor γ is chosen to be a constant value at the beginning of the training process, and follows a geometric progression throughout all future states. However, a recent study by Naik et al. [1] questions the efficiency of this fixed-rate discounting scheme by establishing that the stationary formulation of discounted RL is not a optimization problem. Furthermore, unlike the analogue processes of human cognition, fixed-rate discounting fails to leverage past experiences in evaluating delayed preferences.

Such shortcomings of fixed-rate discounting give rise to increased sample complexity and training time. This is of particular importance in Deep Reinforcement Learning (DRL), in which the training process and cost are notoriously high.

To address such issues, this thesis investigates the idea of adaptive discounting in Reinforcement Learning. Accordingly, this work proposes a novel method for adaptive discounting entitled *State-wise Adaptive Discounting from Experience (SADE)*. This method leverages the experiential observations of state values in episodic trajectories to iteratively adjust state-specific discount rates. We report experimental evaluations of SADE in Q-learning agents, which demonstrate significant improvements in sample complexity and convergence rate compared to fixed-rate discounting. Additionally, this thesis proposes a second adaptive discounting method for deep RL entitled *Batch-wise Adaptive Discounting from Experience (BADE)*, and reports the experimental analyses of Deep Q-Network (DQN) agents with BADE discounting in an Atari game environment. Finally, the thesis concludes with remarks on future direction of research.

This thesis is organized as follows: Chapter 2 presents an overview of the required preliminaries and background for this work. This includes a brief review of MDPs and RL theory, followed by a survey of the related literature in Chapter 3. Chapter 4 introduces the proposed SADE scheme for adaptive discounting, and provides an experimental analysis of its performance in comparison to fixed-rate discounting. Chapter 5 presents the BADE approach for adaptive discounting in deep RL, and reports preliminary experimental results for performance comparison. Finally, Chapter 6 concludes the thesis with a summary of the main contributions of this work, as well as remarks on future directions of research.

Chapter 2

Preliminaries

2.1 Reinforcement Learning: A Paradigm of Machine Learning

Reinforcement Learning is concern about AI agent's ability to make good sequential decisions. In RL, The learner is not explicitly told which action to take, But instead must find out most rewarding action via exploration and experimentation. one might be tempted to think of RL as a kind of unsupervised learning because it does not rely on examples of correct behavior, RL is trying to maximize a reward signal rather than trying to find hidden structure. Uncovering structure in an agent's experience can certainly be useful in RL, but by itself does not address the RL problem of maximizing a reward sign [2]. Therefore, along with supervised learning and unsupervised learning paradigms, we consider RL to be a third machine learning paradigm.

2.1.1 Exploration-Exploitation Dilemma and delayed Consequences

One of the challenges that arise in RL, and not in other kinds of learning, is the trade-off between exploration and exploitation. To obtain a lot of reward, a RL agent must prefer actions that it has tried in the past and found to be effective in producing reward. But

to discover such actions, it has to try actions that it has not selected before. The agent has to exploit what it has already experienced in order to obtain reward, but it also has to explore in order to make better action selections in the future. The dilemma is that neither exploration nor exploitation can be pursued exclusively without failing at the task. The agent must try a variety of actions and progressively favor those that appear to be best. On a stochastic task, each action must be tried many times to gain a reliable estimate of its expected reward.

A master chess player playing game with making sequence of clever moves, is a very good example of RL in real life, which involve interaction between an active decision-making agent and its environment, within which the agent seeks to achieve a goal despite uncertainty about its environment. The agent's actions are permitted to affect the future state of the environment, thereby affecting the actions and opportunities available to the agent at later times. Correct choice requires taking into account indirect, delayed consequences of actions, and thus may require foresight or planning. At the same time, in all of these examples the effects of actions cannot be fully predicted; thus the agent must monitor its environment frequently and react appropriately.

Action taken by agent may not just affect the immediate reward, but also subsequent situations as well as rewards. Delayed consequences is one of the most important feature of RL.

2.1.2 Elements of Reinforcement Learning

Agent can have defined *actions* which allows it to interact with responsive *environment*. Every scenario which encountered by agent is called a *state*. Agent starts its exploration with *an initial state* and reach *terminal state* by travelling through various states. Agent's travelling from initial state to terminal state is called *an episode*. In episode, A path formed by states in which agent transitions happened is known as *a trajectory*. If an episode does not reach a terminal state, it falls in the category of infinite-horizon episodes.

In RL, A numerical value provided by the environment to agent as response to agent's

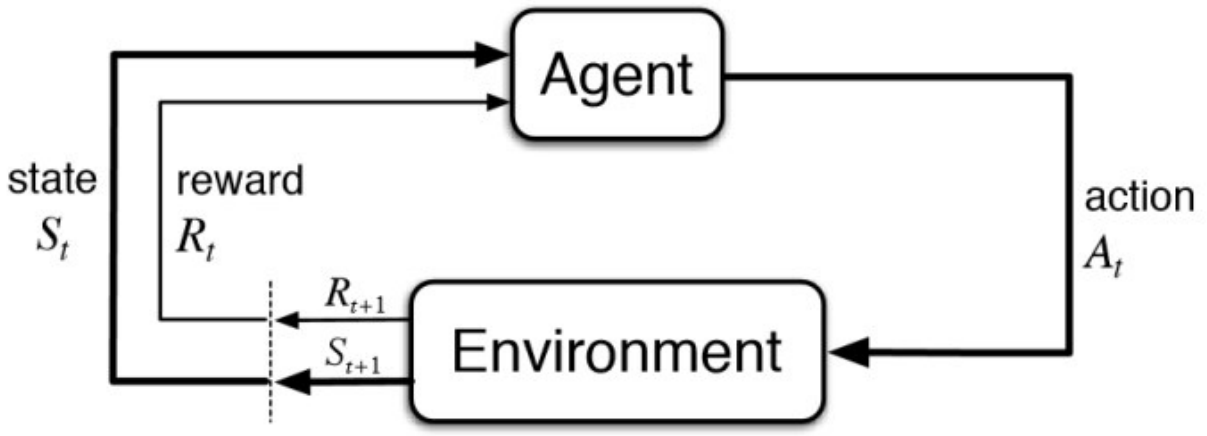

Figure 2.1: *Components of Reinforcement Learning(RL)*

action is called *a Reward*. The sum of all rewards received by agent during an episode is known as *Return* denoted by *G*. The Agent's goal is to maximize Return G, in order to achieve optimal solution.

Four key sub-elements of a RL system can be defined, beyond the agent and the environment: ***a policy***, ***a reward signal***, ***a value function***, and, optionally, ***an environment model***.

A Policy

A policy determines the way of acting of the learning agent at a given time. A policy is a mapping, roughly speaking, of perceived environmental states to measures to be taken in those states. It corresponds to what in psychology would be called a set of stimulus–response rules or associations. In some cases the policy may be a simple function or lookup table, whereas in others it may involve extensive computation such as a search process. The policy is the core of a RL agent in the sense that it alone is sufficient to determine behavior. In general, policies may be stochastic, specifying probabilities for each action.

If the agent is following policy π at time t, then $\pi(a|s)$ is the probability that $A_t = a$ if $S_t = s$. Like p, π is an ordinary function; the "|" in the middle of $\pi(a|s)$ merely reminds that it defines a probability distribution over $a \in A(s)$ for each $s \in S$. RL methods specify

how the agent's policy is changed as a result of its experience.

policy π specifies what action to take in each state. Policy can be deterministic as well as stochastic. Deterministic policy π_d have clear defined action for every state. While Stochastic policy π have probability distribution of each action can be taken by agent: $\pi(a|s) = P(a_t = a|s_t = s)$.

A Reward Signal

The objective of a RL problem is defined by a reward signal. At each step, the environment sends a single number called the reward to the RL agent. The agent's sole objective is to maximize the total reward it receives over the long run. The reward signal thus defines what are the good and bad events for the agent. In a biological system, we might think of rewards as analogous to the experiences of pleasure or pain. They are the immediate and defining characteristics of the problem the agent faces. The primary basis for modifying the policy is the reward signal; if an action chosen by the policy is followed by a low reward, then the policy can be modified to choose any other action in the future in that scenario. In general, reward signals may be stochastic functions of the state of the environment and the actions taken.

A Value Function

A value function determines what is good in the long term, while the reward signal shows what is good in an immediate sense. Roughly speaking, the value of a state is the cumulative amount of reward, starting from that state, that an agent can expect to accumulate over the future. Whereas rewards determine the immediate, intrinsic desirability of environmental states, values indicate the long-term desirability of states after taking into account the states that are likely to follow and the rewards available in those states. For example, a state might always yield a low immediate reward but still have a high value because it is regularly followed by other states that yield high rewards. Or the reverse could be true. Rewards are somewhat like pleasure (if high) and pain (if low) to make a human analogy, while values

correspond to a more refined and farsighted judgment of how happy or unhappy we are that our environment is in a specific state.

