DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2021

Dopamine Waves Lead to a Swift and Adaptive Reinforcement Learning Algorithm

GERGÖ GÖMÖRI

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Dopamine Waves Lead to a Swift and Adaptive Reinforcement Learning Algorithm

GERGŐ GÖMÖRI

Master's Programme, ICT Innovation, 120 credits
Date: July 29, 2021

Supervisors: Arvind Kumar, Emil Wärnberg
Examiner: Jörg Conradt
School of Electrical Engineering and Computer Science
Swedish title: Dopaminvågor ger upphov till en snabb och adaptiv förstärkningsinlärningsalgoritm

Abstract

Accumulating evidence suggests that dopaminergic neurons show significant task-related diversity. Curiously, dopamine concentration and dopamine axon activity show spatio-temporal wave patterns in the dorsal striatum. What could be the function of this wave-like dynamics of dopamine in the striatum, particularly in Reinforcement Learning? This work introduces a novel Reinforcement Learning algorithm that exploits the wave-like dynamics of dopamine to increase speed, reliability and flexibility in decision-making. An agent can form a cognitive map by exploring the environment and obtaining the information about the expectation of time spent in each future state given a departing state (i.e. the Successor Representation). This map captures the temporal connections of the visited states and outlines several possible state transition trajectories leading to the reward. Using the cognitive map, following a single reward delivery, the reward prediction errors can be computed for each state. In the cognitive map, states leading to the reward possess a high positive error, while temporally distant states retain smaller errors. Thus, the dynamics of errors exhibit a wave front travelling in the cognitive map. Under the assumption of the neurons representing adjacent states in the cognitive map are also spatial neighbors, it automatically follows that the reward prediction error carrying signal will also show wave-like dynamics in space. By exploiting the dopamine waves, the proposed Reinforcement Learning approach outperforms three classical Reinforcement Learning algorithms: basic SARSA, the Successor Representation and SARSA with eligibility traces. Consequently, the algorithm suggests conditions under which wave-like dynamics of dopamine release in the striatum can have direct functional implications for learning.

Keywords: Reinforcement Learning, Dopamine, Basal Ganglia, Successor Representation

Sammanfattning

En ökande mängd bevis pekar på att dopaminerga nervceller uppvisar en betydande uppgiftsrelaterad diversitet. Märkligt nog uppvisar såväl dopaminkoncentrationen som aktiviteten i dopaminerga axon i dorsala striatum en vågliknande dynamik. Vilken funktion kan dopaminets vågliknande dynamik tänkas fylla i striatum, särskillt vid förstärkningsinlärning? I detta arbete introduceras en ny förstärkningsinlärningsalgoritm som utnyttjar dopaminets vågliknande dynamik för att öka snabbheten, tillförlitligheten och flexbiliteten vid beslutsfattande. En agent kan skapa en kognitiv karta genom att utforska en miljö och tillgodogöra sig information om den förväntade tiden som kommer tillbringas i varje framtida tillstånd givet ett starttillstånd (en så kallad successionsrepresentation). Denna karta fångar upp de tidsmässiga förbindelserna mellan besökta tillstånd och ger en skiss för flera möjliga serier av tillståndsövergångar som leder till belöning. Genom att använda denna kognitiva karta efter en enskild belöning kan belöningsförutsägningsfel beräknas för varje tillstånd. I den kognitiva kartan har tillstånd som leder till belöning ett stort positivt fel, medan tidsmässigt avlägsna tillstånd har mindre fel. Detta ger upphov till att dynamiken för felen uppvisar en vågfront in den kognitiva kartan. Under antagandet att nervceller som representerar närliggande tillstånd i den kognitiva kartan också är fysiska grannar, följer det automatiskt att signalen för belöningsförutsägningsfel också uppvisar en våglikannde dynamik i rummet. Genom att utnyttja dopaminvågor överträffar den föreslagna förstärkningsinlärningsalgoritmen tre klassiska förstärkningsinlärningsalgoritmer: vanlig SARSA, successionsrepresentation, och SARSA med kvalificeringsspår. Algoritmen förslår därför betingelser under vilka en vågliknande dynamik av dopaminfrisättning i striatum kan ha direkta funktionella implikationer för inlärning.

Nyckelord: förstärkningsinlärning, dopamin, basala ganglierna, successionsrepresentation

Acknowledgments

I am grateful for the many exciting conversations with my supervisors, Arvind Kumar and Emil Wärnberg and their contionous attention throughout this research. I greatly appreciate the help of Emil in translating my abstract. I would also like to thank my family their emotional support, I could not have been able to finalize my thesis without their encouragement.

Stockholm, July 2021
Gergő Gömöri

Contents

List of Figures

List of Tables

List of acronyms and abbreviations

AGI Artificial General Intelligence

RL Reinforcement Learning

SDG Sustainable Development Goal

SR Successor Representation

TD Temporal-Difference

UN United Nations

VTA Ventral Tegmental Area

Chapter 1

Introduction

From the traditional holistic perspective, this work lies in the intersection of Computer Science, Neuroscience and Psychology. Animlas, including humans tend to learn by reinforcing actions which led to rewards, thus there is great interest in finding the neural correlates of Reinforcement Learning (RL) [4]. My research focused on how certain biological processes shape behaviour and affect action selection in simple, virtual environments. However, in contrast to classical approaches, algorithms in the emerging field of Artificial Intelligence were in the spotlight. The key concept of this project was to incorporate experimental neuroscientific discoveries into the domain of RL. This not only led to a new hypothesis in understanding biological processes, but inspired the development of a novel algorithm as well.

1.1 Problem

The basal ganglia, in particular its input nucleus, the striatum have emerged as the primary locus of RL in the brain [5]. Significant experimental evidence supports the theory that in RL tasks, the neuromodulator called dopamine carries information about reward prediction errors [2, 6]. According to a recent research, wave-like patterns in the concentration dopamine were observed in the striatum [3]. What could be the function of these highly organized patterns? Would relating this phenomenon to a computational process integrated in a RL algorithm lead to advantages over classical approaches?

Among the wide variety of RL methods, the Successor Representation (SR) lays down the foundations for separating the learning process of the environment dynamics, and the estimation of rewarding states [7]. In this work, I address the following question: is there a way to extend this approach via

parametrization, and would it be advantageous over already existing algorithms?

1.2 Purpose

This work contributes to the "Good Health and Well-being" (3rd) and the "Industry, Innovation and Infrastructure" (9th) goals of the Sustainable Development Goals (SDGs) formulated by the United Nations (UN) [8]. Providing a deeper understanding of the function of dopamine, it aims to inspire research carried out in the fields of addiction and Parkinson's disease, both of which are related to SDG number 3. The project also connects biological processes to computational methods, thus takes a step towards neuroscience-inspired artificial intelligence, an area which is related to SDG number 9.

1.3 Goals

The main goal of this project is to develop a suitable algorithm and to provide answers to the questions formulated in Section 1.1. This goal was divided into the following five subgoals:

1. Acquire an understanding of the recent discoveries in the field of the neural basis of decision making. Develop solid foundations in RL.
2. Implement a simulation environment which is suitable for executing experiments in a RL setting.
3. Develop an algorithm which is related to the outcome of Subgoal 1 and builds upon the SR. It shall possess guaranteed convergence properties.
4. Implement the novel algorithm and several classic RL algorithms.
5. Benchmark the new algorithm against classic RL approaches.

Section 4.3 serves as the outcome of Subgoal 1. The pseudocode, as the main devilerable of Subgoal 3 can be found in Appendix A. The results of Subgoal 5 are presented in Chapter 5. Finally, the code related to the Subgoals 2, 4 and 5 can be found online [9].

1.4 Research Methodology

The present work is purely theoretical, simulations were executed in a virtual environment. The SR approach was taken as a basis for further development. The new design was shaped in a way to match the specific biological process of wave-like dopamine propagation in the dorsal striatum. Taking such an incremental approach, one modification could be analyzed instead of having an overwhelming set of extensions.

1.5 Delimitations

Only a limited set of biological phenomena was incorporated into a RL algorithm. The aim of this project was not to model a high number of biological processes underlying decision-making, rather to extend an already existing algorithm based on the aforementioned observation of dopamine waves.

Chapter 2

Background

2.1 Reinforcement Learning

2.1.1 Foundations of Reinforcement Learning

RL is a simple, but profound framework designed to understand decision-making. The basic setup is shown on Figure 2.1.

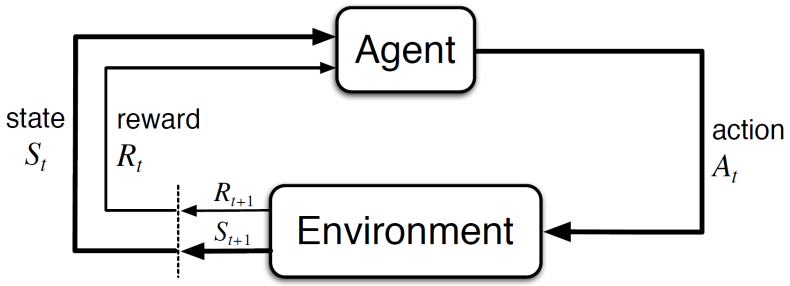

Figure 2.1: The Reinforcement Learning framework highlighting the sequential process of interaction between the environment and the agent [1]

At its core, it has two key components: an "agent" and an "environment". A simulation in this framework shall be thought of as a sequential process, where at a given time t, the agent observes the current state of the environment S_t, executes an action A_t based on its observation, obtains a scalar-valued reward R_{t+1} and arrives to the next state S_{t+1}. From there, the same process is repeated until termination. One complete sequence from the initial moment until termination is referred to as an episode and in an episode, the agent fol-

lows a trajectory. It is common to mark the next state with S' and the next action with A'.

The state space $\mathcal{S}$ contains all possible states of the environment. An element of it, $S_t \in \mathcal{S}$ can be represented with one or more values and in general these values can be both an element of a discrete set or have a continuous value. The action space $\mathcal{A}$ contains all possible actions the agent can take. Its elements $A_t \in \mathcal{A}$ can be represented similarly: by one or more values from discrete sets or by continuous values. In general, it is conditioned on the actual state S_t, since it can happen that some actions are not available in certain states. However, in order to simplify the notation, this is not indicated in this work, but it was considered throughout the project. The RL problem definitions in this work have discrete state $\mathcal{S}$ and action $\mathcal{A}$ spaces, both are represented with only one value.

