Reinforcement learning in partially observable tasks: state uncertainty and memory dependence

Adnane Ez-zizi

A dissertation submitted to the University of Bristol in accordance with the requirements for award of the degree of Doctor of Philosophy in the Faculty of Science.

School of Experimental psychology

June 2016

word count: 47150

Abstract

Reinforcement learning models have been successfully used to explain a wide range of psychological and neural features of human and animal learning behaviour. The most common reinforcement learning models were initially designed to work in Markovian environments where the currently observed input gives complete knowledge about the current state of the system, and where rewards, optimal responses and transitions between states are all dependent on the present state only. However, these conditions are rarely satisfied in the real problems that humans or animals face. In this thesis, we examine how people can deal or actually deal with tasks where these conditions are not met; a problem that is often referred to as the problem of partial observability. We consider two types of partial observability: (1) that which is due to state uncertainty, and (2) that which is due to the dependency of some states on past information, and hence requires some sort of memory. In two separate experimental studies, we show that people are, in general, able to deal with both these types of partial observability when the tasks involved are simple. In a series of modelling studies, we introduce a number of modifications to the Actor-Critic gating model, one of the seminal models for solving memory-dependent partially observable tasks, which substantially improved its learning speed and plausibility as a model of human learning. Our modelling work also generates a number of predictions, which can be used to guide future experimental investigations.

Acknowledgements

This thesis is dedicated to my parents, who have constantly propelled me to excel in my studies and supported me throughout my life. I also have a special thought for my wife; many thanks for being such a kind and supportive wife since we have first met.

I am immensely grateful to my principal PhD advisors, David Leslie and Simon Farrell for the skills they taught me during my PhD, academic and beyond. I am very proud to have had the chance to work with you. David, thank you for introducing me to the field of reinforcement learning and for always pushing me to improve my critical thinking and to focus on specific problems rather than wandering around different problems to solve. I was always amazed by your ability to quickly understand the most complex interactions even when I missed in clarity. Simon, thank you for introducing me to the field of working memory. I have enjoyed enormously discussing research with you, and have learned a lot from your ability to ask good questions and to be open to embrace others' perspectives. Thank you also for your hospitality during my visit to the University of Western Australia.

Many thanks go to Roland Baddeley for kindly taking over my official supervision after both Simon and David have moved from Bristol. I must also express my profound gratitude to Edward Cripps for hosting me at the University of Western Australia's school of statistics. I learned a lot from his vast expertise in Bayesian modelling and his ability to break down complex problems into smaller, more manageable subproblems. Our collaboration was particularly valuable in gaining insights into the data presented in the second chapter of my thesis.

I would also like to thank all the other members of the decision making group for making it such a great place to work. In particular, many thanks are due to the other PhD students and postdocs for being supportive and willing to discuss ideas and share knowledge. A special thanks goes also to Iain Gilchrist for being such an effective leader and for always trying to make our experience at the decision making group productive and enjoyable.

Finally, I would like to thank the administrative staff at the decision making group and at the school of Experimental Psychology for providing quick and effective support in any day to day needs.

My PhD research has been funded by the UK Engineering and Physical Sciences Research Council via the decision making project grant EP/I032622/1.

Declaration

Contents

List of Figures

List of Tables

Chapter 1

General introduction

Whether we are learning basic tasks such as how to walk or how to ride a bicycle, or more complex activities such as which series of moves to make in a chess game in order to win, we are constantly learning by interacting with our environment. Reinforcement learning (RL) provides a set of techniques to solve the learning problem of an agent—human, animal or robot—that aims to maximise its long-term reward in an unknown environment. The agent (e.g., a chess player) is assumed to select actions (moving the chess pieces) depending on the current state of the environment (the positions of the pieces on the chessboard), and depending on the state of the environment these actions may produce reward feedback (checkmating the opponent's king). The reward feedback can be used by the agent to learn which actions in which states will maximise long-run reward (Sutton and Barto, 1998).

The classical models of RL assume that the currently observed input gives complete knowledge about the current state of the system, and that rewards, optimal responses and transitions between states are all dependent on the present state only. However, these conditions are rarely satisfied in the real problems that humans, animals or artificial machines face. This thesis is concerned, to a large extent, with how people could deal or actually deal with tasks where these conditions are not met; a problem that is often referred to in machine learning as the problem of partial observability. We will consider two types of partial observability: that which is due to state uncertainty, and the one due to the dependency of the state on past information, and hence requires some sort of memory. We will particularly focus on the latter case since in many problems, as we shall illustrate in this chapter, resolving state uncertainty can be achieved through the use of memory.

There is a vast literature on learning in partially observed environments within the machine learning community (see for example reviews in White, 1991; Kaelbling et al., 1998; Hasinoff, 2002). However, despite the long history of behavioural investigations of learning (e.g., Thorndike, 1911), models of RL in partially observable tasks have had remarkably little impact in the psychology community. Initial advances have been made by Zilli and Hasselmo (2008), Todd et al. (2009) and Larsen et al. (2010), and this thesis builds from these key papers.

A possible reason for this behindhand interest in psychology is the slow learning results often reported in modelling works, which promote the belief that testing partially observable tasks in the lab would be infeasible (see however Krueger, 2011, in which it is demonstrated that humans can outperform standard machine learning algorithms in one such a task). A first step to bring these tasks into the lab, for now, has been to convert machine learning algorithms into psychologically (mostly by reframing them into psychological terms) and neurally plausible models. This thesis pursues what we believe is the next step to understand human RL in partially observable environments, which is to build fast, yet plausible models for how people learn in these domains.

This chapter, as well as reviewing the literature related to our topic, gives the necessary background for the rest of the thesis. We start the chapter by giving an historical overview of RL in general. This is to justify the need for an interdisciplinary approach to study the subtopic of partial observability. We then define the theoretical framework for the study of RL problems in general. After that, we introduce the problem of partial observability in RL in both the forms considered in this thesis: state uncertainty and past-dependent state, and review the related literature from machine learning, psychology and neuroscience perspectives. Here, and in the whole thesis, we only treat model-free RL methods. Other methods such as model-based RL (Daw et al., 2005; Rangel et al., 2008) or evolutionary algorithms (Moriarty et al., 1999)—although also interesting—are not broached, or at most, discussed briefly. Finally, we conclude by summarising the key contributions of major studies to our topic and by justifying our research proposal.

1.1 Reinforcement learning

Reinforcement learning (RL) is an interdisciplinary field that draws upon knowledge from psychology, computer science, optimal control, and more recently neuroscience. In the following, we will give a brief overview of how each of these fields has contributed to the study of RL. For more detailed reviews, see Sutton and Barto (1998) (general perspective), Niv (2009), Walsh and Anderson (2014) (psychology and neuroscience perspectives) or Kaelbling et al. (1996) (computer science perspective).

1.1.1 A brief historical overview

The key ideas of RL have their roots in psychology. Edward Thorndike (1911) was one of the first scientists to study the idea of trial-and-error learning using his famous puzzle box. Thorndike placed cats and chickens inside a cage, which could only be opened by operating latches. He noticed that his research animals did not necessary show flushes of insights in order to solve the problem, instead they were engaged in a process of trial-and error, initially trying random actions, then repeating those actions that led them to exit the box and get their reward. Thorndike's experiment is an instance of what is known as instrumental conditioning (Skinner, 1935)—a learning subtopic concerned with the study of how humans and animals learn to select actions in the face of rewards and punishments.

Pavlovian conditioning (Pavlov, 1927) is another area of psychology closely related to RL. Pavlovian conditioning, in contrast to instrumental conditioning, studies situations where an animal has to learn the predictive relationship between events in the environment without being able to affect the received outcome. Hence, it focuses on how the animal learns to predict the values of the different states of its environment. This is an important requirement of many RL methods, since being able to accurately estimate the state values means that the learner can pursue the actions that lead to the most valuable states.

One of the most successful approaches to the problem of estimating state values, and which had an impact on subsequent RL methods, was developed by Rescorla and Wagner (1972). Their model is based on a simple error correcting learning rule that the animal is assumed to compute and use to update the state values in each new trial. The correction, and hence learning of the state-values, occurs only when the animal is faced with unexpected outcomes. More specifically, the update of a state value is based on the discrepancy between the expected and obtained outcome, such that the direction of the update—whether it increases the state value or decreases it—depends on the sign of the difference between the expected and observed outcome. This powerful idea had already been used before under the model of Bush and Mosteller (1951), but they modelled the probability of association between the stimuli instead of the strength of association as in Rescorla and Wagner's (1972) model (Rescorla and Wagner made other important modifications to the original model of Bush and Mosteller, but we will not discuss them here; for more details, see Le Pelley, 2004). A major part of the success of the Rescorla-Wagner model is that it explains many of the behavioural puzzles about Pavlovian conditioning, such as blocking (Kamin, 1969) and superconditioning (Rescorla, 1971).