Rewards are in a sense primary, whereas values, as predictions of rewards, are secondary. Without rewards there could be no values, and the only purpose of estimating values is to achieve more reward. Nevertheless, it is values with which we are most concerned when making and evaluating decisions. Action choices are made based on value judgments. We seek actions that bring about states of highest value, not highest reward, because these actions obtain the greatest amount of reward for us over the long run. Unfortunately, it is much harder to determine values than it is to determine rewards. Rewards are basically given directly by the environment, but values must be estimated and re-estimated from the sequences of observations an agent makes over its entire lifetime. In fact, the most important component of almost all RL algorithms we consider is a method for efficiently estimating values. The central role of value estimation is arguably the most important thing that has been learned about RL over the last six decades.

The state Value function usually denoted with $V(s)$ is a measure of overall expected reward assuming the agent is in state s and then transitions through states until end of episode. State - Value function of state s can be defined mathematically as:

$$V^{\pi}(s) = \mathbb{E}\Big[\sum_{n=0}^{N} \gamma^n R_n | s, \pi\Big] \tag{2.1}$$

While State-Value function V calculates the return associated with the state, State-Action value function - Q provides the total expected reward after taking action a in state s and thereafter following policy π. Q value following some policy π can be defined mathematically as follow:

$$Q^{\pi}(s, a) = \mathbb{E}\Big[\sum_{n=0}^{N} \gamma^n R_n | s, a, \pi\Big] \tag{2.2}$$

An Environment Model

A model of the environment is the fourth and final element of some RL systems. This is something that imitates the environment's behavior, or more generally, that enables inferences to be made about how the environment is going to behave. The model could predict the resulting next state and next reward, for example given a state and action. Models are used for planning, by which we mean any way of deciding on a course of action before they are actually experienced by considering possible future situations. Methods for solving RL problems that use models and planning are called model-based methods, as opposed to simpler model-free methods that are explicitly trial-and-error learners—viewed as almost the opposite of planning. RL systems that simultaneously learn by trial and error, learn a model of the environment, and use the model for planning. Modern RL spans the spectrum from low-level, trial-and-error learning to high-level, deliberative planning.

2.2 Markov Decision Process (MDP) Formulation

Markov Decision Process (MDP) is classical formalization of sequential decision making, where actions influence immediate as well as future rewards. MDPs are mathematically idealized form of RL problem [2].

Any state s is considered to be Markov if possible future states from that state is independent of the past states. In other words, a state is called Markovian if next reachable states only depend on the current state rather than history of states. Consider a shortest GPS route from place A to place B, where the shortest path is independent from the previous positions of the agent. Because Place A is sum of all possible states in history as the agent is now in location A.

Any RL problem can be conceived as studying how to regulate the Markov Decision Process (MDP). MDP is described by tuple $MDP = (S, A, R, P, \gamma)$, where S is the set of reachable states, A is the set of available actions, R is the mapping of transitional immediate rewards,P represents the transitional probabilities and γ is discount factor.

Discounting Future Rewards : As Agent's goal is to maximize overall rewards, it is wise to take consideration of future rewards with the immediate reward while making transition into the next state. An agent can be immediate reward oriented or future reward oriented, In traditional RL, this can be preset with setting γ value from 0 to 1. In order to maximize the cumulative reward it receives in the long run, Agent seeks to maximize expected return. In fact, it selects Action t to optimize the anticipated discounted return:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.3}$$

In equation (2.3), G_t is anticipated discounted return, R_t is reward at timestamp t and γ is discount factor.($0 \leq \gamma \leq 1$) The discount rate measures the present value of future rewards: the reward received k time steps in the future is worth only γ^{k-1} times what it would be worth if it was received immediately.

If $\gamma < 1$, the infinite sum in 2.3 has a finite value as long as the reward sequence R_k is bounded. If $\gamma = 0$, the agent is "myopic" in being concerned only with maximizing immediate rewards: its objective in this case is to learn how to choose At so as to maximize only R_{t+1}. If each of the agent's actions happened to influence only the immediate reward, not future rewards as well, then a myopic agent could maximize 2.3 by separately maximizing each immediate reward. But in general, acting to maximize immediate reward can reduce access to future rewards so that the return is reduced. As γ approaches 1, the return objective takes future rewards into account more strongly; the agent becomes more farsighted. Similarly, Agent becomes shortsighted when γ is approaching to 0.

2.3 Optimal Policies and Optimal Value functions

The solution to a RL task is an optimal policy π^* that achieves the maximum expected cumulative reward over the horizon of interest (e.g., length of an episode). For finite MDPs, we can precisely define an optimal policy in the following way. Value functions define a

partial ordering over policies. A policy π is defined to be better than or equal to a policy π' if its expected return is greater than or equal to that of π' for all states. In other words, $\pi \geq \pi'$ if and only if $\forall S : V_\pi(s) \geq V_{\pi'}(s)$.

Similarly, the optimal value function and state-action value function V^* or Q^* are defined as [2]:

$$V^*(s) = max_\pi V^\pi(s) \tag{2.4}$$

$$Q^*(s,a) = max_\pi Q^\pi(s,a) \tag{2.5}$$

for all $s \in \mathbb{S}$, $a \in A(s)$ and $s' \in \mathbb{S}^+$. ($\mathbb{S}^+$ is state-space $\mathbb{S}$ plus a terminal state if RL problem is episodic)

The optimal policy can also be derived from the optimal value functions:

$$\pi^*(s) = \arg\max_a \sum_{s'} P(s,a,s')[R(s,a,s') + \gamma V^*(s')] \tag{2.6}$$

$$\pi^*(s) = \arg\max_a Q^*(s,a) \tag{2.7}$$

for all $s \in \mathbb{S}$, $a \in A(s)$ and $s' \in \mathbb{S}^+$.

2.4 Value Iteration, TD learning and Q-Learning

Value iteration refers to a class of RL algorithms that optimize the estimate of a value function (i.e., $V(.)$ or $Q(.,.)$) to extract the optimal policy from it. These algorithms utilize the recursive formulations of the value functions (Eq. (2.8) to enable the iterative estimation of such values via Bellman equations.

$$V(s) = max_a \sum_{s',R} \mathbb{P}(s', R|s, a)\Big[R + \gamma V(s')\Big] \tag{2.8}$$

$$Q(s,a) = \sum_{s',R} \mathbb{P}(s', R|s, a)\Big[R + \gamma max_{a'} Q(s', a')\Big] \tag{2.9}$$

for all $s \in \mathbb{S}$, $a \in A(s)$ and $s' \in \mathbb{S}^+$.

Temporal difference (TD) learning refers to a subset of model-free methods of reinforcement learning that learn from the current value function estimation by bootstrapping. Q-Learning is a TD control algorithm. It is considered as one of the early breakthrough of RL. It is the process of iteratively updating Q-values for each state-action pair using bellman equation until the Q-function eventually converges to Q^* (optimal q). Q-learning is defined by following equation:

$$Q(s_t, a_t) \longleftarrow Q(s_t, a_t) + \alpha\Big[R_t + \gamma \sum_{s',r} \mathbb{P}(s', R|s, a)\Big[R + \gamma max_{a'} q_*(s', a')\Big]\Big] \tag{2.10}$$

2.5 Deep Q-Learning: Q-Learning in DRL

Instead of calculating Q-values directly through value iterations, Using of neural networks as a function approximator to estimate the optimal Q-function is referred as Deep Q-Network.

Q-networks develop the Q-learning process with a number of core issues. As Q-networks are trained on each iteration, the sequential processing of consecutive observations break the *iid* (Independent and Identically Distributed) requirement of training data as successive samples are correlated. Furthermore, Slight modifications to Q-values result in accelerated policy shifts measured by Q-network, thus enabling policy oscillations. Also, since the magnitude of the rewards and the Q-values is uncertain, and the gradients of Q-networks could be broad enough to make the backpropagation mechanism unstable.