An example for a RL problem definition inspired by a real-world task can be climbing a cliff. An intermediate moment could be that the climber is standing on two specific spots on the crag, while grabbing two pebbles. An exact description of her location and her surroundings is the state she is in. The set of actions she could choose from are to grab another pebble or step on a different edge. During the climb, no reward is received, however reaching the peak is a joyful moment. Two special attributes of environments are shown in this example. The first one is that it has a terminal state, the peak of the mountain which ends the episode of the agent. These problems are called episodic tasks. The second one is that in this example there is only one rewarded state. Intermediate rewards are 0, except for the terminal state, which can be modelled with a reward of 1.

2.1.2 The value function

A naturally arising question is, how does the agent make decisions? At this point, RL branches out. As it is an emerging field, many possible approaches exist. The algorithms in this work utilize the concept called value function, thus only that approach is presented. This function assigns a value to each state [1]:

$$v_\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s \right] \tag{2.1}$$

Where:

- π is the policy,

- $\gamma \in \mathbb{R}$ is the discount factor. Possible values are $0 < \gamma < 1$.

The discount factor γ is used to consider the rewards that the agent will receive in the future. Thus a small value of γ reduces the time horizon what the agent considers, however a large γ expands such a limitation of the agent. Due to this concept, the term in brackets is also called "discounted future return". Different concepts for defining the policy π exist, however in this work the policy π shall be thought to be a function, which assigns one action for each state. This means that according to Equation 2.1, the value of an arbitrary state s is the expected return of the trajectory which is generated when the agent starts at S_t and follows the policy π until termination.

Considering an episodic task with only one rewarded state, a trajectory can be found which leads to the terminal state in the least number of steps an agent needs to make. Given this trajectory, the value of each subsequent state is monotonically increasing, due to the fact that the agent is getting closer and closer to the reward with every action it makes. The increase of each further step depends on the discount factor γ.

The value function can not only be defined with respect to states, but also with respect to state-action pairs [1]:

$$q_\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s \wedge A_t = a \right] \tag{2.2}$$

According to Equation 2.2, the value of a state-action pair (s, a) is the expected value of the trajectory which is generated when the agent starts at S_t, takes action A_t and follows the policy π until termination.

State-action pairs can be thought of as an extension of the state space $\mathcal{S}$. Coupling these two values is beneficial, since these tuples represent the reward for taking a specific action in a specific state, capturing the most crucial signal for agents in RL problems.

Learning the true value function is not possible without exploring the environment, such a trial-and-error process is inevitable. An approach to force exploration is the ε-greedy policy, where for all of the occurring states, there is a small chance that a random action is selected from the action space $\mathcal{A}$, while in all remaining cases the "greedy action" is selected. The greedy action is the action with the largest value among all state-action pairs for a given state. The ratio of the random actions is controlled by a value which is denoted by ε and its value is $0 \leq \varepsilon \leq 1$. In case $\varepsilon = 0$, the policy is purely greedy, if $\varepsilon = 1$ the agent is only exploring. A common approacth is to use an adaptive exploration ratio ε which decreases with the number of episodes taken. This

means that as the agent learns a better and better value function, exploration is not necessary anymore and the greedy strategy becomes closer to the optimal policy.

2.1.3 Temporal-Difference Learning

Besides the value function, another important concept in RL is Temporal-Difference (TD) Learning [10]. This is the crucial idea which serves as an important link between Reinforcement Learning and Neuroscience. (The details are described in Subsection 2.2.1.)

As the agent starts exploring the environment, given a state s and the next state s', it is able to update its approximation of the value function according to Equation 2.3 [1]:

$$\hat{v}(s) \leftarrow \hat{v}(s) + \alpha[r + \gamma\hat{v}(s') - \hat{v}(s)] \tag{2.3}$$

- r is the reward obtained after taking an action from state s,
- $\alpha \in \mathbb{R}$ is the learning rate with $0 < \alpha < 1$, most RL algorithms use a value closer to 0.

The term multiplied by the learning rate α is called the TD error, because it computes the difference between an estimate made previously, the $\hat{v}(s)$ and a better estimate namely $r + \gamma\hat{v}(s')$. The "temporal" term originates from the fact that it is an error made at exactly the previous moment, when the next state s' was not known yet.

2.1.4 The Successor Representation

Finally, in this subsection one particular concept in the wide range of RL algorithms is introduced. The approach is called "Successor Representation" and it was developed by Peter Dayan in the early 1990s [7]. Despite the fact that its publication was decades ago, it was not until recently that it began to receive more attention due to its neuroscientific implications [11]. There is experimental evidence that humans implement a decision-making process aligned with SR [12].

At its core, the SR decouples the process of learning the environment dynamics and the reward estimation. Due to this concept, this algorithm is very flexible against changes in the reward signal. Given an environment of d states, it computes the value function with the matrix product shown in Equation 2.4.

$$\hat{v}(s) = \mathbf{M}(s) \cdot \mathbf{R} \tag{2.4}$$

Where:

- $\mathbf{M} \in \mathbb{R}^{d \times d}$ is the successor representation matrix, $\mathbf{M}(s)$ is the row corresponding to s,
- $\mathbf{R} \in \mathbb{R}^{d}$ is the reward estimation column vector.

The successor representation matrix $\mathbf{M}$ captures the total time expected to be spent in each state given a starting state. Thus, it is a matrix with the same number of rows and columns as the number of states. Every row represents a starting state and every column indicates a future state. This means that a given state s highlights a row $\mathbf{M}(s)$. Each element of that row (i. e. each column of that matrix) depicts an estimation on how often the agent will end up in all next states. Naturally, if a state s in the environment directly leads to another state s', then $\mathbf{M}(s, s')$ will have an increased value over the other future states.

The reward estimation vector $\mathbf{R}$ aims to predict how much reward is obtained when arriving to a given state. Thus, its length is equal to the total number of states and given an environment with only one rewarded state, except for one specific index of $\mathbf{R}$ every other value is 0.

Both components in Equation 2.4 are learned during each step in each episode. In the update of the SR matrix, first an error term is computed (Equation 2.5) which is used as an update (Equation 2.6). The error computed by Equation 2.5 is a vector-valued TD error.

$$\delta = \mathbf{1}_s + \gamma \mathbf{M}(s') - \mathbf{M}(s) \tag{2.5}$$

Where:

- $\mathbf{1}_s \in \mathbb{R}^{d}$ is a row vector with only zeros, except for the index representing the actual state s where the value is 1 (i. e. a one-hot vector).

$$\mathbf{M}(s) \leftarrow \mathbf{M}(s) + \alpha\delta \tag{2.6}$$

Given an arbitrary state s, the reward estimation vector $\mathbf{R}$ can be learned by adding a small increment to the previous estimate $\mathbf{R}(s)$. The increment can be the observed error scaled by a factor.

Equations 2.4, 2.5 and 2.6 can be defined on state-action pairs in a similar way. This concept was used in the novel algorithm described in Chapter 4.

The remaining rather classical RL algorithms used in this thesis are described in [1].

2.1.5 Outlook

In recent years, algorithms embracing the concepts of RL achieved remarkable results. An example is the first algorithm, which defeated the world champion in the game of Go [13]. In addition, last year an algorithm called "Agent 57" managed to outperform the human benchmark in all 57 games of the Atari game pool [14]. Real world applications of RL are emerging as well. For instance, there is a method designed to control balloons in an outer layer of the Earth's atmosphere [15]. Finally, RL is a promising field in the development of Artificial General Intelligence (AGI), since it is hypothesized that the reward signal is by itself is sufficient to develop intelligent behaviour [16].

2.2 The neural correlates of RL

2.2.1 Initial experiments

It was known from the 1950s that electrical stimulation of certain brain areas have a significant impact on animal behaviour [17]. Later it was discovered that these areas activate dopaminergic pathways, thus in the 1990s, an interesting set of experiments were conducted with the aim of unfolding the relationship between dopamine neuron activity and motor control in monkeys. In one particular experiment, the task of the animal was to touch a lever after the appearance of a light. The dopamine neuron activity altered after several repetitions of the same task [2]. The prominent discovery made by this thoughtful analysis is shown on Figure 2.2.

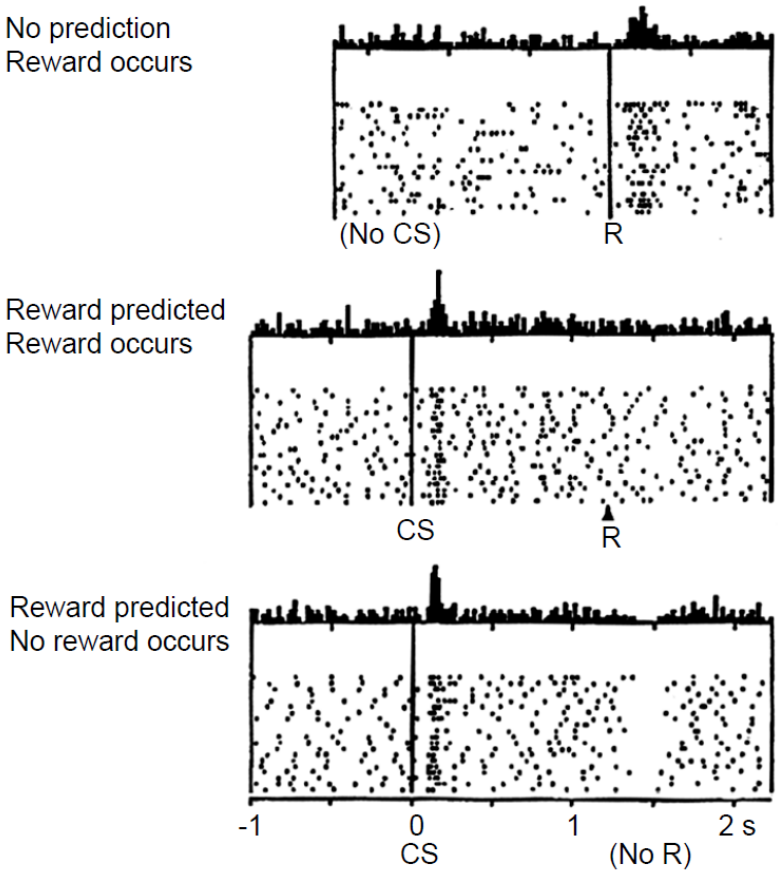

Figure 2.2: Spike raster plots and histograms of dopamine neuron activity represent Temporal-Difference error [2]. Measurements obtained from animal experiments

On Figure 2.2 three spike raster plots and three histograms can be seen. The "x" axis is the time axis, each black dot represents the firing of a dopamine neuron. Distinct neurons are lined up along the "y" axis. In the moment marked with "R", the monkey received a reward in the form of a drop of fruit juice. "CS" marks the time when a "conditioned, reward-predicting stimulus" was initiated. The phenomenon behind such a stimulus is that if a certain event happens, and subsequently the animal receives a reward, from the perspective of the animal the event becomes associated with the reward after a certain period of training. In the aforementioned experiment, this event was the illumination of a small light. Thus, when a light was on, the animal was expecting the reward. Above the raster plots, a histogram can be seen which summarizes the firing of the observed neurons: if a large ratio of the neurons

fired at approximately the same short time period the height of the bar is large, if most neurons remained silent, its height is small.