An important breakthrough in RL research was the development of the temporal difference (TD) framework (Sutton and Barto, 1981) as a refinement of the Rescorla-Wagner model. Many of the widely used RL algorithms today are based on this framework, such as Q-learning (Watkins, 1989), SARSA (Rummery and Niranjan, 1994) and Actor-Critic (Barto et al., 1983). TD learning proposes a method for not only predicting the immediate reward in each state, but also the sum of all rewards in subsequent states. The updating rule of TD is very similar to that of the Rescorla-Wagner model, that is, the state values are adjusted by adding a proportion of the prediction error (i.e., the discrepancy between the expected and obtained outcome). However, the two models differ in how they compute the prediction errors. Another difference is that TD offers more flexibility in modelling the timing of events, as the change in value of an event can be computed at any time within a trial, whereas in the Rescorla-Wagner model the change in associative strength of an event is specified as the result of a trial as a whole. This distinction between times within a trial allows the TD model to predict the effect of varying the temporal relationship among stimuli within a trial, which cannot be predicted using the Rescorla-Wagner model.

The impact of TD-based models goes beyond explaining behavioural data or solving difficult machine learning programs. Researchers have recently started applying these models to explain neural data and to understand the roles of neuromodulators like dopamine and serotonin. For example, several studies suggest that dopamine responses encode the computation of the reward prediction error of TD learning

(Schultz et al., 1997; Hollerman and Schultz, 1998; Bayer et al., 2007). Also by tracking the pathway of dopaminergic signals in RL tasks, studies have shown that the dorsal and ventral stratum—two principal components of the basal ganglia—are highly targeted by dopamine neurons, and that they function like an Actor-Critic architecture, with the ventral stratum representing the "Critic" part of the model, which predicts state values, and the dorsal stratum learning actions values as in the "Actor" (Joel et al., 2002; Balleine and O'Doherty, 2010). The basal ganglia, in general, has been associated with action selection and reward learning (Packard and Knowlton, 2002). Although the neural data is generally consistent with temporal difference learning, it is still not clear which algorithm is specifically supported. For instance, Morris et al. (2006) favoured SARSA over Actor-Critic, whereas Roesch et al. (2007) found evidence supporting Q-learning against SARSA.

The contributions to RL from the field of optimal control came mainly from the idea of formulating the RL problem as a Markov decision process (MDP) (Andreae, 1969), and using dynamic programming to iteratively solve the estimation problem of state and action values. More specifically, under the MDP formulation, optimal state(-action) values follow from an equation called the Bellman equation (Bellman, 1957), which can be solved using dynamic programming techniques such as policy iteration (Howard, 1960) or value iteration (Puterman and Shin, 1978).

The idea of using trial-and-error learning in computer programs and artificial machines dates back to the 1950s (Minsky, 1954; Farley and Clark, 1954). The early applications consisted mainly of learning to solve games such as checkers (Samuel, 1959), tic-tac-toe (Michie, 1961), blackjack (Widrow et al., 1973), multi-armed bandits (Tsetlin, 1973), Backgammon (Tesauro, 1992) and more recently the game of Go using deep RL (Silver et al., 2016). Applications of RL in artificial intelligence now include robotic (see Kober and Peters (2013) for a survey), intelligent heating systems (Van Vaerenbergh et al., 2015), intelligent tutoring systems (Chi et al., 2010) and autonomous helicopters (Kim et al., 2004).

A current hot topic in neuroscience related to RL is concerned with distinguishing brain areas that are responsible for model-free RL and those areas involved in model-based RL, and how the brain arbitrates between them (Daw et al., 2005; Balleine and O'Doherty, 2010). The difference between these two classes of RL is that model-free methods (usually associated with habitual learning) do not require the learner to construct a model of the environment in order to learn to act in it, as opposed to model-based ones (usually associated with goal directed learning), which try to learn optimal behaviour strategies by learning a model of the environment. In this work, however, we only use model-free methods.

Recently, another class of RL techniques has emerged based on the Bayesian framework. Relevant to this project are works by Engel et al. (2003), and Geist and Pietquin (2010) in machine learning, which highlighted the advantages of the Bayesian approach over classical RL methods, but still have not received the over-due interest in the behavioural sciences (though there are a few exceptions; see for example, Daw et al., 2005). Bayesian RL models have the advantage of taking into account uncertainty when estimating state(-action) values, which is often ignored in classical RL methods like those discussed above. Being able to track uncertainty is an important aspect of human cognition, as it allows people to track changes in the environment and to have a measure of how sure their value estimates are when selecting actions. Brain studies have shown that there exist special neuromodulators, namely acetylcholine (ACh) and norepinephrine (NE), for processing different types of uncertainties in the environment. According to Yu and Dayan (2005), expected uncertainty, which refers to known unreliability in stochastic environments (e.g., a cue that we know, from accumulated experience, predicts reward about 60%

times), is tracked by ACh, while unexpected uncertainty, which reflects a fundamental change in the environment that was not expected by the agent (e.g., a non signalled switch in the response mapping rules), is signalled by Ne. We will come back to these two types of uncertainties under RL in Chapter 2.

The Bayesian approach is also compelling because, in principle, it proposes a natural way to deal with the exploration-exploitation dilemma encountered in RL problems—that is, whether to exploit the knowledge acquired so far and select actions that have the highest values or to explore more the state-action space to see if there are alternative actions with higher values than the current ones. The idea is that a Bayesian RL method would estimate the whole distribution over the state(-action) values and use the distribution uncertainty to guide exploration. This idea has been exploited extensively, for example, in Bandit problems, which have been used in many fields such as medical decision making (e.g., design of clinical trials; see Kuleshov and Precup, 2000) and web optimisation (e.g., adverts to display; see May et al., 2012). In Chapter 5, we will explore the Bayesian approach by applying it to tasks from the psychology literature that are related to the present work.

Let us now define the RL framework that the discussed and other related methods are based upon. We will only consider model-free RL techniques since, as mentioned before, model-based methods are not treated in depth in this thesis.

1.1.2 Reinforcement learning framework and models

An RL task consists, in general, of a decision maker (e.g., an animal) that interacts with its environment at each of a sequence of discrete time steps ($t = 0, 1, 2, \ldots$). At each time t, the agent chooses an action a_t (e.g., forage in patch X or Y), depending on the current state of the environment s_t (e.g., morning or evening; abundant or scarce patch). The agent's action might change the environment state (e.g., patch becoming scarce), and results in a reward given r_t (e.g., amount of food that has been found), which represents the short-term desirability of action a_t in state s_t. The agent is assumed to seek actions that have the highest values, which represent the long-term desirability of actions (e.g., food will be associated with a large positive value, whereas pain with a negative value). Theoretically, the value of a state-action $Q_t^\pi(s,a)$ is defined as the total amount of reward that the agent expects to accumulate over the future, starting from state s and taking action a, and then following an action-selection policy π (i.e., a response mapping between states and actions):

$$Q_t^\pi(s,a) = \mathbb{E}_\pi \left[\sum_{i=0}^{\infty} \gamma^i r_{t+i} \middle| s_t = s, a_t = a \right] \tag{1.1}$$

where $0 \leq \gamma \leq 1$ is a discount factor that controls the impact of rewards as a function of their distance in the future. The discount factor is used to capture the idea that rewards in the distant future are less valuable to the agent than rewards in the near future. The value of a state $V_t^\pi(s)$ is defined as the total expected reward if the agent starts from s (to simplify notation later on, we will omit the superscript π, and just write V_t and Q_t to refer to the state and state-action value function):

$$V_t^\pi(s) = \mathbb{E}_\pi \left[\sum_{i=0}^{\infty} \gamma^i r_{t+i} \middle| s_t = s \right] \tag{1.2}$$

Thus, the ultimate goal of the agent is to find a policy that maximises the Q-values, which implicitly requires the agent to effectively estimate them. There are several algorithms that can perform this estimation on-line; two that are relevant to our work are SARSA and Actor-critic. In the following, we will start by showing how both algorithms predict state (-action) values, then we will discuss possible methods that can be used to govern action selection.

SARSA model

One of the most well-known TD-based RL algorithms is SARSA (Rummery and Niranjan, 1994). It works by keeping track of point estimates of state-action values of the current policy, which are updated, at each time step t, according to:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha\delta_t \tag{1.3}$$

where α is a learning rate that scales the magnitude of the updates, and δ_t is a prediction error that represents the discrepancy between the expected reward and the observed reward at time t. δ-errors are computed using the TD Equation 1.4 (see Figure 1.1 for more details about the steps followed by the learner in SARSA):

$$\delta_t = r_t - (Q_t(s_t, a_t) - \gamma Q_t(s_{t+1}, a_{t+1})) \tag{1.4}$$

Actor-critic model

Actor-Critic (Barto et al., 1983) is another RL method based on temporal differences. It differs from SARSA in that it keeps track of both state values and state-action values. The state-action values serve the action selection process, and are computed and held in the "Actor" component of the model (see Figure 1.2). The state values are used by the "Critic" component to compute the TD error δ_t, which serves to update both the state and state-action values.