A *Deep Q-Network(DQN)* [3] is designed to overcome the issue of correlation between consecutive observation with a technique called *experience replay*: instead of focusing on successive observations, experience replaying experiments from a random collection of prior tests placed in the replay memory to learn on. As a result, the correlation between successive training samples is broken and the *iid* setting is re-established. The issue of unstability in backpropagation is also solved in DQN by technique called reward normalization: normalizing the reward values to the range $[-1, +1]$, thus preventing Q-values from becoming

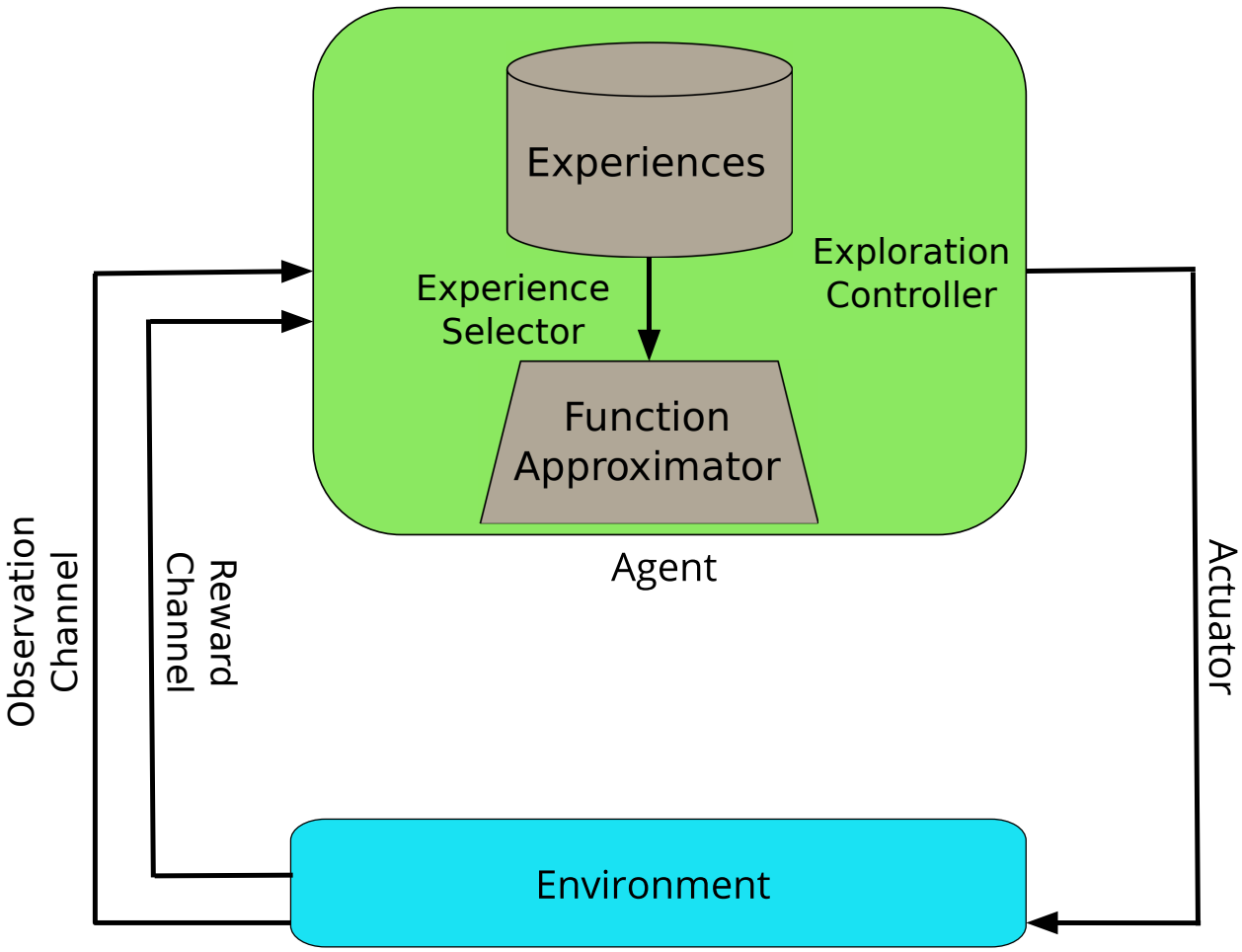

Figure 2.2: *Components of a DRL agent*

too large. In order to avoid oscillations, DQN fixes the parameters of a network $\hat{Q}$. These parameters are then updated at regular intervals by adopting the current weights of the Q-network.

With end-to-end learning of Q values in playing Atari games based on observations of pixel values in the game environment, Mnih et al. [3] demonstrate the application of this new Q-network technique. Mnih et al. use stacks of four consecutive image frames as the input to the network for capturing the movements in the game environment. To train the network, a random batch is sampled from the previous observation tuples (s_t, a_t, r_t, s_{t+1}), where r_t denotes the reward at time t. Each observation is then processed by two layers of convolutional neural networks to learn the features of input images, which are then employed by feed-forward layers to approximate the Q-function. The target network $\hat{Q}$, with parameters

θ^-, is synchronized with the parameters of the original Q network at fixed periods intervals.

The procedure of the original DQN technique is presented in Algorithm 1:

Algorithm 1 Deep Q-Network (DQN)[3]

Input: observations x_t, reward value r_t

Output: Q-function

Initialize replay memory D

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$

for *episode* $= 1$ to M **do**

Initialize sequence $s_1 = x1$ and pre-processed sequence $\phi_1 = \phi(s1)$

for $t = 1$ to T **do**

Following ϵ-greedy policy, select $a_t = \begin{cases} \text{a random action} & \text{with probability } \epsilon \\ \arg\max_a Q(\phi(s_t), a; \theta) & \text{otherwise} \end{cases}$

Execute action a_i in the emulator and observe reward r_t and state x_{t+1}

Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

Store transition $< \phi_t, a_t, r_t, \phi_{t+1} >$ in D

{// Experience Replay}

Sample random minibatch of transitions $< \phi_j, a_j, r_j, \phi_{j+1} >$ from D

Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$

Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ w.r.t. the network parameter θ

{// periodic update of target network}

Every C steps reset $\hat{Q} = Q$, i.e., set $\theta^- = \theta$

end for

end for

Chapter 3

Related Works

In order to gain a broad view of the delay discounting in RL, We synthesized latest related studies in the area of the problem. Following literature review discusses common and emerging approaches for delay discounting. Furthermore, The study provide insights into problems of stationary formulation of delay discounting.

Abhishek Naik et al.'s [1] study implies that straightforward formulation of discounted RL is fundamentally incompatible with large-scale RL in continuous tasks. This incompatibility stems from the very idea of discounting itself. The study further proves that greedily maximizing discounted future rewards do not always yields maximum average rewards.

RL methods which work by maximizing return of policy work well when environment has dense rewards that are easy to find. But when RL environment rewards lies into far future states or rewards are sparse and hard to find, RL methods tends to fails due to consideration of reward with larger exponents of a constant discount factor. Yuri Burda et al. [4] introduce a method to deal with large scale sparsely distributed rewards in RL. Yuri Burda et al.'s proposed technique incorporates a exploration bonus in the state reward. This method works well with high-dimensional obseravations and can be used with any RL policy optimization algorithm.

Vincent François-Lavet et al. [5] discuss role of discount factor in the stability and convergence of DRL algorithms. The study depicts that instead of fixed discount factor,

iteratively increasing the discount factor lead towards better learniong process. François-Lavet et al. [5]'s study inspired by empirical studies about cognitive mechanism in delay of gratification [6]. An experiment called marshamallow experiment [6] in which child is given a choice of immidiate small reward vs large reward after short period of time. child's capicity to wait for desired reward affected ability to achieve long terms rewards. If child has ability to wait for very short period of time, after becoming adult it is advantageous to achieve long term goals which requires more patience comperatevely marshamallow waiting time. It implies the potential benefit of variable for future reward. Similarly, RL may also benefit from starting to learn maximizing rewards on short horizon and progressively giving more weights to delayed rewards. So, Vincent François-Lavet et al. [5] implemented deep Q-Learning algorithm having Discount factor γ which increases after 2,50,000 steps. This algorithm provides evidence of effectiveness of adaptive discount factor γ.

To handle uncertain probabilistic future rewards, Chris Reinke et al. [7] introduce the Average Reward Independent Gamma Ensemble (AR-IGE). AR-IGE is an ensemble of discounting Q-Learning modules with a different discount factor for each module. AR-IGE can change policy when the running policy become sub-optimal compared to other Q-learning modules which are in the ensemble. While traditional algorithms only learn the optimal policy and its average rewards; AR-IGE learns multiple policies and their average rewards, so it outperforms existing RL algorithms in episodic and deterministic problems where rewards are given at several goal states. AR-IGE addresses the need adaptive discounting for different states.

William Fedus et al. [8]'s research studies hyperbolic discount factor over exponential discounted future rewards. The study indicates that in classical approaches based on monotonic decrease in exponential rewards, agent is not able to leverage maximum rewards over large horizons. This study is inspired by recent evidence of hyperbolic discounting in animals and humans.[8]

An advantage of exponential discounting is in the resulting convergence guarantees in recursive formulations (e.g., Bellman equations). To preserve this property, William H. Alexander and Joshua W. Brown [9] define a learning algorithm with Hyperbolically Dis-

counted Temporal Difference (HDTD) learning which constitutes a recursive formulation of the hyperbolic model.