In the first experiment, obtaining a reward causes a peak in the average firing rate of the dopamine neurons. In the second experiment, the conditioned, reward-predicting stimulus causes a similar increase, however reward acquisition does not show any particular consequence. In the third experiment, the conditioned stimulus causes a peak again, however in this case no reward follows, the lack of reward results in a dip at the exact moment when it was expected based on the previous two experiments.

Hypothesizing that the average firing rate of dopamine neuron corresponds to the reward prediction error, in particular to the TD errors computed by a RL agent, the resemblance is striking. In the first experiment, the animal was not expecting a reward, but a sudden reward was delivered, which can be thought of a high positive TD error, which reflects dopamine activity. In the second experiment, the moment the animal sensed the conditioned stimulus, it realized that a reward will follow, which again is a high positive TD error in accordance with dopamine activity. Since it was expecting the reward, gaining the drop of fruit juice was synchronous with its expectations, thus the TD error was zero. The third experiment is the same as the second one, except for the moment when the expectations of the animal were not fulfilled due to withholding the reward. According to the RL model, this means a large negative TD error which reflects the dip of the firing rate of dopamine neurons.

These initial experiments are consistent with novel work [6]. As a remark, the fact that there is only one rewarded state in the environments utilized in this thesis is aimed to reflect similar systems neuroscience experiments. Without considering the internal needs and biological processes of the animal, a reward as an external event is delivered to it at the terminal state.

2.2.2 Striatal dopamine waves

Dopamine neurons can be found in two brain regions: the pars compacta of the substantia nigra and the Ventral Tegmental Area (VTA). The dopamine neurons of the former project to the dorsal striatum, the latter project to the ventral striatum and the prefrontal cortex. These connections give rise to the nigrostriatal and the mesocorticolimbic pathways [18].

As introduced in Subsection 2.2.1, the initial hypothesis was that the firing rate of dopamine neurons is solely the reward prediction error, a scalar-valued signal which is being broadcasted to target structures. New experimental results however are able to depict subtle differences, which highlight task-related

diversity among dopamine neurons [5, 19, 20]. Such a variety is elegantly reflected in the observation of spatiotemporal dopamine waves in the dorsal striatum of mice [3]. This work was in the spotlight throughout this research as indicated in Section 1.1.

Three main wave patterns were observed: latero-medial, medio-lateral and center-out waves. (The naming convention of the former two follows the anatomical terminology.) The three motifs are visualized on Figure 2.3.

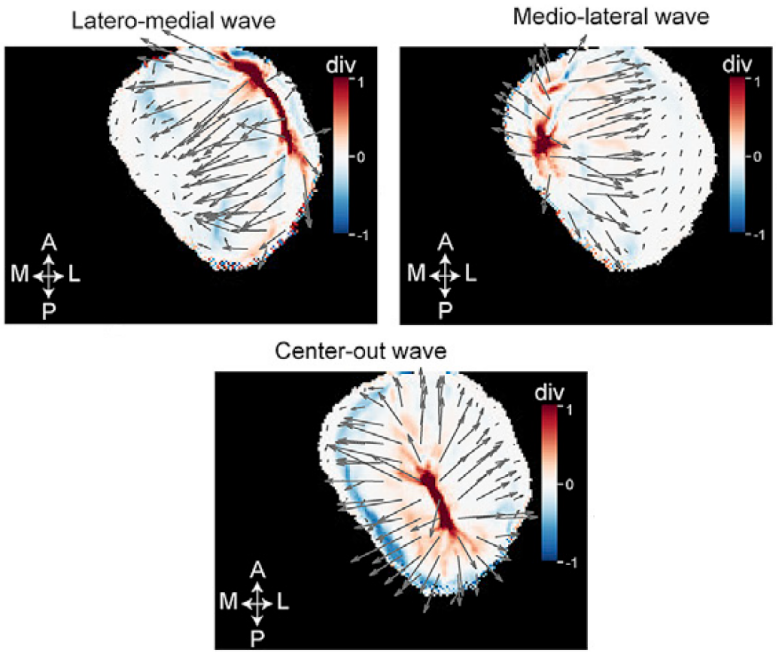

Figure 2.3: The three motion types of wave-like dopamine neuron activity in the dorsal striatum [3]. The naming convention of the patterns are indicated above each diagram

Two tasks were given to mice. In both cases they had to run on a wheel. In the first task if one was running faster, it got the reward in the form of water sooner. However, in the second task it had no control of the time when it got the reward, the time between the start of the experiment and the termination was fixed. Interestingly, after training, the execution of the different tasks were reflected in the direction of the striatal dopamine waves: latero-medial waves were initiated when the mouse had no control on reward delivery, and medio-lateral waves were observed when running faster resulted in getting water sooner. A general RL model was proposed in the article, however an in-depth description of the function of the center-out waves was missing. The

goal of this project was to find an explanation for this specific pattern.

Lastly, experiments show that the right timing of dopamine delivery is able to regulate structural plasticity, which is a crucial contribution to learning [21].

2.2.3 Cognitive maps

Almost a century ago, in the late 1920s, a very interesting observation was made [22]. A group of rats were given the task to explore a maze, but no reward was obtained in the goal location. Suddenly a reward was introduced and the animals learned how to get it swiftly. When compared to a group of animals which had a reward already from the beginning, the pace of reward acquisition of the former group was higher. Thus, it seemed that the former group of animals were memorizing the layout of the maze even without a reward, this helped them to obtain the reward sooner in the second part of the experiment. This process is called “latent learning”.

Latent learning leads to the formation of a “cognitive map”, an abstract object which aims to capture the environment [23]. This can conveniently be done by recording sequences of "state, action, reward and next state" and organizing these experiences. States can be, but not necessarily are locations in space. The establishment of the cognitive map can be linked to the hippocampus [24, 25].

The algorithm introduced in Chapter 4 also constructs a cognitive map, which provides the basis for computing the true value of state-action pairs. This process significantly contributes to the speed of the agent in solving the tasks described in Chapter 3.

2.3 Recent related ideas

Neuroscience, Psychology and Reinforcement Learning are interwined areas of research, mutually drawing inspiration as they evolve. In this section I would like to highlight the work which I believe is the most promising.

Recent experiments indicate that a backward replay process of previous experiences aids decision making in humans [26].

Distributional RL extends the concept of the value function. Instead of mapping states to scalar values, this approach computes a distribution representing the value of each state [27, 28, 29]. Intrestingly, VTA dopamine neurons operate similarly: some neurons are “optimistic” and expect a large reward, while some neurons are “pessimistic” expecting small rewards. Thus a distribution is formed on a population level [20].

The SR influenced several neurally-plausible algorithms [30, 31]. The collaboration of the hippocampus and the striatum is efficintely utilized in an algorithm where one region of the stritum is hypothesized to compute instant responses to stimuli, while the hippocampus constructs a cognitive map [32].

The algorithm described in Chapter 4 takes a similar perspective, however in this thesis, a particular pattern of the spatiotempral dopamine distribution presented in Subsection 2.2.2 is linked with the formation of a cognitive map via an experience replay mechanism.

2.4 Interim summary

RL agents aim to construct a close approximation of the true value function with the least amount of exploration. Based on a reliable value function, decisions on action selection can be made. The formation of a cognitive map supports constructing a decent representation of the environment within a limited time, thus can be involved in the computation of the state values. Dopamine neurons indicate prediction errors and exhibit wave-like motion properties in the dorsal striatum, however the exact function of such a biological phenomena is not yet well understood. This is the gap I intend to fill in the present work.

Chapter 3

Methods

3.1 Overview of the research process

The sequence of steps followed throughout this work aligns with the subgoals presented in Section 1.3. Three custom grid-world environments were designed to achieve Subgoal 2, a detailed description of these can be found in Section 3.2. The prototypes of the algorithms were tested in this framework. After the completion of Subgoal 4, the performance of each algorithm was gauged in order to complete Subgoal 5.

The implementation was carried out in Python, including the use of the following packages:

- NumPy [33],
- Matplotlib [34],
- pandas [35],
- seaborn [36],
- NetworkX [37].

3.2 Experimental design

Three experiments were designed, all based on a grid-world environment, however either the environment dynamics or the layout was different among the experiments. Each cell was considered as a separate state, and in general, from all states four actions could be taken: move left, move right, move up, move down. In some cases, some actions were blocked, either due to being

on the edge of the environment, or due to an obstacle on an adjacent cell. The obtained reward is zero, unless the next state is the terminal state. In the latter case the reward is one.

On Figures 3.1, 3.2 and 3.3 the light green cell marks the starting location of the agent and the orange cell marks the terminal state, which is also the rewarding state.

3.2.1 The simple grid-world

In the first experiment, the agent is in a world with size 8×8 (Figure 3.1). The environment is static and has no obstacles, but is bounded by the outer boarders. On the edge of the layout, actions leading off the map are not available.