In this new setting, the TD equation that gives the δ-errors becomes:

$$\delta_t = r_t - (V_t(s_t) - \gamma V_t(s_{t+1})) \tag{1.5}$$

The computed prediction error is then used to update both the state values and state-action values in the same way as in SARSA. That is:

$$V_{t+1}(s_t) = V_t(s_t) + \alpha\delta_t \tag{1.6}$$

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha\delta_t \tag{1.7}$$

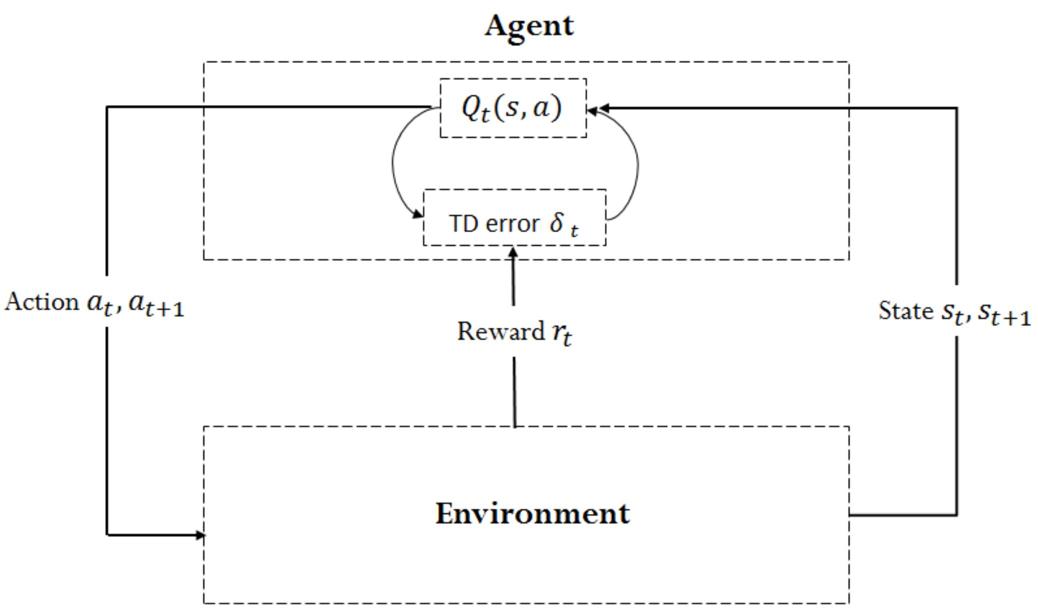


Figure 1.1: SARSA model. At time t, the agent chooses an action a_t based on the observed state s_t and the current estimates of state-action values $Q_t(s, a)$. As a result, the environment produces a reward signal r_t, which is used by the agent to update its Q-values. Before proceeding with the updates, the agent needs to first observe the next state s_{t+1} and plan the next action a_{t+1} (again based on the current estimates of the state-action values $Q_t(s, a)$). Only then the temporal difference prediction error is computed. Adapted from Sutton and Barto (1998).

Eligibility trace

For now we presented SARSA and Actor-Critic in their basic forms, but there are more general forms of these algorithms that use eligibility traces (Sutton, 1984) to allow the learner to modify state (-action) values other than the values of the currently visited state (and chosen action). This is particularly useful when dealing with non-Markovian or partially observable Markovian tasks (Loch and Singh, 1998).

Under the new versions of SARSA (referred to as SARSA(λ)) and Actor-Critic, all state and/or state-action values are potentially updated using modified versions

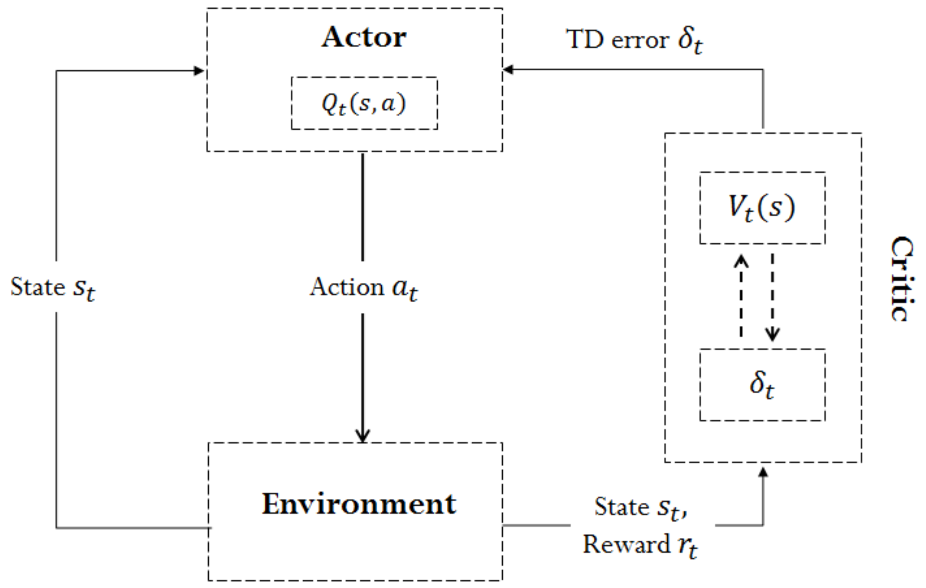


Figure 1.2: Actor-Critic architecture. This model has two major modules: the Actor and the Critic. The actor component is where the actions are chosen based on the state-action values $Q(s, a)$. After an action has been taken, the environment sends a reward signal to the critic, which generates a prediction error signal δ and updates the state values $V(s)$. The actor then uses this error signal to update the state-action values.

of Equations (1.3) and (1.6):

$$V_{t+1}(s) = V_t(s) + \alpha\delta_t e_t(s) \tag{1.8}$$

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha\delta_t e_t(s, a) \tag{1.9}$$

where $e_t(s)$ and $e_t(s, a)$ are respectively the eligibility traces of state s and state-action pair (s, a). An eligibility trace works as a historical record of the previously visited states (and chosen actions), so as to modify their values accordingly using the TD-error. More specifically each state/state-action pair will be associated with a trace that decays over time by a parameter $\lambda \in (0, 1)$, called decay parameter. If $\lambda = 1$, then there is no decay, and all state and/or state-action pairs are updated with the same weight using the computed δ-error. On the other hand, if $\lambda = 0$, then there is a complete decay after one time step, and hence only the value of the current state/state-action pair is updated (as in SARSA for example). Formally, eligibility traces are updated as follows:

$$e_{t+1}(s) = \begin{cases} 1 & \text{if } s = s_t, \\ \gamma\lambda e_t(s) & \text{otherwise.} \end{cases} \tag{1.10}$$

$$e_{t+1}(s,a) = \begin{cases} 1 & \text{if } s = s_t \text{ and } a = a_t, \\ \gamma\lambda e_t(s,a) & \text{otherwise.} \end{cases} \tag{1.11}$$

Notice that here we used a "replacing trace" form for updating eligibility traces, but there exist other forms such as the "accumulating trace" (see Sutton and Barto, 1998, for more details).

Action-selection

Now that we showed how SARSA and Actor-Critic methods can be used to estimate state and state-action values, there remains the problem of how to select actions based on those estimated values. There are several widespread methods for this problem, such as ϵ-greedy or softmax (Sutton and Barto, 1998).

For example, under the ϵ-greedy rule, the agent chooses the action that has the highest Q-value with probability $1-\epsilon$ and randomly chooses another action with a small probability ϵ. The problem with this method is that it does not take into account the difference in values when randomly selecting an action (suboptimal actions with high or low Q-values are selected with the same probability).

Softmax (or Boltzmann) action-selection rule tries to overcome this by selecting actions with probabilities proportional to their values. More precisely, the probability of selecting, at time t, action a in state s, is given by:

$$p(s,a) = \mathbb{P}(a_t = a | s_t = s) = \frac{\exp(Q(s,a)/T)}{\sum_{a'} \exp(Q(s,a')/T)} \tag{1.12}$$

where T is the exploration or temperature parameter, which determines the degree of stochasticity in action selection. High values of T lead to more exploration (actions tend to become equiprobable). Decreasing T results in more exploitation (actions with the highest state-action values become more likely to be chosen).

RL methods like SARSA and Actor-Critic are only suitable for Markovian RL tasks—that is, tasks where the next state and reward depend only on the current state and chosen action. When the Markov property breaks, these methods are no longer guaranteed to work. The standard generalisation of Markovian processes is the partially observable Markov decision process.