This Research address aforementioned issues of stationary formulation of discounting scheme in RL. The main contribution of our work is to formulate novel method for efficient discounting scheme in RL. Furthermore, Research also provides promising leading way into Adaptive Discounting in DRL.

Chapter 4

State-wise Adaptive Discounting from Experience (SADE)

It is common practice for the discount factor to be assigned a constant value and propogate exponentially to estimate expected rewards of further states. However, the efficiency of this scheme has been questioned in recent studies. A Study by Naik et al. [1] establishes that in RL environments with continuous tasks, the stationary formulation of discounted RL is not an optimization problem.Furthermore, Fixed discounting does not leverage human-level cognition ability, which leads towards higher sample complexity and greater training time. Fixed discounting scheme dismental experiential information which is highly significant in terms of convergence rate.

To address issues in traditionally used fixed discounting scheme, we propose State-wise Adaptive Discounting from Experience (SADE)[10] as a novel adaptive discounting scheme for RL agents. SADE leverages the experiential observations of state values in episodic trajectories to iterative adjustment of state-specific discount rates. We report experimental evaluations of SADE in Q-learning agents, which demonstrate significant enhancement of sample complexity and convergence rate compared to fixed-rate discounting.

Main contributions of SADE work are:

1. Proposal of State-wise Adaptive Discounting scheme as a method to use experiential information in discounting

2. Introduce Dynamic Discounting in Reinforcement Learning

3. Development of method to reduce training time in Reinforcement Learning

The remainder of this chapter is organized as follows: Section 4.1 presents the formulation and algorithm of SADE. Section 4.2 introduces an efficiency metric as a measure of performance evaluation for discounting schemes. Section 4.3 presents the details of the environment selected for experimental analysis, and Section 4.4 provides the details of our experiment setup. Section 4.5 reports the results obtained from our experiments, and Section 4.6 concludes this chapter with a discussion on the findings.

4.1 SADE Algorithm

Considering the settings of episodic RL, SADE replaces the classical discount factor γ with a discount function $\lambda : S \rightarrow [0, 1]$, which provides a mapping of states to a corresponding discount rate. SADE hypothesizes that in the evolution of trajectories in consecutive episodes, the expected return increases for states that are more likely to be on or close to optimal trajectories. Accordingly, SADE increases the discount rate of states with increasing estimate of returns, and decreases the discount rate for vice versa.

In SADE, each state s is mapped to a specific discount factor, denoted by λ_s. Assume that at timestep t, expected return is G_t^{SADE} and the expected return of the immediate next state is G_{t+1}^{SADE} then, $G_t^{SADE} = R_t + \lambda_s * G_{t+1}^{SADE}$. The proposed discount function incorporates the past experiences of agent by adjusting the discount rate of each state based on the timesteps needed to achieve it. Accordingly, the SADE-discounted return of an agent within the

finite horizon k is given by:

$$G^{SADE}(s_t) = R_t + \sum_{k=1}^{\infty}(\prod_{1}^{k} \lambda_k(s))R_{t+k+1} \tag{4.1}$$

Where R_t is the reward at timestep t, and k is number of steps in the horizon.

Initially $\forall\lambda_s$ are assigned a value $\in (0, 1)$. After each episode during training, values of λ_s for all $s \in$ trajectory will be adjusted with a predefined adjustment rate according to SADE hypothesis. To prevent $\lambda(s)$ from becoming zero during the adjustment, we also define the values $0 < p < q < 1$ where p and q are the lowest and highest permissible values of $\lambda(s)$ for all states, respectively. The adjustment procedure is presented in Algorithm 2.

Algorithm 2 State-Wise Adaptive Discounting from Experience Algorithm

Input: adjustment rate $a\%$, upper and lower bounds λ_{min},λ_{max}, InitialGammaValue λ_{in}

Output: Q-function

Initialize $\lambda_s \leftarrow \lambda_{in}, \forall s \in S$

for each *episode* **do**

  for each $s \in Trajectory\, T_r$ **do**

    if $G^{SADE}(s_{t-1}) < G^{SADE}(s_t)$ **then**

      $\lambda_{s_t} \leftarrow \min(\lambda_{max}, a\% - increased - \lambda_{s_t})$

    end if

    if $G^{SADE}(s_{t-1}) > G^{SADE}(s_t)$ **then**

      $\lambda_{s_t} \leftarrow \max(\lambda_{min}, a\% - decreased - \lambda_{s_t})$

    end if

  end for

end for

4.2 Performance Metric

For comparison of classical approach to SADE approach for future discounting, we define a performance evalution metric for RL environment. In RL, the agent aims to maximize total rewards while reaching terminal state with minimum number of state-transitions. the agent performs poorly at the beginning of training, since it does not have any experience and needs to explore the environment. The agent improves throughout the training. An efficient RL training process is one that converges to the optimal policy in fewer episodes of training. So, it becomes necessary to compare how fast an RL process can achieve convergence in addition to how much it learns until convergence.

For comparing overall efficiency, we created a performance evaluation metric for RL approaches with consideration of maximization of total rewards, minimization of state-transitions, and rapidness to reach at convergence. With total return obtained and total visited states (which is another measure of state-transitions) during training process, we can define performance scale as follows in 4.2:

$$\begin{aligned} \text{Efficiency of RL} &= \frac{\text{(Total Reward)}}{\text{(No. of Visted States)}} \\ &= \frac{\sum Return\, R}{\sum |Trajectory|} \end{aligned} \tag{4.2}$$

for $\forall R \in$ Training and $\forall$ Trajectory $\in$ Training

4.3 Grid-world

Grid-world Game is basic and classical problem in Reinforcement Learning. With simple $(m \times n)$ Grid setting, It effectively represents RL problem formulation. With increasing m and n, Complexity of Grid-world problem increases due to increasing number of states. In Grid-world, Agent start exploration from being in (x_s, y_s) cell and reach goal cell (x_g, y_g) via travelling through cells in grid while avoiding block cells.(where $[x_s, x_g] \leq m$ and $[y_s, y_g] \leq n$

) Agent's goal is to find optimal path in grid to reach desired goal cell from starting cell.

Grid-World, shown in 4.1, contains (3x4) grid board in which black cell represents block state and agent;s goal is to find a way to end up in positive rewarding cell (1,4) from starting at (3,1).