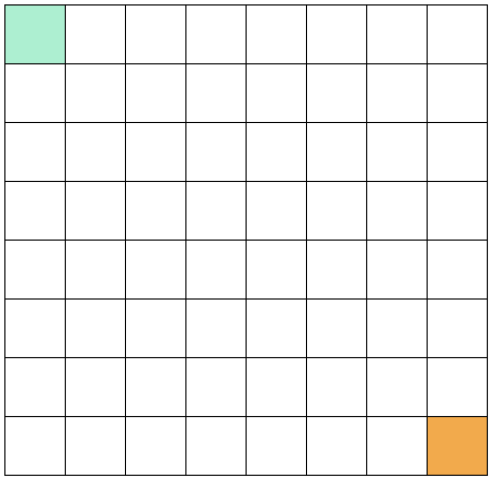

Figure 3.1: The basic layout of the grid-world. The green cell indicates the starting location, the orange cell marks the rewarded state

3.2.2 Change in reward location

In the second experiment, the agent was mostly trained in the same environment as in the first experiment (Figure 3.1), however for the last few trials (the exact value is indicated in Appendix B) the location of the reward was changed. The arrow on Figure 3.2 indicates this update.

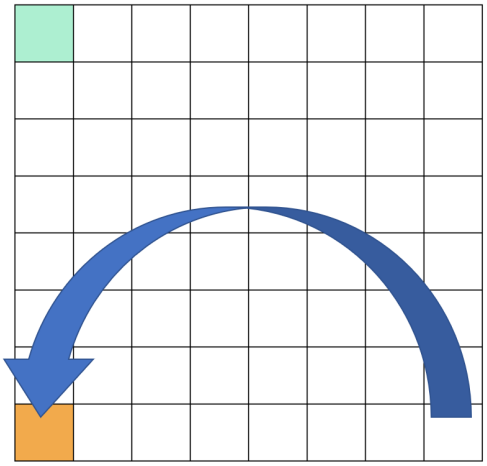

Figure 3.2: The environment in the experiment when the rewarded state changed. The blue arrow starts from the initial rewarded state and marks the new rewarding location

3.2.3 The labyrinth

In the last experiment, a layout inspired by [38] was constructed. The maze is shown on Figure 3.3. In this environment, the agent has to make four correct decisions at the appropriate decision-making cells. These states are marked in light blue. At each decision-making point, the agent has to choose between two actions (besides the action of turning back). The remaining possible actions are blocked either by obstacles marked in black or the borders of the environment.

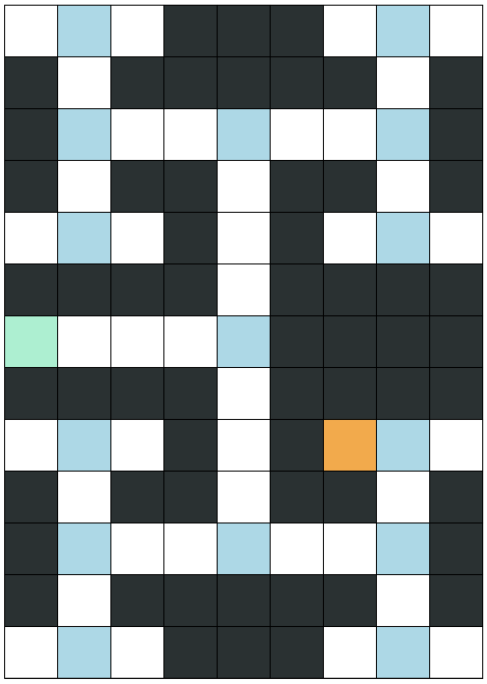

Figure 3.3: The labyrinth. Light blue cells mark the decision-making cells

3.3 Data collection and evaluation

The following three key attributes were used for comparison between the algorithms:

- Speed
- Reliability
- Flexibility

In case of speed and reliability tests, the agent was given a fixed set of trials to explore the environment in an episodic manner. This means that in each trial, as soon as it reached the rewarding state its location was reset to the starting state. Nonetheless, the agent was exposed to the environment more and more with each additional trial. For all trials, the agent was solely exploring, however at the end of each episode a greedy policy was computed from its experience, then an attempt to reach the reward was made. The outcome

(success or failure) of these attempts were monitored. The reason of this approach is that the richness of the learned representation over the trials was the target for the analysis. During the greedy attempt, the agent was not learning.

In the speed test, the first greedy attempt was registered, which lead to the reward in the minimum number of steps. In the reliability test, the ratio of greedy attempts leading to a reward over the total number of trials was computed. Both speed and reliability tests were executed in test suites, with each batch having the same number of fixed trials.

In case of the flexibility test, the robustness of the learned representation against perturbations was assessed. After the learning phase with a number of trials, a modified policy was extracted based on the learned value function. The value function kept fixed during the upcoming steps of this test, no additional learning was done. The modified policy means that for each test case, a ratio of the states were corrupted, such that for these states, the greedy action was blocked, and the action with the second largest value was selected instead of the largest value. Then, the same task as before was given to the agents for a number of occasions. However, for each instance, a different set of randomly selected states were perturbed. What this case illustrates is that in case of an unexpected limitation, can the agent shift to another trajectory which leads to the reward based on what it already experienced.

The exact value of all of the governing parameters in all experiments are indicated in Appendix B. An overview of the evaluation framework can be found in Table 3.1. The checkmarks indicate the executed tests.

Table 3.1: Overview of the evaluation framework. Checkmarks indicate executed tests

	Basic grid-world (Subsection 3.2.1)	**Change in reward (Subsection 3.2.2)**	**Labyrinth (Subsection 3.2.3)**
Speed	✓	✗	✓
Reliability	✓	✗	✓
Flexibility	✓	✓	✗

Chapter 4

The novel algorithm

4.1 Design principles

The novel algorithm focuses on learning a weight vector, which is used to parametrize the state-action value function, in contrast to the original SR which simply updates the reward estimation vector. The update rule of the successor representation matrix $\mathbf{M}$ is kept the same.

In all of the following equations, vectors are treated as column vectors. d marks the total number of state-action pairs, independent of whether an action can be taken from a given state or not.

The computation of the approximate state-action value function:

$$\hat{q}(s,a) = \mathbf{w}^\intercal \mathbf{M} \mathbf{x}(s,a) \tag{4.1}$$

Where:

- $\mathbf{w} \in \mathbb{R}^d$ is the weight vector,
- $\mathbf{M} \in \mathbb{R}^{d \times d}$ is the successor representation matrix,
- $\mathbf{x}(s,a)$ is a one-hot vector with length d.

As shown in Equation 4.1, the feature vector representing the state-action pair (s,a) is computed using $\mathbf{M}$ and $\mathbf{x}(s,a)$.

Based on Equation 4.1, the greedy action can be extracted by selecting the action with the largest value for a given state s:

$$a(s) = \underset{b \in \mathcal{A}}{\operatorname{argmax}}(\mathbf{w}^\intercal \mathbf{M} \mathbf{X}(s,b)) \tag{4.2}$$

Where:

- $\mathbf{X}(s, b)$ is a matrix composed of one-hot column vectors, b takes on all actions in the action space $\mathcal{A}$.

In Equation 4.2, the argmax operator returns an integer which is between 0 and the total number of actions minus one. This is convenient, since the different possible actions are assigned to integers starting with 0.

With respect to $\mathbf{w}$, the gradient of the state-action value function can be derived following the rules of multivariable calculus:

$$\nabla \hat{q}(s, a) = \mathbf{M}\mathbf{x}(s, a) \tag{4.3}$$

The discounted return, thus the true value of each state-action pair can be computed by:

$$q(s, a) = \gamma^t \tag{4.4}$$

Where:

- $\gamma \in \mathbb{R}$ is the discount factor (could take a different value from the one used to learn $\mathbf{M}$),
- $t \in \mathbb{N}_0$ is the number of steps to be made to reach a rewarding state-action pair. ($t = 0$ is included, it means that the next state is the terminal state.)

Equation 4.4 is valid, because all state-action pairs yield zero rewards, except for the ones which lead to the terminal state for which the reward is one. As an example for Equation 4.4, given a rewarding state-action pair $t = 0$, for state-action pairs which lead to a rewarding state-action pair $t = 1$, and so on. t can be obtained by the generated cognitive map, as described in Section 4.2.

The weight update upon obtaining a reward is done according to stochastic gradient-descent methods [1], with the equation:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha[q(s, a) - \hat{q}(s, a)]\nabla \hat{q}(s, a) \tag{4.5}$$

Where:

- $\alpha \in \mathbb{R}$ is the learning rate (could take a different value from the one used to learn $\mathbf{M}$).

In theory, Equation 4.5 ensures guaranteed convergence given the correct choice of α, however in practice fixed small values work well [1].

4.2 Implementation details

Each trial starts with the agent exploring the environment and learning the successor representation matrix. As soon as a reward is obtained, two functions are being executed consecutively. The first one is responsible for generating a cognitive map of the environment, the second one is responsible for updating the weight vector. The cognitive map should be thought of as a tree, where nodes represent state-action pairs and edges between the nodes are indicating reliable connections between the pairs, meaning that if from a given state-action pair, another state-action pair can be reached, then an edge shall connect the two nodes representing the two aforementioned pairs. An example of an instance of the cognitive map is shown on Figure 4.1.

The main input for generating the cognitive map is the successor representation matrix and the state-action pair which led to a reward. Using this information, an iterative process is being initiated. First the state-action pairs which could potentially lead to the recently rewarded state-action pair are extracted from the successor representation matrix. This can be done, since the main function of the matrix is to indicate that from a given state, taking a specific action, what is the most possible next state-action pair. Thus, given that it is known which state-action pair is desired, it is possible to find which state-action pairs lead to the rewarded state-action pair. This process is similar to the cognitive process of tracking back how an individual arrived to a certain state and what other states and actions could have lead to that specific state. The successor representation matrix registers all steps taken during the exploration phase, and their temporal connectivity, in contrast to solely focusing on the exact trajectory taken in the given trial. Thus it proves to be a compact structure for capturing the environment dynamics. The more the agent is exposed to the environment and the more it explores with each additional trial, the successor representation matrix is becoming more reliable.