1.2 Reinforcement learning of partially observable tasks

The partially observable Markov decision process (POMDP) framework was first introduced by Aström (1965) to represent situations where the agent cannot unambiguously observe the actual state of the environment. The environment is assumed to be Markovian, but the observations that the agent perceives might be insufficient to identify the state of the world. Two situations that cause this are: (1) uncertain observations (for example due to perceptual noise) and (2) dependency of some states on past events (some sort of memory is needed). Next, we consider each of these cases in turn.

1.2.1 Reinforcement learning with state uncertainty

State uncertainty refers to a situation where the agent is unable to tell for sure what is the current state of the world. This might happen because its perceptual system is noisy (e.g., noise in the brain or in robot sensors), or because the observations are ambiguous. To illustrate the latter case, consider a task from the neuroscience literature: the random dot motion discrimination (Shadlen and Newsome, 1996). In this task, the decision maker—usually a monkey—observes a set of moving dots with some dots coherently moving in one direction, and the remaining dots in a random direction. As the number of coherently moving dots decreases, the stimulus become more ambiguous, and hence the uncertainty of detecting the correct state (i.e., direction of motion) becomes greater. The decision maker's task is to report the direction of the coherent motion either by pressing a button or making a saccadic eye movement. One interesting question is: if you consider this type of stimuli in an RL problem, how should the agent deal with uncertainty while learning? More specifically, how should the agent perform TD-like updates if it is not sure which state is responsible for the observed reward?

Interestingly, this question has received less attention than the problem of dealing with uncertainty in general decision making research (Shadlen and Newsome, 2001; Bogacz et al., 2006). Instead, behavioural and neural studies in RL have focused mainly on reward uncertainty. In fact, a body of research has been concerned with differentiating between expected and unexpected uncertainty both computationally and in terms of brain functions, inspired by the work of Yu and Dayan (2003). Some works have also included volatility—that is when unexpected changes are frequent as opposed to rare in normal unexpected uncertainty (Behrens et al., 2007; Nassar et al., 2010; Wilson and Niv, 2011; Bland and Schaefer, 2012). Thanks to these studies, we now know that the human brain deals differently with these different types of uncertainty.

One of the few exceptions to this is a computational study by Larsen et al. (2010), where they proposed a modification of the TD learning rule capable of successfully dealing with state uncertainty. The basic idea behind their model is that rewards carry information about the true state, and hence could be used along current value beliefs to estimate the (posterior) probability of each state being the true state. More specifically, the model updates the Q-values using a TD rule that is weighted according to the estimated state probabilities. However, their model fails to adapt to the unexpected uncertainty caused by a switch in the response mapping rules in the case of a normally distributed reward function. This happens because by using rewards and current value beliefs to inform state beliefs, the model can fail to respond to new reward information that contradicts its current beliefs about the true state values, and might instead wrongly adjust its state beliefs. Thus for example, a lack of reward might be attributed to a misidentification of the state rather than to incorrect Q-value estimates, so the learner would not adjust her value estimates.

Larsen et al.'s (2010) paper introduced other possible models such as one with a very simple version of TD updating, where the learner updates only the state that she believes is the most likely current state, but this alternative model failed to converge to the correct Q-values (although it better adapted to the switch in response contingencies). Their modelling work generated a clear prediction: if people were to do a task with both expected state uncertainty and unexpected uncertainty, and with deterministic rewards, they would manage to learn the rules until the switch, after which, most of them will fail to adapt. This prediction will be discussed in depth and tested in Chapter 2.

Another line of research related to learning with state uncertainty comes from the machine learning field, which concentrates on overcoming what is known as the problem of perceptual "aliasing". Aliasing happens, for example, in spatial navigational tasks, when two or more observations appear identical, but come from different states, and hence might require different actions. To illustrate this, consider this simple T-maze (see Figure 1.3), which was proposed by McCallum (1993). The agent starts randomly in one location, and can only detect whether there is a wall immediately north, east, south or west of itself. At each location, the agent can choose among four possible actions: go north, east, south, or west, which move the agent in the indicated direction. The objective of the agent is to reach the goal position, which is labelled "G" on the figure, and where it gets a positive reward of +1. When attempting to move into a barrier, the agent gets punished by −1. All other actions result in a negative reward of −0.1. As can be seen in the figure, the three states defined by the coordinates (2,3), (4,3) and (6,3) are aliased since they are all perceptually equivalent to the agent (all of them have two walls in the east and west), but require different actions: in the middle location, the agent has to learn to go south, whereas on the two other ones, it must go north. Similarly, states (3,4) and (5,4) are also perceptually aliased.

A popular approach to deal with perceptual aliasing, and state uncertainty in general, is to construct and keep track of a belief state vector, which encodes the relative likelihood of each possible current state. The agent would then update its belief state as it takes new actions, and receives new observations and reward feedbacks (Kaelbling et al., 1998; Littman, 2009). However in this scheme, the agent chooses actions based on its current belief vector rather than depending on current states and Q-values as in classical RL approaches. Larsen et al.'s (2010) model, which we discussed previously, uses a similar approach, but tries to attribute reward directly to possible states instead of attributing it to belief states (i.e., the model generates a look up table policy mapping from states to actions). The disadvantage of the belief-based approach is that it assumes that the agent has complete knowledge about the possible underlying states, which is rarely the case in real problems.

Other approaches have been proposed in the machine learning literature for dealing with perceptual aliasing. For instance, Whitehead and Ballard (1991) proposed

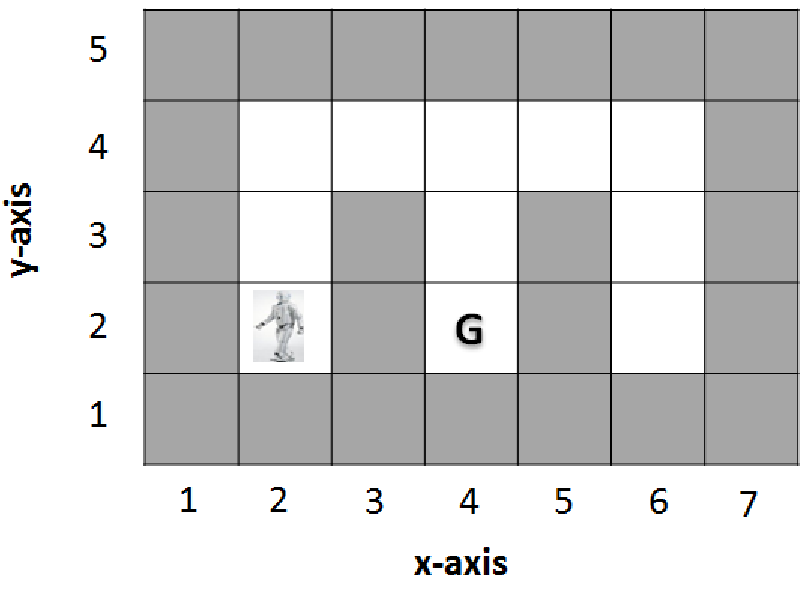


Figure 1.3: Example of perceptual aliasing in a T-maze environment. States (3,4) and (5,4) are perceptually aliased because both have walls in the south and north, and hence get identified by the agent's sensors as identical. The x and y axes were added solely to help us describe the agent's location when needed, and are not part of the task. Adapted from McCallum (1993).

an algorithm that avoids passing through aliased locations, but obviously this trivial solution would not work on tasks where the agent cannot avoid aliased states in order to reach its target position, such as the T-maze that we described above.

Another simple solution is to ignore the hidden states and use only the observable state along with eligibility traces (Loch and Singh, 1998). This method works well only in POMDP problems that admit memoryless optimal policies (i.e., policies that directly map observations to actions). In particular, it can get stuck into a deterministic loop. For example in McCallum's T-maze (Figure 1.3), imagine that the agent has learned to go right when it senses walls in the north and south as in state (3,4), then the agent would use the same action in state (5,4), which would result in the agent going back and forth between state (6,4) and (5,4) (potentially until the agent has unlearned the action of going right and replaces it with left, but then this would create a similar problem in state (3,4)). One way to overcome this would be to use stochastic memoryless policies (Singh et al., 1994), but these usually require a long time to be learned.

A better approach consists of supplementing the agent with a memory device that can store past observations or/and actions, allowing the agent to disambiguate the current state based on past observations (Littman, 1994; Peshkin et al., 1999). Before discussing this type of methods, it is worth noting that there are other efficient approaches for solving memory-dependent partially observable tasks, which do not require memory, but we will not discuss them here since they are out of the scope of this thesis. One example is the HQ-learning model of Wiering and Schmidhuber (1997), which could successfully learn some partially observable tasks with large state-spaces.