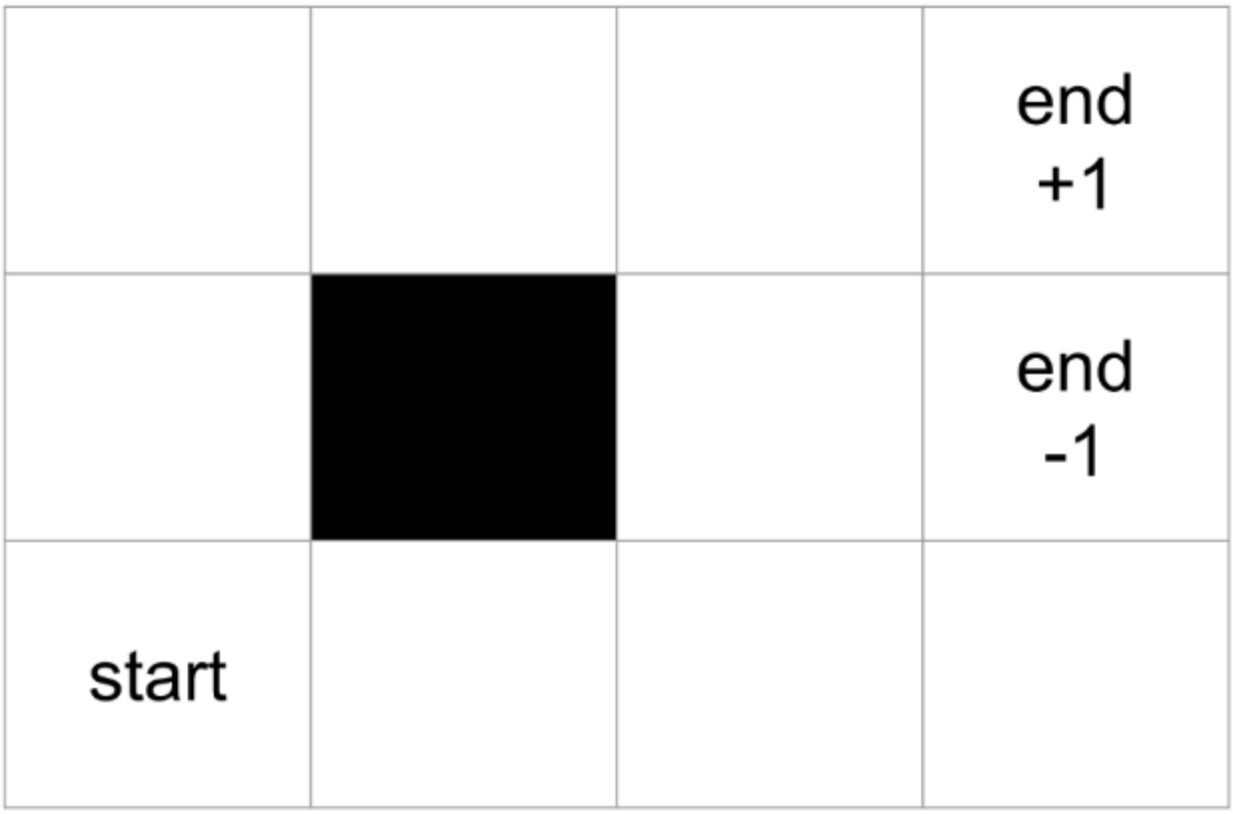

Figure 4.1: *Grid Board*

Each cell in Grid-Board is a *state*. Agent can perform 4 actions to change state : up, down, left and right. Environment setup in such a way that certain action cannot possible at certain state, like agent cannot move up or left while being in upper-left corner cell/state (1,1). Grid-world environment cannot allow agent to be in specific cells of grid, these cells known as block states of grid-world. Block-state represented by black coloured cell in 4.1. Grid-world can have multiple terminal states. States associate with numerical reward, agent can obtained the reward by being in the state.

4.4 Experiment Setup

Q-learning can be implemented to solve Grid-world environment. In which, each state (cell of the grid-board), associate with set of available actions in the state. Q value can be represent with Q(state,action) in which state $\in$ GridBoard and state $\notin$ Block-states.

We implemented Q-Learning Algorithm using python programming language. In order to provide efficiency comparison , We created 3 versions of Q-learning with separate discounting schemes: classical exponential fixed gamma factor scheme, hyperbolic fixed discount factor scheme [9] and State adaptive discount factor(SADE) scheme.

We created 6 different Grid-world environments with different complexity level listed as: $3 \times 3, 3 \times 5, 4 \times 5, 5 \times 7, 10 \times 10$, and 10×11. In each Grid-world, we perform Q-learning algorithm till it converges. In other words, We executed Q-learning algorithm until change of q-values attenuated.

For performing SADE algorithm 2, Necessary parameters are adjustment rate a which determines how many percentage change will occurs in discount factor each time, λ_{min} and λ_{max} for bounding range of discount factor varitions and finally λ_{in} for initial value of discount factor of each state.($\lambda_{in} \in (\lambda_{min}, \lambda_{max})$). For simplicity, All experiments performed with parameters described in 4.1:

Parameter	**Value**
Adjustment rate a	[1,2,3,4,5,7,10,15,20,25,30,35,40,45]
λ_{min}	0.3
λ_{max}	0.7
λ_{in}	0.5

Table 4.1: *SADE Algorithm Parameters of performed Experiments*

4.5 Results

We performed 100 experiments for each adjustment percentage mention in 4.1. As there are 14 variations of adjustment rates, each Grid-world experimented with $14 \times 100 = 1400$

experiments. In total 6 Grid-worlds, 8400 (6×1400) SADE algortihm performed with Q-Learning. Each experiment consist of Exponential and Hyperbolic fixed discount factor scheme in addition to SADE algorithm with same grid setting and After each algorithm convergence efficiency of each approach is measured for comparision. For Fixed-Exponential and Fixed-Hyerbolic discounting scheme, we used 0.5 as fixed discount factor.

4.5.1 3×3 Grid-world

Out of 1400 experiments performed, 266 times Classical Exponential fixed discounting, 250 times Hyperbolic fixed discounting and 884 times SADE turn out to be best in terms of efficiency. In 3×3 Grid-world, SADE has 230% higher success rate than Classical Exponential discounting scheme, whereas 250% increased efficiency than hyperbolic discounting scheme.

4.5.2 3×5 Grid-world

Out of 1400 experiments performed, 304 times Classical Exponential fixed discounting, 302 times Hyperbolic fixed discounting and 794 times SADE turn out to be best in terms of efficiency. In 3×5 Grid-world, SADE has 160% higher success rate than Classical Exponential discounting scheme and same 160% better than hyperbolic discounting scheme.

4.5.3 4×5 Grid-world

Out of 1400 experiments performed, 363 times Classical Exponential fixed discounting, 340 times Hyperbolic fixed discounting and 697 times SADE turn out to be best in terms of efficiency. In 4×5 Grid-world, SADE has 160% higher success rate than Classical Exponential discounting scheme and same 160% better than hyperbolic discounting scheme.

4.5.4 5×7 Grid-world

Out of 1400 experiments performed, 423 times Classical Exponential fixed discounting, 399 times Hyperbolic fixed discounting and 578 times SADE turn out to be best in terms of

efficiency. In 5×7 Grid-world, SADE has 160% higher success rate than Classical Exponential discounting scheme and same 160% better than hyperbolic discounting scheme.

4.5.5 10×10 Grid-world

Out of 1400 experiments performed, 383 times Classical Exponential fixed discounting, 404 times Hyperbolic fixed discounting and 613 times SADE turn out to be best in terms of efficiency. In 10×10 Grid-world, SADE has 160% higher success rate than Classical Exponential discounting scheme and same 160% better than hyperbolic discounting scheme.

4.5.6 10×11 Grid-world

Out of 1400 experiments performed, 419 times Classical Exponential fixed discounting, 436 times Hyperbolic fixed discounting and 545 times SADE turn out to be best in terms of efficiency. Even in 10×11, most complex Grid-world out of 6, SADE outperforms both other approaches.

4.5.7 Analysis of Efficiency vs. Environment Complexity

Figure 4.2 illustrates the comparison of efficiency between exponential, hyperbolic, and SADE discounting in 8400 experiments. It is observed that SADE maintains superior performance over fixed-rate discounting across all degrees of environment complexity.

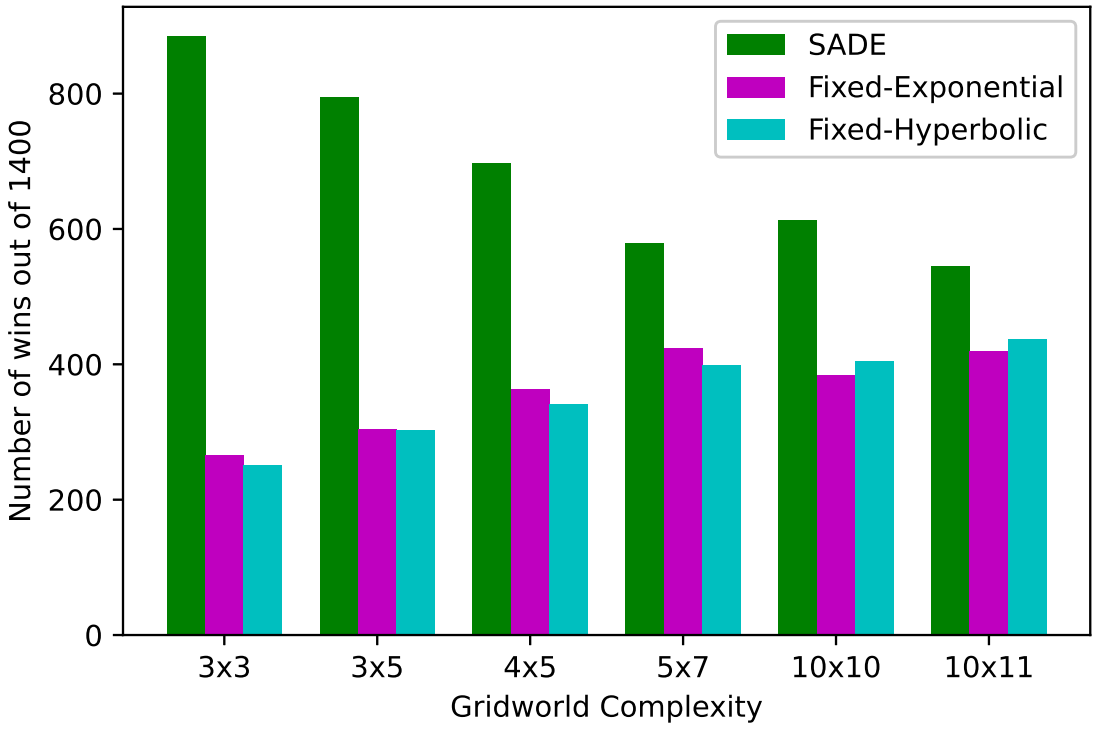

Figure 4.2: *Comparison of Efficiency between Exponential, Hyperbolic and SADE Discounting*

As Exponential and Hyperbolic curve follow quiet similar initial progression, Fixed-Exponential and Fixed-Hyperbolic discounting scheme have given similar performance on different Grid-world complexities. Although, SADE comparatively degrade performance over increasing complexities. This is due to consistent SADE parameters (4.1) on different Grid-world settings. Better performance can be achieved with parameter optimization.