After extracting the state-action pairs leading to the rewarding state-action pairs, these "neighbours" are added to the cognitive map. For all of the neighbours, the exact same procedure can be used to extract their neighbours. This process continues until all relevant nodes are added to the cognitive map. The map does not contain loops and each node is added only once. This design decision was made due to the fact that its main function is not to capture all possible trajectories, but to reliably provide the information of how many steps away one specific state-action pair is from the reward, so that the true value of the state-action pair could be computed via Equation 4.4.

On Figure 4.1 an illustration of the internal mechanism of the algorithm

is shown. The colours of the nodes indicate the reward prediction error, i. e. "$q(s,a) - \hat{q}(s,a)$" as in Equation 4.5. The numbers on each node are representing the unique ID of the state-action pairs. In order to ease the handling of state-action pairs in the code, every pair was assigned a unique ID. The figure was generated in a small grid-world in order to make it more transparent. In case of Figure 4.1, the state-action pair marked with "57" led to the reward. It takes place in the center of the graph. The initial value of that state-action pair is 0, however the true value is 1, thus the prediction error is high. Given the successor representation matrix $\mathbf{M}$, the neighbours of node "57" are "37" and "43". Their true values are reduced by the discount factor when compared to the rewarding state-action pair, but are still large, thus the prediction error is still significant. Utilizing the same process, first their neighbours, then the true values of their neighbours can be computed. This sequence continues until all relevant nodes are added to the graph. The lengths of the edges have no significance and solely depend on the used visualization libraries.

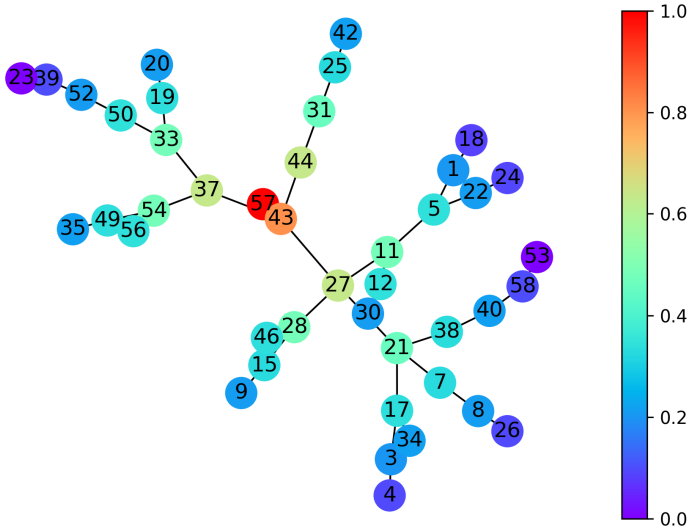

Figure 4.1: The cognitive map. Numbers indicate ID-s of specific state-action pairs, the colour encodes the reward prediction error

The update of the weight vector is based on the previously described cognitive map and the hyperparameters indicated in the equations specified in Section 4.1. For each node on the map, the weight vector is being updated according to Equation 4.5.

The psudocode can be found in the Appendix A and the complete code

implemented in Python is available online at [9].

4.3 Neural correlates of the wave-like update rule

There is evidence, that under certain conditions, wave-like dopamine waves can be observed in the dorsal striatum [3]. One particular pattern of these waves can be described by center-out motion properties as shown on Figure 4.2. (More details can be found in Chapters 1 and 2.)

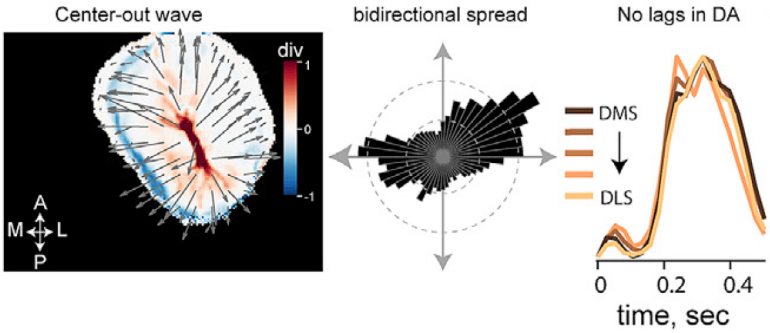

Figure 4.2: Center-out dopamine waves in the dorsal striatum [3]. The dopaine activity exhibits no delay in this particular pattern

As explained in Section 2.2, there is significant evidence that the activity of dopamine neurons signal a quantity very similar to reward prediction errors. Having this idea in mind and comparing figures 4.1 and 4.2 a striking realization can be made. In case the neuronal populations in the dorsal striatum are organized in space according to the cognitive map, which means that spatially adjacent populations represent state-action pairs which are adjacent on the cognitive map as well, then the function of the center-out dopamine waves in the dorsal striatum could be to spread information about reward prediction errors to targeted subregions. The center-out waves do not exhibit any temporal delays, neither does the update rule previously described in Sections 4.1 and 4.2.

Chapter 5

Results

In order to evaluate the strengths and weaknesses of the novel algorithm, the following three classical RL algorithms were chosen for comparison given the same task:

- Classical SARSA
- Original SR
- SARSA with eligibility trace, referred to as SARSA(λ)

These algorithms are not only among the most widely used solutions for many RL problems, but their general structure is the same as the internal mechanism of the new algorithm introduced in Chapter 4, the SR with wave-like update rule. All algorithms follow the SARSA process, which means that given a state (S) and an action (A), the reward (R) is obtained, the next state is the result of the action (S'), finally a next action (A') is selected. Using this sequence, abbreviated to SARSA, the agent carries out internal computations. In addition, all of them use state-action pairs, thus are suitable candidates for highlighting the strengths and weaknesses of the novel algorithm. The four algorithms were given the problem of the three experiments described in Chapter 3. All parameters used to generate the data can be found in Appendix B.

5.1 The simple grid-world

In the first experiment, the aim was to have a general understanding of the new algorithm's capabilities. During the development phase, this simple environment was scaled down to have a size of 4×4, thus was suitable for rapid prototyping.

The speed test of the four algorithms in the simple grid-world experiment is shown on Figure 5.1. It is clear that on average, the novel algorithm is able to find the location of the reward in less trials when compared to the classical RL approaches. The violin plot representation of the data provides a further insight, since it attempts to capture the underlying distribution of the input. As presented, the distribution of the wave-like SR has the smallest variance, thus the algorithm is not only quick in finding the reward, but consistent in this attribute throughout the test batches.

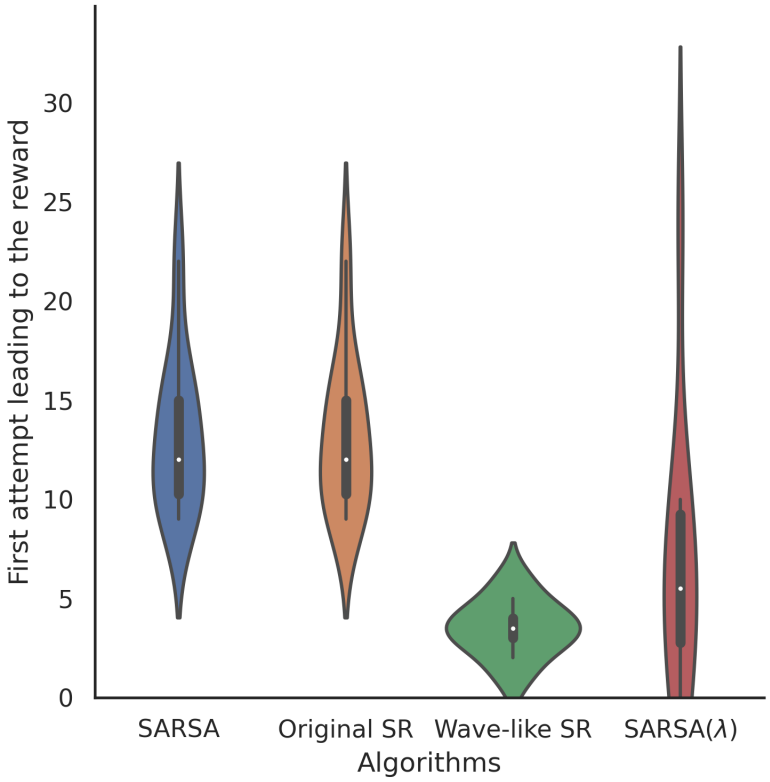

Figure 5.1: The results of the speed test in the simple grid-world experiment. Colors indicate different algorithms

The reliability test of the four algorithms in the simple grid-world experiment is shown on Figure 5.2. As shown, the SR with wave-like update rule tends to be more focused. This means that once it found the reward for the first time during exploration, there is a high chance that it does not loose the target out of sight and finds an optimal trajectory which leads back to the correct destination in the upcoming trials.