1.2.2 Working memory-based reinforcement learning

The idea of supporting RL with memory started in machine learning. Littman (1994) was one of the first to use this idea, despite being primarily interested in finding optimal memoryless policies in partially observable environments. In his paper, he introduced a model-based heuristic algorithm to speed up the search for optimal (or suboptimal) memoryless policies, and applied it to three well-known problems from the machine learning literature: Sutton's grid (Sutton, 1990), Wilson's wood7 (Wilson, 1986) and McCallum's maze (see Figure 1.3). In the first two problems, he could find good memoryless policies; however, he could not find any deterministic memoryless policy that solved McCallum's maze. As a way to overcome this, he supplemented his algorithm with a memory, and showed that this manipulation can help solve the problem. His added memory can be seen as a short-term memory that can store a bit (i.e., 0 or 1), with its contained information providing additional context to the agent when making decisions (the extra memory bit would serve, for example, to remind the agent of the direction it came from when entering the corridor; see Figure 1.3). Also, in this new version of the algorithm, the agent has to learn to take memory actions (i.e., set or clear the bit memory) simultaneously with learning to take the usual motor actions.

Peshkin et al. (1999) used Littman's approach to augment SARSA and another algorithm called VAPs, again using a single bit memory. They showed that the augmented algorithms can solve a wide range of POMDP problems, with some of them only solvable after using the memory extension. Lanzi (2000) followed the same approach to augment the Q-learning algorithm, this time using a multiple bit memory. His paper proposed several recommendations for improving the performance of methods that rely on the use of memory, such as the benefit of using eligibility traces, and the need to separate memory exploration from environment exploration, with the former having to be lower.

The concept of short-term memory used in these machine learning models accords with our understanding of human cognition. Not only do we assume that there is a separate capacity-limited system or process dedicated to retaining information over short periods, but such short-term memory system has recently been implicated in RL in modelling and behavioural studies. Before spelling out these links in more detail, we first introduce some general notions related to working memory that are necessary for our discussion.

What is working memory?

Working memory (WM) is one of the main cognitive systems in humans and animals. It is a memory system that underlies many of our mental abilities such as thinking, planning, problem solving and learning (Engle, 2002). WM refers to the ability to temporarily store and manipulate information. WM should be distinguished from short-term memory, which is only responsible for temporary storage of information, while WM combines storage and manipulation (Baddeley, 2012). It is also considered to be a gateway to long-term memory (Atkinson and Shiffrin, 1971).

It is clear that a WM system that exists only for the pure sake of temporary storage of information would not have been useful to humans or animals. Instead, a system that allows an organism to hold in mind task-relevant information, while performing other cognitive processes is very advantageous (Conway et al., 2005). For instance, the simple task of discussing with other people would not have been possible without it, since we have to keep in mind important aspects of our interlocutor message while processing our own speech (Clark and Clark, 1977).

Mental arithmetic is often used as an example to illustrate the use of WM. For instance, multiplying two numbers (e.g., 31×25), without using a pen and a paper or a calculator, necessitates storing the two numbers temporarily, while carrying intermediate products of successive pair of numbers (e.g. $5 \times 31 + 20 \times 31 \ldots$). Reading is another example of everyday activities that call upon WM. In fact, a reader needs to keep in mind the meaning of what has been read so as to use it to understand the coming parts of the text (Daneman and Carpenter, 1980). It is clear that doing such mental activities requires holding in mind some information while processing other operations.

WM is a very fragile form of memory. It is easily disrupted by other events such as unrelated thoughts that try to capture attention. This fragility is what makes it a successful memory system over short periods. The main reason behind this is that WM has a limited storage capacity. Any attempt to model WM should take into account the noise and capacity limit attached to it.

The capacity limit of working memory

It is now well-established that WM has a limited capacity, although there is still a controversy over the precise capacity limit. Miller (1956) was the first to propose a number describing the capacity limit in his famous paper *the magical number seven plus or minus two.* He showed that people can recall only about seven items or "chunks" from a given list, where a chunk might be a digit, a letter, a word or a combination of two or more items that forms a memorable group. Other estimations of WM capacity were proposed after Miller's work ranging from one to four (see for example, Oberauer, 2002; Gobet and Clarkson, 2004; Cowan, 2001).

WM is also limited in term of how long it can keep information alive. Of course, if a person keeps repeating a list of items to herself, she can prevent them from fading away, but she will not be able to do much else. Baddeley et al. (1975) showed that people are able to remember as many words as they can articulate in two seconds, suggesting that the retention interval is about 2 seconds. But this finding may only concern short-term memory rather than the overall WM. Others have proposed that unrehearsed information can be retained for approximately 15 seconds (Reitman, 1971; Shiffrin, 1973). More recently, Cowan (2005) proposed that information retained in short-term memory is highly degraded within 7 to 15 seconds and completely erased within 20 to 30 seconds. Up to now, as with the capacity of WM, there is no consensus regarding its exact retention interval. However, all WM scholars agree that WM has to be actively maintained against decay or interference in order to retain what is stored in it.

Several hypotheses have been proposed for why WM has a restricted capacity (for a thorough summary, see Cowan, 2005). For example, from a functional perspective, a limited capacity has many advantages: it facilitates searches through information in WM as shown using mathematical modelling (Dirlam, 1972; MacGregor, 1987;

Van Zandt and Townsend, 1993). Also using mathematical simulations, Kareev (2000) showed that the likelihood of detecting existing correlations between items increases as one decreases the sample size, suggesting that having a limited WM capacity makes it easier to find associations between events in the environment. Although, decreasing the sample size also increases the false alarm rate for detecting correlations, the benefit of early detection of correlations can represent the boundary between life and death in some risky domains. Consistent with this model-based observation, Kareev et al. (1997) showed that individuals with lower WM capacity were better at detecting and using correlations than those with higher WM capacity.

More related to the present work, another advantage of having a capacity-limited WM is that it can sometimes be the only way to make learning possible in some RL tasks where the use of past information is needed. For example, Todd et al. (2009) showed that their WM-based RL model failed to learn an artificial grammar learning task in a practical amount of time when they simulated it with a large WM capacity (more details about Todd et al.'s (2009) work will be given below and later in Chapter 3). Next, we will describe some standard methods for testing working memory before defining the class of working memory tasks that we use to test some of the models presented in this thesis.

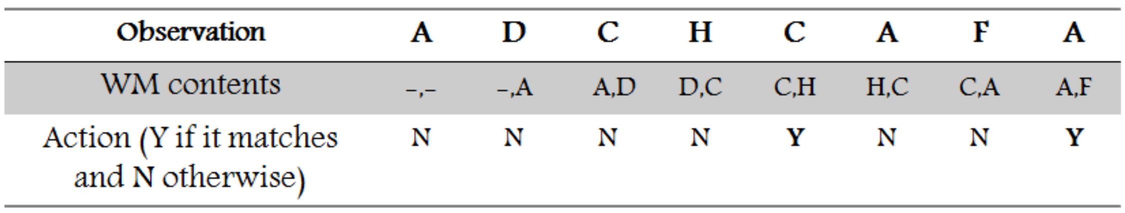
Working memory tasks

There are several WM tasks for measuring WM capacity. Among the first tasks that were proposed in the literature are the digit and word span tasks. In the digit version, participants are asked to repeat back a list of digits (e.g. $< 3, 5, 1 >$) immediately after being presented with it. The length of the list is increased by one digit (e.g. $< 1, 7, 9, 5 >$) if the subject recalls it correctly. A person's digit span is the longest list that he or she can remember. This was actually the task that Miller (1956) used to get his estimate of WM capacity. However, memory span tasks, at best, give a measure of short-term memory, as they only require storage and rehearsal, and do not involve much of information processing, which is an important component of WM. Moreover, studies that use memory span tasks often calculate the number of individual items that can be retained in short-term memory rather than the number of chunks as Miller postulated. This practice probably overestimates the actual WM capacity.

Daneman and Carpenter (1980) introduced a modified version of the memory span task that takes into account the processing component of WM, which they called the reading span task (also known as WM span task). In this task, participants have to process sentences by indicating their veracity, while retaining the last word of each sentence. Turner and Engle (1989) proposed a modified version of this task called operation span, where participants have to judge the correctness of mathematical equations (e.g. $(2 \times 4) - 1 = 7$) instead of sentences. The counting span is another WM task that was proposed in the literature (Case et al., 1982). It is a task that simply involves counting different classes of shapes (e.g. green, yellow and red dots shown on a screen) and retaining their separate counts, and hence can be used with adults, children, elderly and patients.

A more relevant type of WM tasks to our work is the class of dynamic span tasks such as the running-memory span (Pollack et al., 1959), the keeping-track (Yntema and Mueser, 1960), and the n-back (Kirchner, 1958). This class of WM tasks requires participants to continuously update WM contents as they are presented with new stimuli (for a review of different WM tasks, see Conway et al. (2005)). For example, in the n-back task, subjects are shown a sequence of letters, and have to decide for each one of them, whether it matches in identity the letter that preceded it by n positions in the series. For instance, the 2-back version of this task requires subjects to respond positively if the same letter was presented two steps back (Jonides et al., 1997). This task requires two types of WM processes: (1) storage, because subjects need to store one or more letters in WM depending on the value of n, and (2) updating, since subjects have to update their WM contents with the presentation of each letter. They should drop the oldest letter, add the newest, and keep track of the order of presentation of each. Table 1.1 presents a possible sequence of the 2-back task, and shows how subjects should respond and update their WM contents.