4.5.8 Analysis of Adjustment Rates

If results in 4.2 categorize with Adjustment rate instead of Different Complexity level of Grid-world, Then we have 14 different category to compare results with classical as well as hyperbolic fixed discounting. Adjustment-rate comparison does not only allow us to evaluate Different Discounting scheme, But also enables us to study effect of various adjustment rate

a in SADE approach itself. Each adjustment rate performed 100 times per Grid-world. Total 600 complete training is executed with each adjustment rate. After each converge efficiency is compared to other approaches and record winner of each experiment. 4.3 depicts number of discounting outperform its competitors.

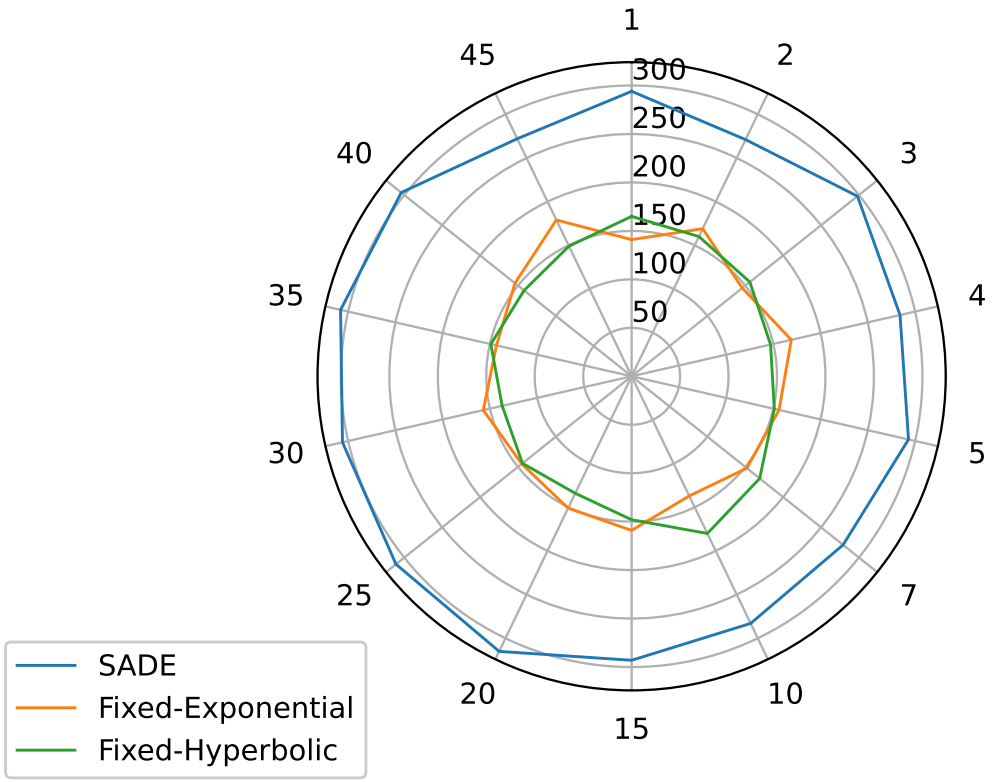

Figure 4.3: *Comparison of performance between SADE with various adjustment rates and Exponential and Hyperbolic discounting*

It is observed that SADE outperforms both fixed-discounting schemes (i.e., exponential and hyperbolic) if the appropriate range of λ_s and adjustment rates are selected. As 4.3 is overall representation of Complexities, It leads towards conclusion that SADE is better than classical Fixed-Exponential as well as Fixed-Hyperbolic Discounting scheme regardless of Adaptive-percentage of SADE.

4.6 Conclusion

As the experimental results demonstrate, SADE outperforms fixed exponential and hyperbolic discounting by at least a factor of 2 not only in terms of speed of convergence, but also in reward to episode-length ratio. As all 3 versions of Q-learning agents have identical settings except for their discounting schemes, our experiments strongly support the advantages of adaptive discounting over the classical fixed discounting.

Chapter 5

Batch-wise Adaptive Discounting from Experience (BADE) in Deep Reinforcement Learning

Considering the settings of episodic RL, SADE replaces the classical discount factor γ with a discount function $\lambda : S \rightarrow [0, 1]$, which provides a mapping of states to a corresponding discount rate. SADE hypothesizes that in the evolution of trajectories in consecutive episodes, the expected return increases for states that are more likely to be on or close to optimal trajectories. Accordingly, SADE increases the discount rate of states with increasing estimate of returns, and decreases the discount rate for vice versa.

In SADE, each state s is mapped to a specific discount factor, denoted by λ_s. Furthernore, SADE emphathize on global initialization for initial discount factor for all states. Furthermore, SADE requires discount function to be in memory, since it updates value of discount factor for each state from its latest value. These properties make SADE limited to RL problem in which state-space is small enough to handle memory issue.

Instead of consecutive observations, Deep Q-Network uses experience replay to establish iid (*Independent and Identically Distributed*) setting in Deep RL. For changing Discount factor based on state, SADE depends on the preservation of the sequence of observations,

which makes SADE inappropriate to use in Deep Reinforcement Learning (DRL). BADE extends the idea of SADE to leverage dynamic discounting scheme in DRL.

Instead of varying Discount Factor solely based on consecutive observations, Batch-wise Adaptive Discounting from Experience(BADE) leverages experiences in experience replay buffer for introducing adaptive discounting in DRL. In classical Deep Q-Network (DQN), Random sample of batch is taken from experience replay to perform Q-update on the basis of experiences found in batch.

5.1 BADE Algorithm

BADE is based on a similar hypothesis to that of SADE, that is if after a state transition the agent discovers that the current state has greater state value than the previous state, then the agent should assign a higher preference to that state from the current state and vice versa. Furthermore, BADE assumes differences of maximum Q-values of the state and its consecutive state follows Gaussian Distribution (also known as Normal Distribution) depicted by Normal curve 5.1.

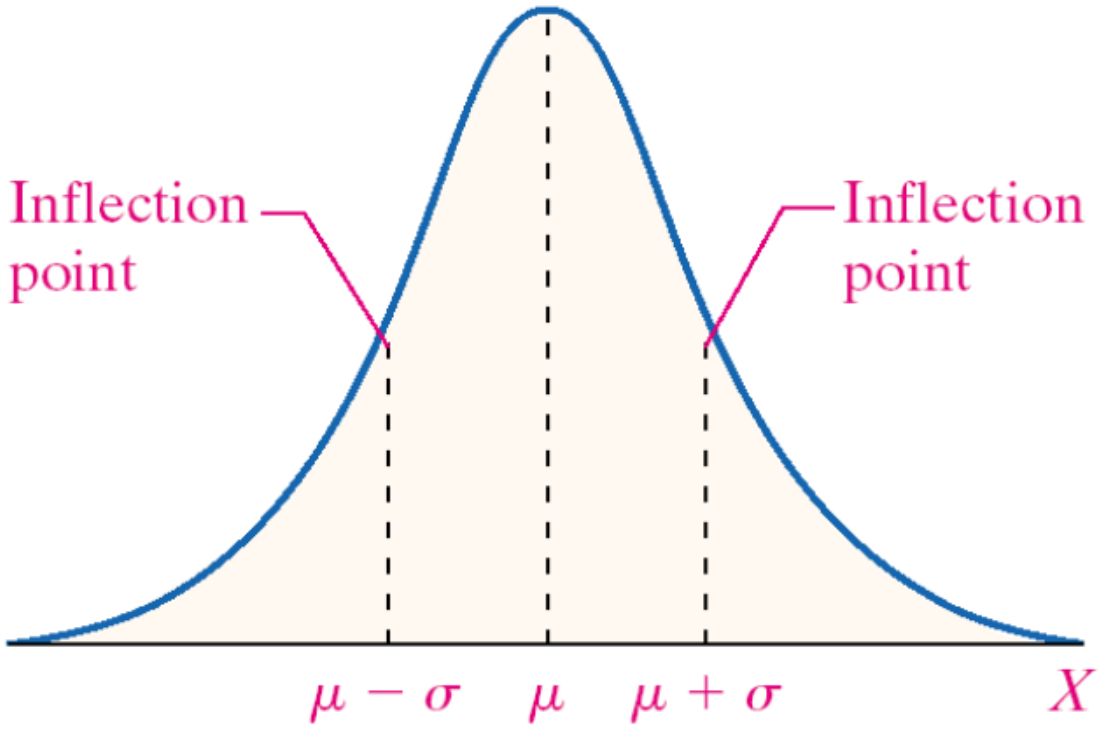

Figure 5.1: *Normal curve (Gaussian Probability Distribution curve)*

Gaussian Distribution which depicted by Normal Curve of random variable X. Normal Curve is probability distribution function:

$$f(x) = \frac{1}{\sigma 2\sqrt{\pi}} * e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{5.1}$$

where μ is sample mean and σ is Standard deviation of sample.