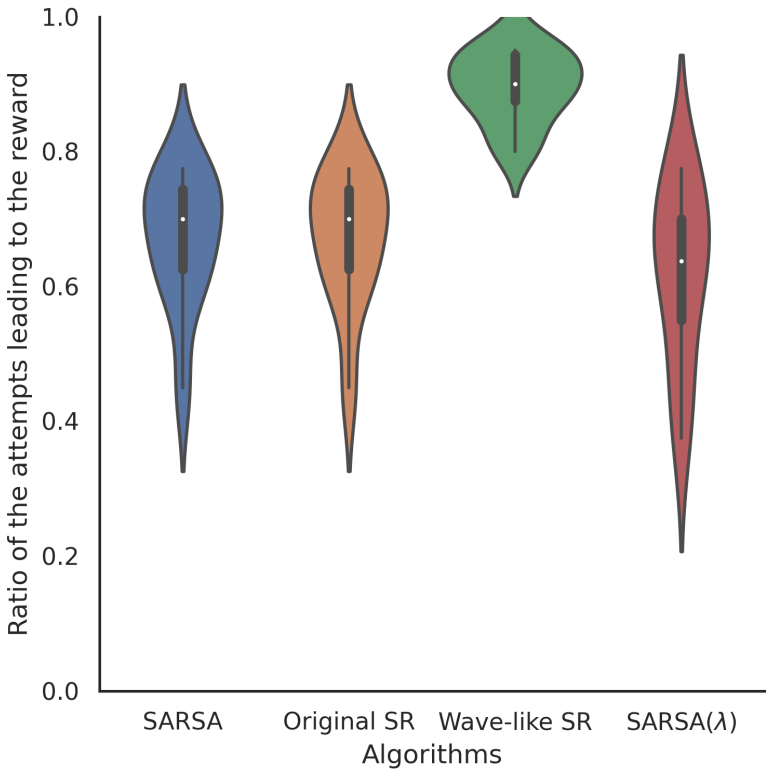

Figure 5.2: The results of the reliability test in the simple grid-world experiment. Color code is same as on Figure 5.1

The flexibility test of the four algorithms in the simple grid-world experiment is shown on Figure 5.3. The figure highlights that the novel algorithm not only learns quickly, but learns a rich representation which is robust against perturbations. Its best competitor in swiftness (according to Figure 5.1), the SARSA(λ) algorithm does not perform well when the action selection is constrained. To give a clear example, from a batch of layouts when 20 % of the states were corrupted, only less than half of the total attempts lead to finding the reward for the SARSA(λ) algorithm, while the wave-like SR algorithm managed to find the reward in most cases (with a ratio around 0.8) for the same conditions. The perturbed states were randomly selected in case of each element of the batch.

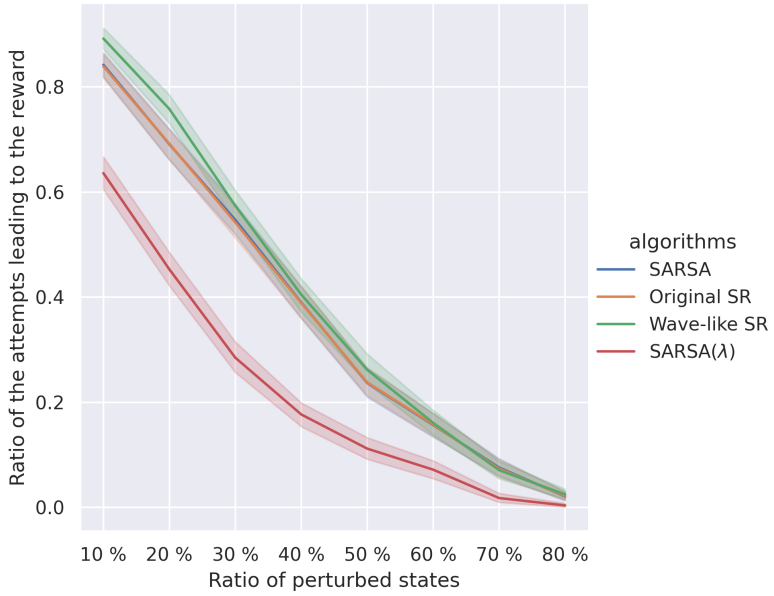

Figure 5.3: The results of the flexibility test in the simple grid-world experiment. Color code is same as on Figure 5.1

5.2 Change in reward location

After it became clear that the new algorithm has a decent performance in all domains of comparison, the next step was to see how it performs in a dynamic environment where the location of the reward is not fixed throughout the trials of one test batch. (More details of the experiment can be found in Subsection 3.2.2.)

In this task, only the flexibility test was executed, since the aim was to observe how these agents could adapt to changing environments, speed and reliability was out of scope. The result is shown on Figure 5.4. It is clear that the original SARSA and the SARSA(λ) algorithms struggle with finding the reward when introducing perturbations, however the SR algorithms are able to utilize their previous experience of the environment dynamics in case the location of reward has changed.

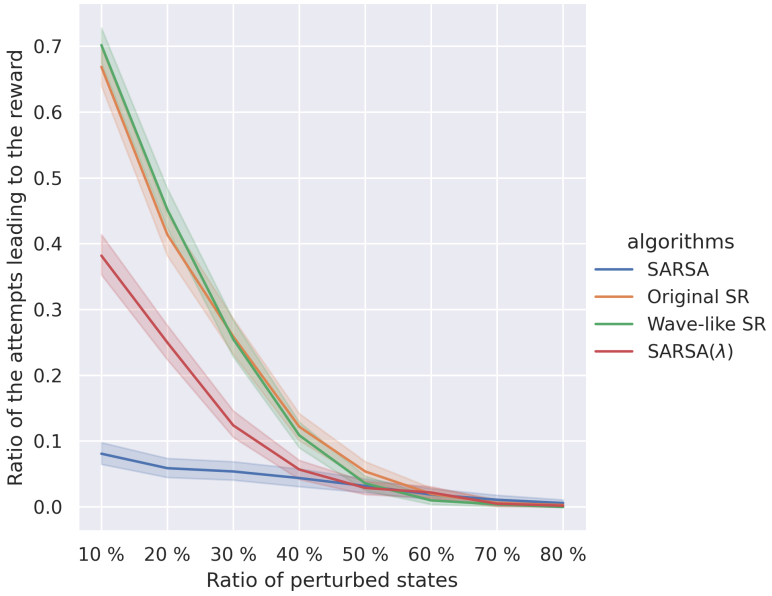

Figure 5.4: The results of the flexibility test in the "change in reward location" experiment. Color code is same as on Figure 5.1

5.3 The labyrinth

The next challenge given to the learning agents was to introduce obstacles to the environment. A maze was constructed, the guiding principles of the layout can be found in Subsection 3.2.3. The length of the optimal trajectory has the same number of steps as in the simple grid-world described in Subsection 3.2.1.

No flexibility test was executed in this setup, since the environment is already very limited, thus perturbations would have caused an almost impossible challenge.

The speed test of the four algorithms in the labyrinth is shown on Figure 5.5. Despite the introduction of obstacles, the new algorithm remained to outperform its competitors in the average number of trials necessary to find the rewarded state. This is achieved with low variance as seen previously in the case of the simple grid-world.

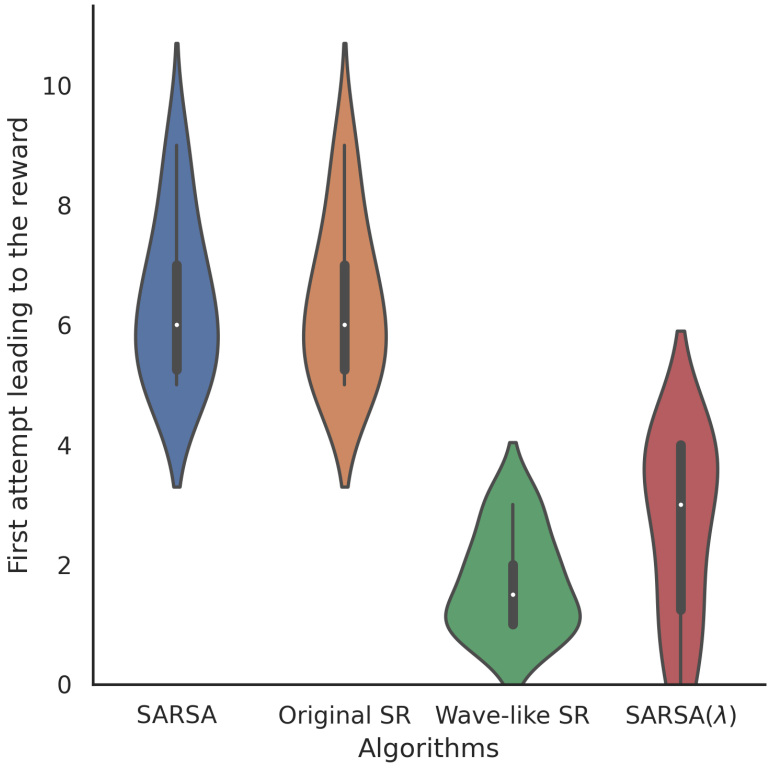

Figure 5.5: The results of the speed test in the labyrinth. Color code is same as on Figure 5.1

The reliability test of the four algorithms in the labyrinth is shown on Figure 5.6, where a disadvantage of the new algorithm is presented. Its reliability does not seem to be independent from the layout of the environment: the variance is much larger in the labyrinth (Figure 5.6), than in the basic grid-world (Figure 5.3). Interestingly, based on the comparison of Figures 5.1 and 5.5, it can be declared that its speed does not expose similar limitations.

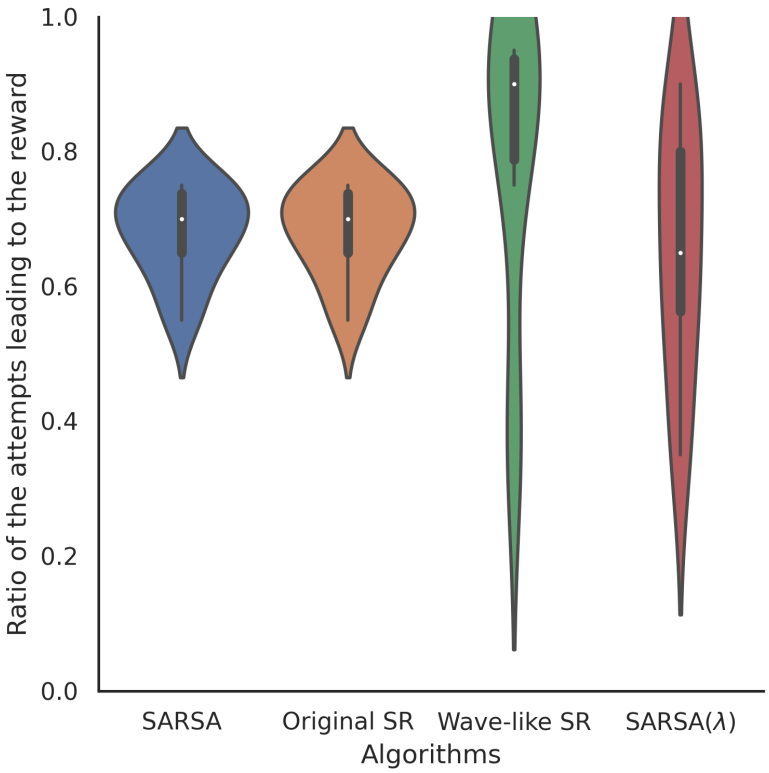

Figure 5.6: The results of the reliability test in the labyrinth. Color code is same as on Figure 5.1

Chapter 6

Discussion

6.1 Conclusion

In this work, a novel Reinforcement Learning algorithm was presented. The core mechanism of the method is the construction of a cognitive map using previous experience, and the computation of reward prediction errors based on this structure. The shown process does not only outperform classical RL approaches, but provides a possible explanation for the function of spatiotemporal dopamine waves in the dorsal striatum. The parameterization of the value function of the SR algorithm proved to be beneficial given the introduced wave-like update rule.