Table 1.1: Example of a sequence in a 2-back task, showing also how subjects should update their WM contents.



Observation	A	D	C	H	C	A	F	A
WM contents	–,–	–,A	A,D	D,C	C,H	H,C	C,A	A,F
Action (Y if it matches and N otherwise)	N	N	N	N	Y	N	N	Y

WM tasks such as dynamic span tasks take WM updating for granted. However, how do people know how to update WM? One theory is that people can learn WM updating through experience (Dahlin et al., 2008; Badre, 2012; Chatham and Badre, 2015). This makes intuitive sense, since nobody taught us explicitly how to use our WM, for example when we read or communicate with others. Instead, we have probably learned it through a learning process that comprised reward feedbacks, which were sometimes accompanied with social cues that indicate the important parts of a text or a speech (Boyd et al., 2011; Tomasello et al., 2005). Before reviewing the literature supporting this thesis and showing how learning WM updating functions can take place, we next define the class of WM tasks that are used to study this question.

Working memory learning tasks

WM tasks are classically presented to subjects by explaining to them, among other things, the motor (i.e. physical) actions that they should use. For example, in the n-back task, participants would know that if the current letter matches the one presented n trials previously, then they should click on a certain button (e.g., the keyboard letter 'Y'). Another way to assign this task, which would transform them into partially observable learning tasks, would be to ask participants to learn the correct motor actions, by giving them a reward feedback after each selected action. Implicitly, subjects would also need to learn how to update their WM contents. This is the approach that we take in many of the tasks that we use in this thesis. This allows us to study how people learn important WM functions like when to maintain and when to update WM; a study topic that is often referred to as the study of learning to use WM (see for example, Lloyd et al., 2012; Gorski, 2012).

To illustrate more what we mean by WM learning tasks and what are the cognitive demands related to them, consider this task: the 12-AX (Frank et al., 2001), which is a generalisation of the continuous performance test (CPT) task (Rosvold et al., 1956). This task will be fully defined in Chapter 3, but for our current purpose, let us briefly introduce it here. Participants see a sequence of digits (1 or 2) and letters (A, B, C, X, Y or Z), and are required to respond to the letters X and Y differently depending on which digit (1 or 2) and letter (A, B or C) cues they have previously encountered. More specifically, there are two target sequences that subjects have to respond to in a special way: "1-A-X" and "2-B-Y" (note that the items in each sequence can be interleaved with distractor items). A key to success in this task is to learn to independently store the digit and letter cues when they are presented, and then retain them until the next cues are shown. Also it is important to learn how to choose a motor action based on the external input and what is held in WM.

The question of how WM learning tasks, such as the 12-AX, can be solved is often studied under the gating framework—a class of models that is concerned with how humans (and animals) learn to use their WM.

Gating framework

The gating framework refers to a collection of theoretical models based on neural and behavioural evidence that address how WM updating functions can be learned (Braver and Cohen, 2000; Rougier et al., 2005; O'Reilly and Frank, 2006). Models working in this framework assume that the prefrontal cortex (PFC), the basal ganglia and the dopaminergic system interact to train WM to choose updating actions. This includes what information to store, when to store it, and how to utilise it in action selection.

The designation "gating" alludes to the analogy of having a gate that controls what enters to WM and when. In other words, gating models assume that there is a WM gate that represents the mechanism by which WM is updated with new inputs (opened gate) or maintained and protected against distracting information (closed gate). The idea of using a gating mechanism to support learning was introduced in the machine learning field by Hochreiter and Schmidhuber (1997) in their Long Short-Term Memory (LSTM) model. The motivation behind introducing the LSTM model was to overcome some of the problems that conventional recurrent neural networks face when used to solve tasks with long-term dependencies between events. In the original LSTM model, gating actions consist of opening or closing access to a number of specialised network units called constant error carrousels, which store useful error signals, either for reading (using the stored information by the output units) or writing purposes (storing new information that can be used by the input units). The model has to learn—among other things—when to open and close access to these constant error carrousels, for example, to prevent them from getting filled with useless information (a later version of the LSTM model includes a third possible gating action, forget, which resets the constant error carrousels to their initial state; see Gers et al., 2000).

In neuroscience inspired gating models, the PFC is assumed to maintain context representations in WM, while the basal ganglia decides what to gate into WM, and how to act given those WM context representations and the current environmental input. Both these functions are assumed to be learned through the same dopaminergic mechanisms that support learning to act in the external world. The role of dopamine, thus, consists of regulating gating responses and facilitating the learning of reward prediction.

Although collecting evidence for (and potentially against) gating models is still at an early stage, several studies show support for the involvement of the aforementioned brain structures in gating (for a concise review, see Badre, 2012). For example, it is now well known that the PFC is implicated in both storage and control of WM (Braver and Cohen, 2001). In particular, dopamine level in the PFC increases during WM tasks (Watanabe et al., 1997). More generally, dopamine activity increases when encoding cues during a WM task (Schultz et al., 1993). Moreover, deficits in dopamine due to pharmacological or systematic manipulations, or due to disorders like Schizophrenia or Parkinson disease is associated with low WM performance (Sawaguchi et al., 1990; Kimberg et al., 1997; Cohen et al., 1999a). The belief that the basal ganglia controls the WM gate comes from studies that show its involvement in adaptive motor control by facilitating some actions, while suppressing other, and given that it has also been shown to support WM functions such as short-term storage and manipulation of internal representations (Frank et al., 2001).

One of the first evidence-based gating models is due to Braver and Cohen (2000). Their model, however, did not include the basal ganglia, and considered that WM can only store a single input representation in the PFC via phasic dopamine signals similar to those involved in classical temporal difference-based RL. A refinement of this model, called PBWM (prefrontal, basal ganglia and working memory), was later proposed by O'Reilly and Frank (2006), which, among other things, introduced three main modifications to the original model of Braver and Cohen (2000): (1) it included the basal ganglia; (2) it was based on an actor-critic architecture; (3) it considered a WM that has multiple buffers, each of which can store one input representation and is updated independently from the other buffers. O'Reilly and Frank (2006) demonstrated the power of their model by showing that it was capable of acquiring optimal internal and external policies in the 12-AX task, as well as in two other complex WM learning tasks. However, their model included supervised and unsupervised learning as well as an RL system, which makes it difficult to assess the role of WM in RL. Moreover, the model used a different updating rule than temporal difference learning, which is more similar to the Rescorla-Wagner rule.

In any case, the main problem with these models is that they are highly complex because they try to be as realistic as possible for capturing human gating and to give a lot of details about how they can be implemented in the brain. More recently, there is a new class of emerging models that synthesise the likes of PBWM into a more abstract and common RL framework (Zilli and Hasselmo, 2008; Todd et al., 2009; Lloyd et al., 2012). Rather than giving a lot of importance to biological details, these abstract models concentrate more on the necessary computations for learning how to act and make use of WM.

Relevant to our work is Todd et al.'s (2009) paper, which used memory-based

RL ideas from machine learning but reframed them into psychological terms and showed how they relate to the more biologically plausible gating framework. The general idea behind Todd et al.'s (2009) approach is as follows: the bit (short-term) memory used in machine learning algorithms is replaced with WM, which can directly store inputs (or a representation of inputs) from the environment instead of abstract bit values. As such, WM can be considered as a collection of past observations that, alongside with the current input, provides context to the learner when making decisions. In this new setting, the agent has to learn how to act in the external environment, while learning to use its WM to track past states. Figure 1.4 summarises their model, which will be defined in more detail in Chapter 3.

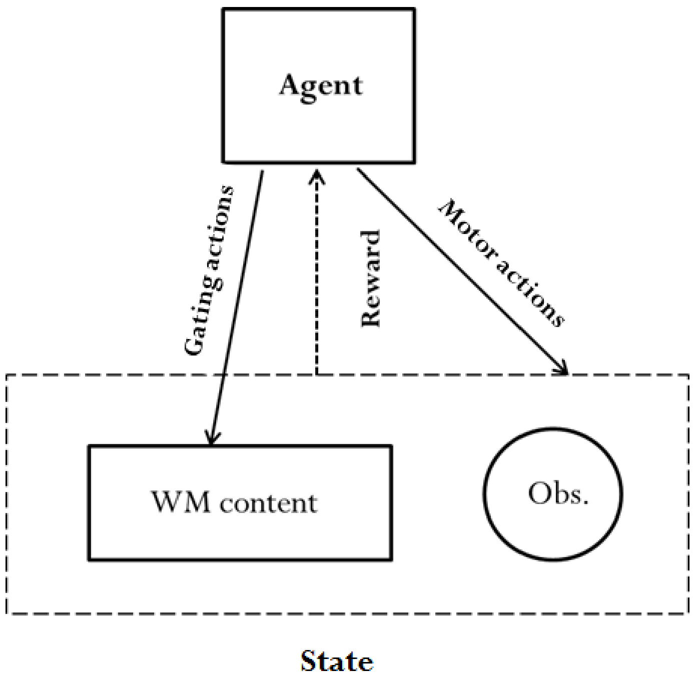


Figure 1.4: The working memory-based reinforcement learning framework as used in Todd et al. (2009). The agent interacts with the environment by taking motor and gating actions, and receiving rewards. Here, the state is defined by both the current observation and the information stored in working memory, as opposed to only the observation in standard RL problems.