To introduce adaptive discounting, BADE builds on the basis of SADE, using the difference of state values between the current state and the preceding state. DQN is unstable in beginning of training of large scale RL environment. By avoiding change in the discount factor when the difference lies between inflection point of normal curve 5.1, BADE ensures stability with allowing dynamic discounting. In other words, the discount factor of an state

will not change when the value of its next state does not differ by at least the standard deviation of the batch. The area under the Normal curve where the discount factor does not change, is designated as the *Rigid Area*, while the remaining area is named the *Adaptive Area*.

The BADE algorithm adjusts the discount rate in the course of an episode, in contrast with SADE which performs adjustments at the end of episode. BADE adjusts the discount factor in 3 values: increased discount factor λ_{up}, decreased discount factor λ_{down} and λ_{rigid} discount factor for the rigid area.

After sampling a batch of experiences, BADE follows algorithm 3 to adjust the discount rate in a DQN. Algorithm 3 assumes all other hyperparameters of the DQN agent are fine-tuned, while making changes to discounting according to BADE.

Algorithm 3 Batch-Wise Adaptive Discounting from Experience Algorithm

Input: increased Discount factor λ_{up}, decreased Discount Factor λ_{down} and λ_{rigid} Rigid Discount factor

Output: Q-function

for each *batch* in Experience replay **do**

$S_b \leftarrow$ States of *batch*

$S_n \leftarrow nextStates(S_b)$

$maxQ_{sb} \leftarrow$ max Q value of each state $\in S_b$

$maxQ_{sn} \leftarrow$ max Q value of each state $\in S_n$

$diff_q \leftarrow maxQ_{sn} - maxQ_{sb}$

$\mu_{diff_q} \leftarrow meanOF(diff_q)$

$\sigma_{diff_q} \leftarrow standardDeviationOF(diff_q)$

$upperBound_{rigid} \leftarrow \mu_{diff_q} + \sigma_{diff_q}$

$lowerBound_{rigid} \leftarrow \mu_{diff_q} - \sigma_{diff_q}$

for each $(s_i, dq_i) \in (S_b, diff_q)$ **do**

if $dq_i < lowerBound_{rigid}$ **then**

$\lambda_{s_i} \leftarrow \lambda_{down}$

else if $dq_i > upperBound_{rigid}$ **then**

$\lambda_{s_i} \leftarrow \lambda_{up}$

else

$\lambda_{s_i} \leftarrow \lambda_{rigid}$

end if

end for

end for

5.2 Experiment: Atari Pong Game

Creating Video Games AI Agent is a classical task of reinforcement Learning. With using images as states and score as reward, RL agent can learn playing game.[11] When Game is over is called an episode for Agent.

For Evaluation of Game Playing Agent, Efficiency metric can be useful to compare two Deep Reinforcement Learning Algorithm. We Implemented BADE algorithm in Atari-Pong Game 5.2, a table-tennis based arcade video game, and Compare with classical Discounting in DQN.

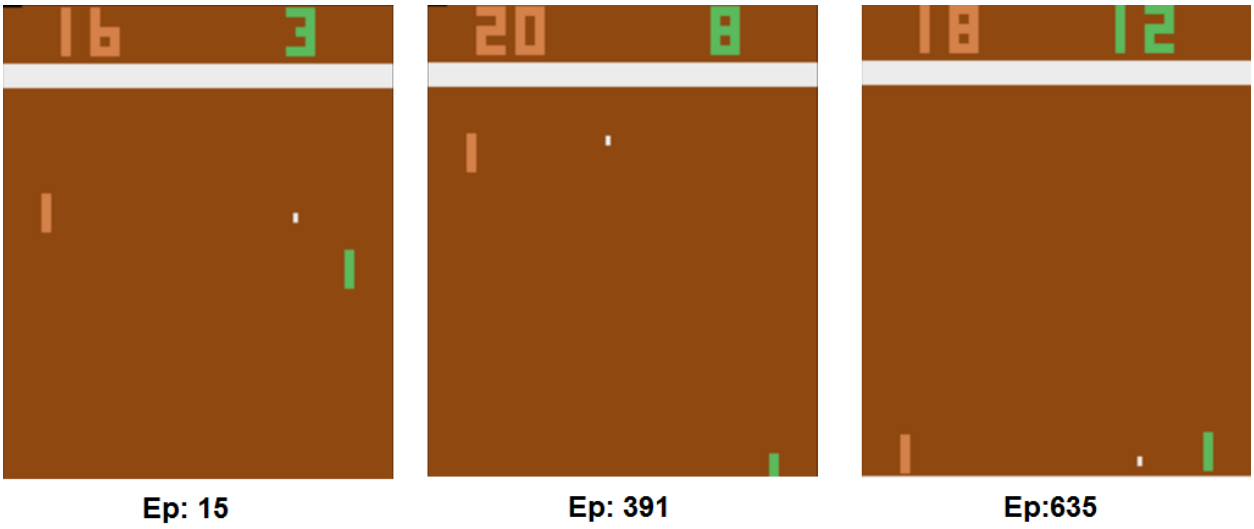

Figure 5.2: *Atari Pong*

5.2.1 Atari Pong Game: Technical Implementation

For Pong Game creation, We used Openai's Gym's 'PongNoFrameskip-v4' [12]. We implemented a DQN using stable-baselines3[13]. Then We inject BADE approach into DQN [3]. Following parameters 5.1, mentioned in stable-baselines3 [13], are used to generate results of Atari-pong based RL environment.

After implementing Classical DQN on parameters mentioned in 5.1, We made classical DQN equipped with BADE discounting algorithm. BADE parameters for experimentation mentioned in 5.2.

Parameter	Value
Environment	PongNOFrameskip-v4
Policy	CNNPolicy
Buffer Size	1e4
Learning rate α	1e-4
Batch Size	32
Learning starts	1e5
Target Network update inteval	1e3
Exploration fraction	1e-1

Table 5.1: *Classical DQN Parameters used in Experiments*

Parameter	Value
λ_{rigid}	0.99
λ_{up}	0.9999
λ_{down}	0.9801

Table 5.2: *BADE Parameters for Experiments*

5.3 Experiment Results

We obtained the following results with BADE in comparison to the classical DQN. Figure 5.3 compared efficiency of classical Fixed-Exponential as well as BADE approach implemented with DQN. In 5.3, Moving Average of latest 100 steps Efficiencies during the period of training of DQN is depicted in Y axis. Whereas, X axis is used for plotting latest step in training. Graph 5.3 represents data of 1500 Episodes i.e. 1500 complete Atari Pong games.