The initially defined subgoals in Section 1.3 served as a proper roadmap for conducting the research, however some items were carried out in parallel. The literature study was pursued until the novel algorithm got its final form. A virtual simulation environment was constructed which was suitable for carrying out experiments in a RL setting. Then, first the new algorithm, then the classical RL algorithms were implemented. As a final step, tests were designed and executed for benchmarking.

In conclusion, results obtained from simulations does show advantages of the new approach over classical RL algorithms, however further investigation of dopamine transmission on a subcircuit level should be carried out in order to have a better understanding of exact function of the wave-like dynamics of dopamine. Further analysis would not only lead to a deeper understanding of the biological phenomena, but might inspire original algorithms taking steps towards neuroscience-inspired AGI.

6.2 Limitations

The algorithm presented in Chapter 4 has certain constraints. First of all, it utilizes two additional hyperparameters, both of which currently need to be tuned, for example by searching the parameter space given a specific task.

Although it is not limited to grid-world environments, in order for the agent to successfully solve the task in limited time, the environment must fulfill a couple of requirements. Firstly, the problem formulation must be episodic, there should be a terminal state, such that when it is reached, the current episode should end and the state of the agent should be reset. Secondly, both state space $\mathcal{S}$ and action space $\mathcal{A}$ must be discretized with the number of distinct elements being relatively low, or in general aligned with the available computational resources. Thirdly, the agent may only receive zero reward unless it reaches the terminal state, which shall be awarded by one unit of reward. Finally, the agent is relatively successful in solving deterministic tasks, however in case the environment is stochastic, the novel algorithm needs to be modified in order to handle uncertainty.

6.3 Future work

First of all, the constraints described in Section 6.2 lead to ideas for improvement. Adding an extra layer to the cognitive map which utilizes a probabilistic relationship between state-action pairs would extend the capabilities of the algorithm to successfully solve tasks in a stochastic environment. In addition, modifying Equation 4.4 could lead to complete problems where reward is obtained more often, not only in the terminal state.

A natural next step would be to conduct a systems neuroscience experiment to verify and further explore how the cognitive map described in Chapter 4 could be the organizing principle of wave-like activity. The experiment should be aimed to answer these questions:

1. Do striatal populations represent state-action pairs?

2. Does the spatial organization of striatal populations share similarities with the graph generated by the novel algorithm?

A sketch of the experiment could be the following. The general setup would be a freely moving animal in a maze, similar to a grid-world problem. The striatal activity and dopamine concentration of the animal should be measured throughout the study.

The environment should have discrete cells, which are larger than the body of the animal. Each cell, which is not considered as an obstacle (thus being blocked from the animal), should have a tone with a discrete frequency, such that it is indicated to the animal that each location represents a separate state. The animal will start moving around looking for a reward, which could be a food pellet or fruit juice. A camera should monitor the motion of the animal from a top-down perspective. This way the complete trajectory can be observed and actions (leading to transitions from one state to another) could be registered. As soon as the animal reaches the reward, a cognitive map according to the novel algorithm can be constructed. The theoretical map and actual brain activity can be compared by visualizing the striatal activity after obtaining the reward via calcium imaging. The comparision might not be straitforward, since the cognitive map is essentially a graph and the way of its visualization is constrained, on contrary to the spatially continuous dopamine waves. A tangential, but interesting analysis would be to observe how the cortex signals striatal populations about the different state-action pairs. Or if it does not, a novel hypothesis could be formalized based on the observations. It is important to mention that an initially completely naive animal shall be given the task, and the progress towards learning the optimal trajectory leading to the reward shall be in focus of this experiment. Furthermore, the temporal and the spatial resolution of the data should be sufficient, and large enough part of the striatum shall be observed.

There are some further issues that could be addressed in future work based on the observation of dopamine waves [3]. An extension to the algorithm could be to investigate how the update rule can be adjusted in a way that the waves become directional, corresponding to the latero-medial and medio-lateral pattern of the waves. In addition, the temporal properties shall be also incorporated. These ideas can open up interesting paths of future developments.

Finally, a key question still remains. Most RL algorithms appear to be highly sensitive to hyperparameters. These values cannot be neglected, and have to be set with care. Given that the central nervous system might implement a reinforcement learning algorithm, what could be the underlying biological process responsible for setting and fine-tuning these parameters? This still is an open question in the research community.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*, 2nd ed., ser. Adaptive computation and machine learning series. The MIT Press, 2018. ISBN 9780262039246

[2] W. Schultz, P. Dayan, and P. R. Montague, "A neural substrate of prediction and reward," *Science*, vol. 275, no. 5306, pp. 1593–1599, Mar. 1997. doi: 10.1126/science.275.5306.1593. [Online]. Available: https://doi.org/10.1126/science.275.5306.1593

[3] A. A. Hamid, M. J. Frank, and C. I. Moore, "Wave-like dopamine dynamics as a mechanism for spatiotemporal credit assignment," *Cell*, 2021. doi: https://doi.org/10.1016/j.cell.2021.03.046. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0092867421003779

[4] M. Botvinick, J. X. Wang, W. Dabney, K. J. Miller, and Z. Kurth-Nelson, "Deep Reinforcement Learning and Its Neuroscientific Implications," *Neuron*, vol. 107, no. 4, pp. 603 – 616, 2020. doi: https://doi.org/10.1016/j.neuron.2020.06.014

[5] J. Cox and I. B. Witten, "Striatal circuits for reward learning and decision-making," *Nature Reviews Neuroscience*, vol. 20, no. 8, pp. 482–494, Jun. 2019. doi: 10.1038/s41583-019-0189-2. [Online]. Available: https://doi.org/10.1038/s41583-019-0189-2

[6] H. R. Kim, A. N. Malik, J. G. Mikhael, P. Bech, I. Tsutsui-Kimura, F. Sun, Y. Zhang, Y. Li, M. Watabe-Uchida, S. J. Gershman, and N. Uchida, "A unified framework for dopamine signals across timescales," *Cell*, vol. 183, no. 6, pp. 1600–1616.e25, Dec. 2020. doi: 10.1016/j.cell.2020.11.013. [Online]. Available: https://doi.org/10.1016/j.cell.2020.11.013

[7] P. Dayan, “Improving generalization for temporal difference learning: The successor representation,” *Neural Computation*, vol. 5, no. 4, pp. 613–624, Jul. 1993. doi: 10.1162/neco.1993.5.4.613. [Online]. Available: https://doi.org/10.1162/neco.1993.5.4.613

[8] United Nations. Transforming our world: the 2030 agenda for sustainable development. [Online]. Available: https://sdgs.un.org/2030agenda

[9] G. Gömöri, “Successor representation with a wave-like update rule,” Jun. 2021. [Online]. Available: https://github.com/gergogomori/wave-like-sr-algorithm

[10] R. S. Sutton, “Learning to predict by the methods of temporal differences,” *Mach. Learn.*, vol. 3, no. 1, p. 9–44, Aug. 1988. doi: 10.1023/A:1022633531479. [Online]. Available: https://doi.org/10.1023/A:1022633531479

[11] S. J. Gershman, “The successor representation: Its computational logic and neural substrates,” *Journal of Neuroscience*, vol. 38, no. 33, pp. 7193–7200, 2018. doi: 10.1523/JNEUROSCI.0151-18.2018

[12] I. Momennejad, E. M. Russek, J. H. Cheong, M. M. Botvinick, N. D. Daw, and S. J. Gershman, “The successor representation in human reinforcement learning,” *Nature Human Behaviour*, vol. 1, no. 9, pp. 680–692, Aug. 2017. doi: 10.1038/s41562-017-0180-8. [Online]. Available: https://doi.org/10.1038/s41562-017-0180-8

[13] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016. doi: 10.1038/nature16961. [Online]. Available: https://doi.org/10.1038/nature16961

[14] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, Z. D. Guo, and C. Blundell, “Agent57: Outperforming the Atari human benchmark,” in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 507–517. [Online]. Available: http://proceedings.mlr.press/v119/badia20a.html

[15] M. G. Bellemare, S. Candido, P. S. Castro, J. Gong, M. C. Machado, S. Moitra, S. S. Ponda, and Z. Wang, "Autonomous navigation of stratospheric balloons using reinforcement learning," *Nature*, vol. 588, no. 7836, pp. 77–82, Dec. 2020. doi: 10.1038/s41586-020-2939-8. [Online]. Available: https://doi.org/10.1038/s41586-020-2939-8

[16] D. Silver, S. Singh, D. Precup, and R. S. Sutton, "Reward is enough," *Artificial Intelligence*, p. 103535, May 2021. doi: 10.1016/j.artint.2021.103535. [Online]. Available: https://doi.org/10.1016/j.artint.2021.103535

[17] J. Olds and P. Milner, "Positive reinforcement produced by electrical stimulation of septal area and other regions of rat brain." *Journal of Comparative and Physiological Psychology*, vol. 47, no. 6, pp. 419–427, 1954. doi: 10.1037/h0058775. [Online]. Available: https://doi.org/10.1037/h0058775

[18] C. Liu, P. Goel, and P. S. Kaeser, "Spatial and temporal scales of dopamine transmission," *Nature Reviews Neuroscience*, Apr. 2021. doi: 10.1038/s41583-021-00455-7. [Online]. Available: https://doi.org/10.1038/s41583-021-00455-7

[19] B. Engelhard, J. Finkelstein, J. Cox, W. Fleming, H. J. Jang, S. Ornelas, S. A. Koay, S. Y. Thiberge, N. D. Daw, D. W. Tank, and I. B. Witten, "Specialized coding of sensory, motor and cognitive variables in VTA dopamine neurons," *Nature*, vol. 570, no. 7762, pp. 509–513, May 2019. doi: 10.1038/s41586-019-1261-9. [Online]. Available: https://doi.org/10.1038/s41586-019-1261-9

[20] W. Dabney, Z. Kurth-Nelson, N. Uchida, C. K. Starkweather, D. Hassabis, R. Munos, and M. Botvinick, "A distributional code for value in dopamine-based reinforcement learning," *Nature*, vol. 577, no. 7792, pp. 671–675, Jan. 2020. doi: 10.1038/s41586-019-1924-6. [Online]. Available: https://doi.org/10.1038/s41586-019-1924-6

[21] S. Yagishita, A. Hayashi-Takagi, G. C. R. Ellis-Davies, H. Urakubo, S. Ishii, and H. Kasai, "A critical time window for dopamine actions on the structural plasticity of dendritic spines," *Science*, vol. 345, no. 6204, pp. 1616–1620, Sep. 2014. doi: 10.1126/science.1255514. [Online]. Available: https://doi.org/10.1126/science.1255514

[22] H. C. Blodgett, *The effect of the introduction of reward upon the maze performance of rats*, ser. University of California publications in psychology, v. 4, no. 8. Berkeley: Univ. of California Press, 1929.