Todd et al. (2009) reported that their model could solve the 12-AX, and also could mimic human performance in an articifical grammar learning task. In particular, their model showed failures similar to those exhibited by humans due to their limited WM capacity. More specifically, they noticed that, as they increased WM capacity, task performance increased until reaching a certain level, after which performance started to decrease. Given this diminishing return pattern, they concluded that a limited WM capacity might be optimal from a learning perspective. However, their results were mainly qualitative, as they did not fit human data directly. Moreover, they only concentrated on asymptotic learning performance (i.e., reward in the last n trials), and did not study the effect of WM capacity on learning speed (in fact, a large WM capacity can produce optimal performance but might take forever to converge). Also, learning speed seemed problematic, at least in the 12-AX task, as they had to run their model for twenty million time steps! This is very slow in comparison with the few hundreds trials required by humans to learn the 12-AX, as reported in one experiment set up by Krueger (2011) (more details are given below). Clearly, there is a big gap between what their model (and gating models in general) can achieve and what people are actually capable of.

Zilli and Hasselmo (2008) also augmented model-free RL with WM, but their model was slightly different. The first difference is that their WM could only store one element (although they discussed the possibility to increase WM capacity and showed how to implement it). The second difference is that, in their model, the agent could only take a gating action or a motor action, but not both at the same time. Also it seems that they concentrated more on the effect of adding episodic memory, as they studied tasks that required either WM alone or both WM and episodic memory. Their work, though, offered some interesting insights about a number of previously unexplained behavioural results, such as why the correct response in some maze tasks is sometimes predicted by neuronal activity in the hippocampus—which is associated with episodic memory—and sometimes it is not. Zilli and Hasselmo (2008) argued—based on their modelling results—that this might be because these tasks could be solved either by relying on WM or on episodic memory, and the difference might come from task demands or training conditions.

There have been also some attempts to relate animal learning behaviour to model-free gating models. For example, Lloyd et al. (2012) fitted Actor-critic and SARSA based gating models to rats' learning performance on a maze task that included a reversal of mapping rules. Both models showed good fits to rats' data before the reversal. However, only the SARSA version managed to mirror the fast adaptation of rats to the rules switch. Also, consistent with the conventional wisdom that RL methods, and particularly those including WM gating, learn slowly, rats also showed difficulty in reaching their criteria of successful learning.

In one of the very few studies assessing human performance in WM learning tasks, Krueger (2011) showed that participants could learn the 12-AX task within a single testing session. This goes against the results reported in previous computational studies, which show that state-of-the-art gating models take tens of thousands of trials to solve the 12-AX (see for example, O'Reilly and Frank, 2006; Todd et al., 2009; Krueger and Dayan, 2009). However, Krueger (2011) found that only a few participants could learn the task when they were not given additional contextual information about the task—that is, when the stimuli were not classified into meaningful categories like in the original 12-AX (e.g., 1-2 as the first set of cues, A-B-C as the second set of cues, and X-Y-Z as the targets).

To explain the observed gap between human and model learning speed in the 12-AX, Krueger and Dayan (2009) pursued the idea that key to human ability to quickly learn new tasks is their ability to generalise and use knowledge from previous tasks.

More specifically, they studied the effect of shaping on learning performance and speed of neurally-inspired gating models in the 12-AX, and showed that sequential training can indeed be beneficial. However, the reported model performances were still far from those that would be expected from humans (even with shaping, more than one hundred thousand of trials were required to reach optimal performance). Although prior knowledge might play a major role in explaining the gap between model performance and human performance, we will here explore the possibility that we can construct models that replicate human performance levels in tasks such as the 12-AX, and hence start to investigate how gating models relate to human behaviour in complex WM learning tasks.

1.3 Summary

In order to succeed in their environment, humans have to learn from their past experiences how to act in the most rewarding way. However, in an uncertain and extremely complex world, the rules governing outcomes are often non trivial and might depend on past information. Also, humans have a noisy perceptual system that might limit their ability to identify the exact state of the word to act upon.

As we have seen in this chapter, humans possess the necessary tools to overcome these challenges: brain structures such as dopamine and brain areas such as the basal ganglia facilitate RL in general (Walsh and Anderson, 2014); special neurotransmitters like acetylcholine make it possible to track uncertainty in the environment (Yu and Dayan, 2003); and WM can keep alive information from the past to provide context to the decision maker. Moreover, WM storage and updating are believed to be adaptive and context-dependent processes that can be learned by reinforcement. Based on neural and behavioural evidence, the gating framework shows how this can be done by describing the underlying computational and mechanistic mechanisms that allow humans to learn to use their WM.

However, classical gating models are too complex and biologically detailed that it would be hard to test behaviourally. Other less sophisticated but easier to analyse models have been proposed by behavioural scientists, which concentrate more on the basic computations that might be carried out in the brain when solving WM learning type of tasks. These mere computational models are very similar to previous models from machine learning literature for dealing with non-Markovian tasks, with the difference that they consider the memory device that supports the agent to be simply the agent's WM. This assumption in itself—that is, whether people rely on WM to overcome partial observability or on a different memory system—has not been tested or even identified in the relevant papers that we are aware of (Zilli and Hasselmo, 2008, consider the possible role of episodic memory but do not question the possible non-involvement of WM in their tasks). This is an assumption that we will test in Chapter 6.

Another problem with gating models is that they are very slow to learn. Computational modelling results—although qualitatively interesting at the image of Todd et al. (2009) who showed that their gating model can mimic capacity limitations found in people's WM through model simulations—are far from mimicking human learning speed. For example, Krueger (2011) reported that the participants who managed to learn the 12-AX task in his experiment, did it in average within 800 trials. This is far from the several thousands of trials required by gating models to learn the 12-AX, as reported in several published papers (Todd et al., 2009; O'Reilly and Frank, 2006; Krueger and Dayan, 2009). However, Lloyd et al. (2012) showed that rats can have a learning speed comparable to that of SARSA and Actor-Critic based gating models.

Based on these results, we hypothesise that people use WM to support RL, and that, in the process, they have to learn how to update and make use of WM content. We also assume that people can learn WM learning tasks faster than the current state-of-the art gating models, and hence, we will try to improve these models to be able to test them against human performance. We expect that our work would have an impact in machine learning as well, since improving this type of models is an active area of research in this field.

Finally, we are also interested in the problem of dealing with state uncertainty in RL, which as WM-based RL has not attracted much interest in psychology. The obvious challenge is to answer the question of how the learner should assign rewards to states (and actions) if she is not sure about the current state of the world. Larsen et al. (2010) proposed two RL models to deal with state uncertainty: the first—a Bayesian temporal-difference-based model—predicts that participants would not be able to adapt to a switch in the mapping rules (unexpected uncertainty) under expected state uncertainty; the second is a simple temporal-difference-based model that predicts the inverse. The next chapter tests these predictions and investigates how people's performance compare to these two models.

PART I:

Partial observability due to state uncertainty

Chapter 2

Reinforcement learning under expected and unexpected uncertainty

2.1 Introduction

Classical reinforcement learning (RL) models such as temporal difference learning algorithms (Sutton and Barto, 1998) assume that the learner knows with certainty the state that is responsible for the observed reward on the basis of the sensory input received from the environment. This is a requirement for computing the prediction error and carrying out the necessary updates of the state (-action) values each time a reward is observed. In a partially observable environment, however, it is not clear how reward feedback should be used if the state of the environment cannot be unambiguously identified. State uncertainty might arise, for example, when the learner's perceptual system is noisy (e.g., noise in the brain or in robot sensors), or when the observations are ambiguous. Another source of uncertainty can be induced by a fundamental change in the environment, such as a switch in the required state-action mapping. These two varieties of uncertainty are often referred to, respectively, as expected and unexpected uncertainty (Yu and Dayan, 2003; Bland and Schaefer, 2012).