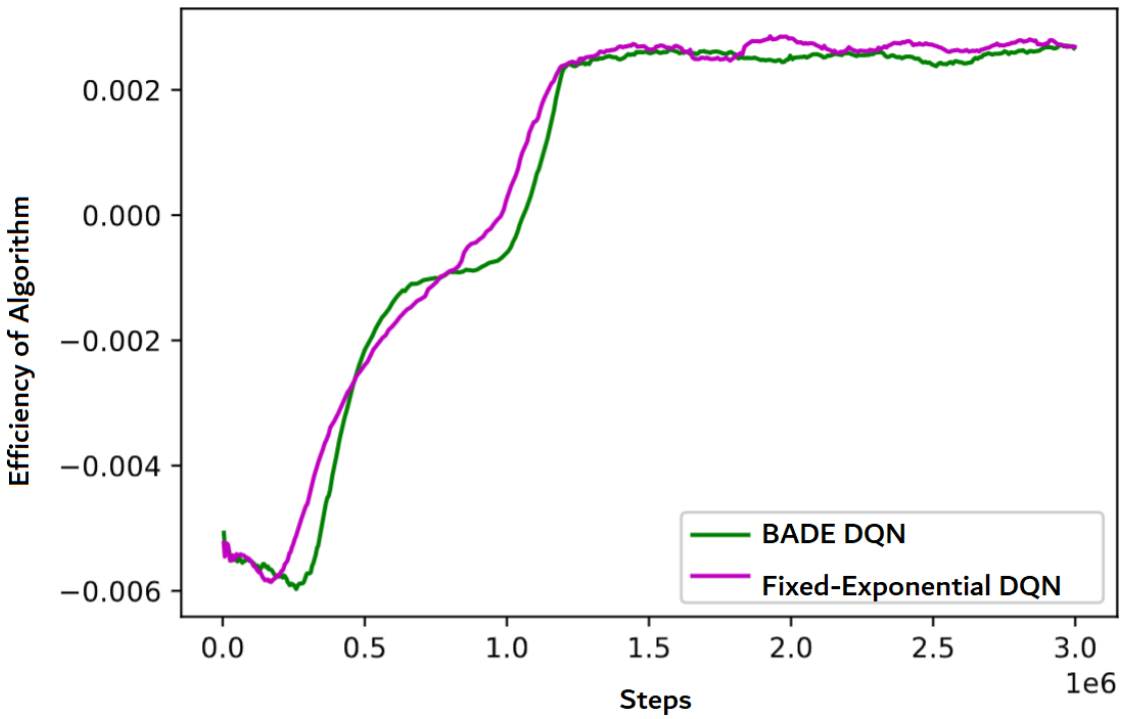

Figure 5.3: *DQN comparision*

5.3 shows how throughout the training with increasing image frames' trained on network, efficiency of respective algorithm improves. As we can see in 5.3, efficiency slope of BADE is better than classic DQN in range of ($\sim 2.5e$ to $\sim 7.5e5$) as well as ($\sim 8e5$ to $\sim 12e5$). BADE approach is performed better than classical approach if slope of BADE is maintained throughout the training. BADE requires further modifications to reach efficient Adaptive Discounting in DRL.

5.4 Concluding Thoughts

The proposed BADE approach support the idea of dynamic discounting in DRL. BADE eliminates dependency on pre-recorded state's Discount Factor, which makes BADE suitable for use in DRL. BADE seems promising path for achieving Adaptive Discounting in DRL.

Chapter 6

Conclusion

To conclude this dissertation, we summarize the main contributions of this work, and follow with remarks on directions and avenues of future research and extensions of these contributions.

6.1 Summary of Contributions

This dissertation investigated the future Discounting in Reinforcement Learning. Throughout this dissertation, the primary goal has been to develop techniques, frameworks, and guidelines that can be use for improved Discounting and overall improvement of Reinforcement Learning. To this end, we studied two fundamental research problems, listed as follows:

- Investigate and develop method for better discounting, which leads to enhance learning ability of RL agent with shorter training time
- Investigate into Deep reinforcement Learning for Dynamic Discounting

We hereafter summarize the main contributions of this dissertation towards each of these problems.

Chapter 4 presented formulations of State-wise Adaptive Discounting from Experience (SADE). Chapter provide in-depth study of Static Discounting methods: Classical Static Exponential Discount Factor and Hyperbolic Discount method. Furthermore, Chapter describe

shortcomings of Fixed discounting and provide Dynamic Method to resolve the shortcomings. Chapter highlighted difference in results of discounting methods with Q Learning implementation.

Chapter 5 investigated the effect of using SADE in Deep reinforcement Learning. Chapter listed out limitations of SADE for use in Deep reinforcement Learning. Furthermore, Chapter proposes improvements in SADE and constructed a dynamic discounting method - BADE for Deep reinforcement Learning. At last, Chapter concludes with Atari-Pong comparison, based on DQN with BADE and classical approaches.

6.2 Frontiers

Sequential decision making and RL are active fields of research with a vast horizon of unsolved challenges. In particular, the study of security considerations and issues in classical and deep RL is still a very young domain filled with research problems and opportunities for foundational contributions. Below, we introduce a number of such problems that build on the findings of the previous chapters.

- *Parameter Optimization*: This dissertation focused only on the development of novel discounting method in RL as well as Deep RL. Methods described in this dissertation are novel, so there is vast area undiscovered for techniques of proper parameter selection.

- *Need of Re-Investigation into Advanced methods using Novel Discounting*: Future Discounting is essential to RL. SADE and BADE are addressing very fundamental concept in Reinforcement Learning, Introduction of SADE and BADE arise need of reassessment of all existing methods which involves future discounting.

Bibliography

[1] A. Naik, R. Shariff, N. Yasui, H. Yao, and R. S. Sutton, "Discounted reinforcement learning is not an optimization problem," 2019.

[2] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[4] Y. Burda, H. Edwards, A. Strokey, and O. Klimov, "Exploration by random network distillation," no. arXiv:1810.12894v1, 2018.

[5] V. François-Lavet, R. Fonteneau, and D. Ernst, "How to discount deep reinforcement learning: Towards new dynamic strategies," no. arXiv:1512.02011v2, 2016.

[6] W. Mischel, E. B. Ebbesen, and A. R. Zeis, "Cognitive and attentional mechanisms in delay of gratification," 1972.

[7] C. Reinke, E. Uchibe, and K. Doya, "Average reward optimization with multiple discounting reinforcement learners," 2017.

[8] W. Fedus, C. Gelada, Y. Bengio, M. G. Bellemare, and H. Larochelle, "Hyperbolic discounting and learning over multiple horizons," 2019.

[9] W. H. Alexander and J. W. Brown, "Hyperbolically discounted temporal difference learning," 2010.

[10] M. Zinzuvadiya and V. Behzadan, "State-wise adaptive discounting from experience(sade): A novel discounting scheme in reinforcement learning," *AAAI*, 2021.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013.

[12] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[13] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, "Stable baselines3." https://github.com/DLR-RM/stable-baselines3, 2019.

milan thesis proposal signature

Final Audit Report 2021-01-05

Created:	2020-12-21
By:	Gail Berardesca (GBerardesca@newhaven.edu)
Status:	Signed
Transaction ID:	CBJCHBCAABAAVxrwVcmQ9oBEMlOcVXr9_vMEgCBI_BgQ

"milan thesis proposal signature" History

Document created by Gail Berardesca (GBerardesca@newhaven.edu)
2020-12-21 - 7:18:19 PM GMT- IP address: 73.218.89.175

Document emailed to Vahid Behzadan (vbehzadan@newhaven.edu) for signature
2020-12-21 - 7:21:53 PM GMT

Email viewed by Vahid Behzadan (vbehzadan@newhaven.edu)
2020-12-21 - 7:23:06 PM GMT- IP address: 69.113.79.130

Document e-signed by Vahid Behzadan (vbehzadan@newhaven.edu)
Signature Date: 2020-12-21 - 7:23:25 PM GMT - Time Source: server- IP address: 69.113.79.130

Document emailed to Mohsen Sarraf (msarraf@newhaven.edu) for signature
2020-12-21 - 7:23:27 PM GMT

Email viewed by Mohsen Sarraf (msarraf@newhaven.edu)
2020-12-21 - 7:25:14 PM GMT- IP address: 104.47.46.254

Document e-signed by Mohsen Sarraf (msarraf@newhaven.edu)
Signature Date: 2020-12-21 - 7:25:38 PM GMT - Time Source: server- IP address: 100.35.174.246

Document emailed to Muhammad Aminul Islam (mislam@newhaven.edu) for signature
2020-12-21 - 7:25:40 PM GMT

Email viewed by Muhammad Aminul Islam (mislam@newhaven.edu)
2020-12-21 - 7:36:51 PM GMT- IP address: 73.182.225.78

Document e-signed by Muhammad Aminul Islam (mislam@newhaven.edu)
Signature Date: 2020-12-21 - 7:37:47 PM GMT - Time Source: server- IP address: 73.182.225.78

Document emailed to Barun Chandra (bchandra@newhaven.edu) for signature
2020-12-21 - 7:37:49 PM GMT

Email viewed by Barun Chandra (bchandra@newhaven.edu)
2020-12-21 - 8:01:48 PM GMT- IP address: 104.47.74.126

Document e-signed by Barun Chandra (bchandra@newhaven.edu)
Signature Date: 2020-12-21 - 8:56:38 PM GMT - Time Source: server- IP address: 69.124.164.237

Document emailed to Ali Golbazi (agolbazi@newhaven.edu) for signature
2020-12-21 - 8:56:40 PM GMT

Email viewed by Ali Golbazi (agolbazi@newhaven.edu)
2020-12-22 - 0:56:22 AM GMT- IP address: 73.218.195.105

Document e-signed by Ali Golbazi (agolbazi@newhaven.edu)
Signature Date: 2020-12-22 - 0:57:04 AM GMT - Time Source: server- IP address: 73.218.195.105

Document emailed to Ronald Harichandran (rharichandran@newhaven.edu) for signature
2020-12-22 - 0:57:06 AM GMT

Email viewed by Ronald Harichandran (rharichandran@newhaven.edu)
2020-12-22 - 3:38:43 AM GMT- IP address: 73.61.255.93

Document e-signed by Ronald Harichandran (rharichandran@newhaven.edu)
Signature Date: 2021-01-05 - 4:07:10 AM GMT - Time Source: server- IP address: 73.61.255.93

Agreement completed.
2021-01-05 - 4:07:10 AM GMT