[23] E. C. Tolman, "Cognitive maps in rats and men," *Psychological Review*, vol. 55, no. 4, pp. 189–208, 1948. doi: 10.1037/h0061626. [Online]. Available: https://doi.org/10.1037/h0061626

[24] K. L. Stachenfeld, M. M. Botvinick, and S. J. Gershman, "The hippocampus as a predictive map," *Nature Neuroscience*, vol. 20, no. 11, pp. 1643–1653, Oct. 2017. doi: 10.1038/nn.4650. [Online]. Available: https://doi.org/10.1038/nn.4650

[25] J. C. Whittington, T. H. Muller, S. Mark, G. Chen, C. Barry, N. Burgess, and T. E. Behrens, "The tolman-eichenbaum machine: Unifying space and relational memory through generalization in the hippocampal formation," *Cell*, vol. 183, no. 5, pp. 1249–1263.e23, Nov. 2020. doi: 10.1016/j.cell.2020.10.024. [Online]. Available: https://doi.org/10.1016/j.cell.2020.10.024

[26] Y. Liu, M. G. Mattar, T. E. J. Behrens, N. D. Daw, and R. J. Dolan, "Experience replay is associated with efficient nonlocal learning," *Science*, vol. 372, no. 6544, p. eabf1357, May 2021. doi: 10.1126/science.abf1357. [Online]. Available: https://doi.org/10.1126/science.abf1357

[27] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. JMLR.org, 2017, p. 449–458.

[28] W. Dabney, M. Rowland, M. G. Bellemare, and R. Munos, "Distributional reinforcement learning with quantile regression," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[29] A. S. Lowet, Q. Zheng, S. Matias, J. Drugowitsch, and N. Uchida, "Distributional reinforcement learning in the brain," *Trends in Neurosciences*, vol. 43, no. 12, pp. 980–997, Dec. 2020. doi: 10.1016/j.tins.2020.09.004. [Online]. Available: https://doi.org/10.1016/j.tins.2020.09.004

[30] M. P. H. Gardner, G. Schoenbaum, and S. J. Gershman, "Rethinking dopamine as generalized prediction error," *Proceedings of the Royal*

Society B: Biological Sciences, vol. 285, no. 1891, p. 20181645, Nov. 2018. doi: 10.1098/rspb.2018.1645. [Online]. Available: https://doi.org/10.1098/rspb.2018.1645

[31] E. M. Russek, I. Momennejad, M. M. Botvinick, S. J. Gershman, and N. D. Daw, "Predictive representations can link model-based reinforcement learning to model-free mechanisms," *PLOS Computational Biology*, vol. 13, no. 9, p. e1005768, Sep. 2017. doi: 10.1371/journal.pcbi.1005768. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1005768

[32] J. P. Geerts, F. Chersi, K. L. Stachenfeld, and N. Burgess, "A general model of hippocampal and dorsal striatal learning and decision making," *Proceedings of the National Academy of Sciences*, vol. 117, no. 49, pp. 31 427–31 437, 2020. doi: 10.1073/pnas.2007981117. [Online]. Available: https://www.pnas.org/content/117/49/31427

[33] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. doi: 10.1038/s41586-020-2649-2. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[34] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. doi: 10.1109/MCSE.2007.55

[35] J. Reback, W. McKinney, jbrockmendel, J. V. den Bossche, T. Augspurger, P. Cloud, S. Hawkins, gfyoung, Sinhrks, M. Roeschke, A. Klein, T. Petersen, J. Tratner, C. She, W. Ayd, S. Naveh, patrick, M. Garcia, J. Schendel, A. Hayden, D. Saxton, V. Jancauskas, M. Gorelli, R. Shadrach, A. McMaster, P. Battiston, S. Seabold, K. Dong, chris b1, and h vetinari, "pandas-dev/pandas: Pandas 1.2.4," Apr. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4681666

[36] M. L. Waskom, "seaborn: statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021. doi: 10.21105/joss.03021. [Online]. Available: https://doi.org/10.21105/joss.03021

[37] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11 – 15.

[38] M. Rosenberg, T. Zhang, P. Perona, and M. Meister, "Mice in a labyrinth: Rapid learning, sudden insight, and efficient exploration," *bioRxiv*, 2021. doi: 10.1101/2021.01.14.426746. [Online]. Available: https://www.biorxiv.org/content/early/2021/01/15/2021.01.14.426746

Appendix A

Pseudocode

Algorithm 1: Successor Representation with wave-like update rule

Result: $\mathbf{M}$, $\mathbf{w}$

$\mathbf{M} = \mathbf{0}$
$\mathbf{w} = \mathbf{0}$
set α
set γ
set α_w
set γ_w
for *each episode* **do**
    initialize s
    choose a according to an ε-greedy policy
    while s *is not terminal* **do**
        Take action a, obtain r and observe s'
        choose a' according to an ε-greedy policy
        $\delta \leftarrow \mathbf{1}_{sa} + \gamma \mathbf{M}(s'a', :) - \mathbf{M}(sa, :)$
        $\mathbf{M}(sa, :) \leftarrow \mathbf{M}(sa, :) + \alpha\delta$
        if s' *is terminal* **then**
            map $\leftarrow$ GENERATE MAP($\mathbf{M}, s, a$)
            $\mathbf{w} \leftarrow$ UPDATE WEIGHTS(map, $\mathbf{M}, \mathbf{w}, \alpha_w, \gamma_w$)
        end
        $s \leftarrow s'$
        $a \leftarrow a'$
    end
end

In the algorithm, $\mathbf{1}_{sa}$ is a one-hot column vector, same as the feature vector $\mathbf{x}(s, a)$ introduced in Section 4.1. $\mathbf{M}(sa, :)$ is the row of $\mathbf{M}$, indicated by s and a. This row estimates the time expected to be spent in each other state-action pair given that the agent is in s and takes the action a.

Algorithm 2: Function for generating the cognitive map

```
Result: map
add the node of the rewarding sa to the buffer
add the node of the rewarding sa the map
while buffer is not empty do
    find the neighbours of the node based on M
    for each neighbour do
        if neighbour is not on the map then
            add neighbour to the map
            add neighbour to the buffer
        end
    end
    delete first element of the buffer
end
```

Algorithm 3: Function for updating $\mathbf{w}$

```
Result: w
for each node on the map do
    compute its one-hot column vector x(s, a)
    compute the approximate state-action value q̂(s, a) with Equation
     4.1
    compute the gradient of the state-action value function ∇q̂(s, a)
     with Equation 4.3
    compute the true value q(s, a) with Equation 4.4
    update w according to Equation 4.5
end
```

Appendix B

Parameters

The parameters of the data generation process can be divided into two main domains: experiment related parameters and algorithm hyperparameters.

The hyperparameters of each algorithm could be categorized into two additional classes: base hyperprameters which are used by all of the algorithms and specific hyperparameters, which are utilized by only one of the RL methods. Except for the wave-like update rule parameters, all values were set based on previous experience and taking the accessible computational resources into consideration. The wave-like update rule parameters were set using simple hyperparameter search scripts and experience gained during the project.

The following three sections are aligned with the experiments introduced in Chapter 3.2.

B.1 The simple grid-world

- Experiment related parameters
  - Number of test batches: 10
  - Number of different perturbations for each test batch: 100
  - Number of trials in each batch: 40
- Algorithm related hyperparameters
  - Base hyperparameters
    - Learning rate (α): 0.05
    - Discount factor (γ): 0.99
    - Exploration ratio (ε): 1.0

    - Specific hyperparameters
        - Learning rate for the wave-like update rule (α_w): 0.5
        - Discount factor for the wave-like update rule (γ_w): 0.5
        - Trace decay of the eligibility trace (λ): 0.9

B.2 Change in reward location

- Experiment related parameters
    - Number of test batches: 10
    - Number of different perturbations for each test batch: 100
    - Number of trials in each batch with the default location of the reward: 35
    - Number of trials in each batch with the new location of the reward: 5
- Algorithm related hyperparameters
    - Base hyperparameters
        - Learning rate (α): 0.05
        - Discount factor (γ): 0.99
        - Exploration ratio (ε): 1.0
    - Specific hyperparameters
        - Learning rate for the wave-like update rule (α_w): 0.5
        - Discount factor for the wave-like update rule (γ_w): 0.5
        - Trace decay of the eligibility trace (λ): 0.9

B.3 The labyrinth

- Experiment related parameters
    - Number of test batches: 10
    - Number of trials in each batch: 20
- Algorithm related hyperparameters
    - Base hyperparameters

    - Learning rate (α): 0.05
    - Discount factor (γ): 0.99
    - Exploration ratio (ε): 1.0
- Specific hyperparameters
    - Learning rate for the wave-like update rule (α_w): 0.5
    - Discount factor for the wave-like update rule (γ_w): 0.6
    - Trace decay of the eligibility trace (λ): 0.9