Previous studies have shown that people are capable of appropriately adjusting their learning behaviour in response to both of these types of uncertainty when the source of stochasticity is the reward function (Behrens et al., 2007; Nassar et al., 2010). For example, Behrens et al. (2007) manipulated the volatility in a simple RL paradigm, and showed that participants adjusted their learning rate depending on the degree of volatility, as would an optimal Bayesian learner do (their learning rate was small when the environment was stable, and large in volatile conditions). Nassar et al. (2010) replicated this result, but reported that human learners did not perform optimally as judged by their overall average accuracy (see also Wilson and Niv, 2011, though their task was made more complicated by including another source of uncertainty due to unknown rewarding stimulus features). Other studies have pointed out the neural mechanisms that allow people to track different types of uncertainty (see Bland and Schaefer, 2012, for a review). For example, special neurotransmitters like acetylcholine (ACh) and norepinephrine (NE) make it possible to track respectively expected and unexpected uncertainty in the environment (Yu and Dayan, 2005). Thus, people are capable of successfully dealing with different types of uncertainty, including presumably expected state uncertainty.

Unlike previous studies involving expected and unexpected uncertainty, the present work investigates the case of state uncertainty instead of reward uncertainty. We set up an RL experiment where the stimuli are noisy, so that participants cannot identify their true state with certainty, and test whether participants can learn the correct rules under this state uncertainty condition. We also introduce a switch (i.e., reversal) of stimulus-response contingencies for some participants without telling them to see if they can detect the unsignaled change despite the presence of state uncertainty.

Larsen et al. (2010) proposed two plausible models for how people might learn under state uncertainty: (1) a simple RL model that assumes that people update the Q-value of only the identified state, and (2) a more complicated Bayesian RL model where the learner is assumed to update the Q-value of each possible state according to its posterior probability of being the true state, given the observed reward. The main difference between these two models is that the Bayesian approach uses the current state-action values along with the observed reward to help inform the beliefs about the current state, whereas the simple model just uses the state information from the environment.

Larsen et al. (2010) showed that, under expected state uncertainty, the Bayesian model can learn the correct state-action values (using both model simulations and the Kullback-Liebler divergence), whereas the non-Bayesian model converges to incorrect values (note that incorrect state-action values do not necessarily result in incorrect actions, so the simple RL model can still be used to potentially learn the correct stimulus-response mapping rules). Given that in practice we can only observe actions—and not values— Larsen et al. (2010) introduced an unexpected switch of stimulus-response contingencies to try to differentiate between the two models. The simple RL model adapts well to this type of environmental changes; the Bayesian RL model, on the other hand, fails in general to adapt to the unexpected change (unless the state uncertainty is small; see Larsen et al., 2010, for further details). The Bayesian RL model fails because by using rewards and current value beliefs to inform state beliefs, the model can fail to respond to new reward information that contradicts its current beliefs about the true state values, and might instead wrongly adjust its state beliefs. Thus for example, a lack of reward might be attributed to a misidentification of the state rather than to incorrect Q-value estimates, so the learner would not adjust her value estimates. Taken together, the non-Bayesian model has the advantage of being simple and more flexible, while the Bayesian version can produce more accurate value estimates at the expense of being more complicated and less flexible.

Given these predictions, if participants learn the task rules before the switch but fail to adapt to it, this would provide support for the Bayesian RL model, whereas if they do adapt to the switch, this would instead favour the simple RL model as a characterisation of human behaviour. To support this simple analysis, we also fit each of the two models to the behavioural data in order to see which model better explains participants' choices.

2.2 General methods

In most previous demonstrations of RL under expected and unexpected uncertainty, the stimuli were unambiguous, whereas the reward was drawn from a random distribution. The current experiment considers a deterministic reward function and uses ambiguous stimuli to study the effect of a rule reversal on the ability to perform RL under state uncertainty.

2.2.1 Participants

24 subjects were tested: 12 served as a control group for whom we did not switch the stimulus-response mapping rules; for the remaining 12 subjects we switched the mapping at the middle of the learning process. Their payment contained a variable amount which depended on their performance on all trials. Therefore, participants

had to perform well in order to earn more.

2.2.2 Stimuli

In each trial, participants saw a patch filled with a variable proportion of white and black dots, in the middle of a gray background (see Figure 2.1). When the majority of the dots were white, the patch appeared lighter than the background ("light" state). On the other hand, when the majority of the dots were black, the patch was darker than the background ("dark" state).

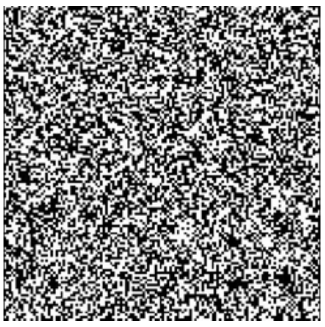


Figure 2.1: An example of stimuli used. Participants saw a patch containing 400 by 400 white or black pixels, placed in the middle of the screen.

As the stimulus was noisy, the learner could only identify the correct state with a certain probability ρ called level of state uncertainty. In our experiment, we aimed to fix ρ at 0.65 for all participants. A preliminary phase was set up in order to create a noisy stimulus that corresponds to this level of state uncertainty.

2.2.3 Procedures

The procedure was the same for the control and test group, except that for the control group, we did not switch the stimulus-response mapping rules. The control condition was set up in order to examine whether people are able to perform RL under state uncertainty (similarly, results from the test group before the switch can be used to answer this question), whereas the test condition should inform us about people's ability to detect an unsignaled reversal in state-action mapping rules in the presence of state uncertainty, and how this change would affect their learning performance. Participants from both groups completed two phases: a preliminary phase and a learning phase.

Preliminary phase

This part aimed at measuring the proportion of white/black pixels that corresponds to the level of state uncertainty $\rho = 0.65$ for each participant (i.e., to construct the light and dark patch that the participant could accurately identify with a frequency of 65%). For that, we measured a full psychometric function—that is, the proportion of "light" responses as a function of the proportion of white dots—using the method of constant stimuli with 9 proportions of white dots that varied between 0.4 and 0.6 (0.4 corresponding to an easily identifiable dark state, and 0.6 corresponding to an easily identifiable light state) spaced at 0.02 intervals (Kingdom and Prins, 2009). There were 25 presentations at each of the nine contrast levels, for a total of 225 trials. Note that the positions of the white and black pixels varied randomly from trial to trial, hence two stimuli can have the same proportion of white and black dots, but might appear different.

In each trial (Figure 2.2), after being presented with the stimulus, participants had to click, using the mouse, on one of two shapes (triangle or rectangle), depending on the state of the patch (light or dark). The correct answer was counterbalanced across participants, so that half of the participants were asked to click on the triangle if they thought the patch was light and the rectangle if the patch was dark; the other half had to use the reversed mapping. Also to reduce any potential cost or bias toward choosing one response stimulus (triangle or rectangle), we positioned the two response stimuli, in each trial, randomly on a virtual circle with the mouse cursor being displayed at the centre of the circle.

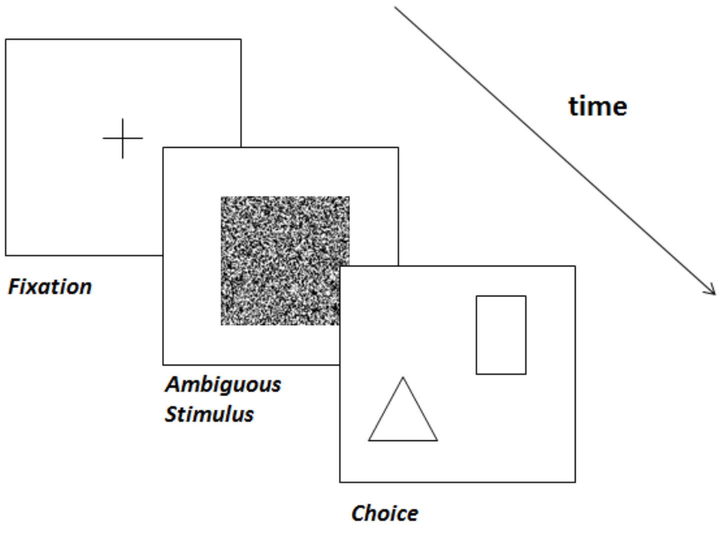


Figure 2.2: Sequence in one trial from the preliminary phase. Firstly, the subject saw a fixation cross to indicate where the stimulus will be shown. Then the stimulus was displayed for about 250ms. After that, the subject had to choose between two shapes depending on the state of the patch. The next trial started following a blank screen that was displayed for 500ms.

After collecting participants' responses, we fitted the obtained frequencies of light responses for each participant with a cumulative Gaussian psychometric function, as shown in Figure 2.3. Using each participant's fitted function, we computed the proportions of white dots that corresponded to the level of state uncertainty $\rho = 0.65$ for both the light and dark state (for the dark state, this was obtained from the figure by computing the proportion of white dots that corresponded to a frequency of light response equal to $1 - \rho$).

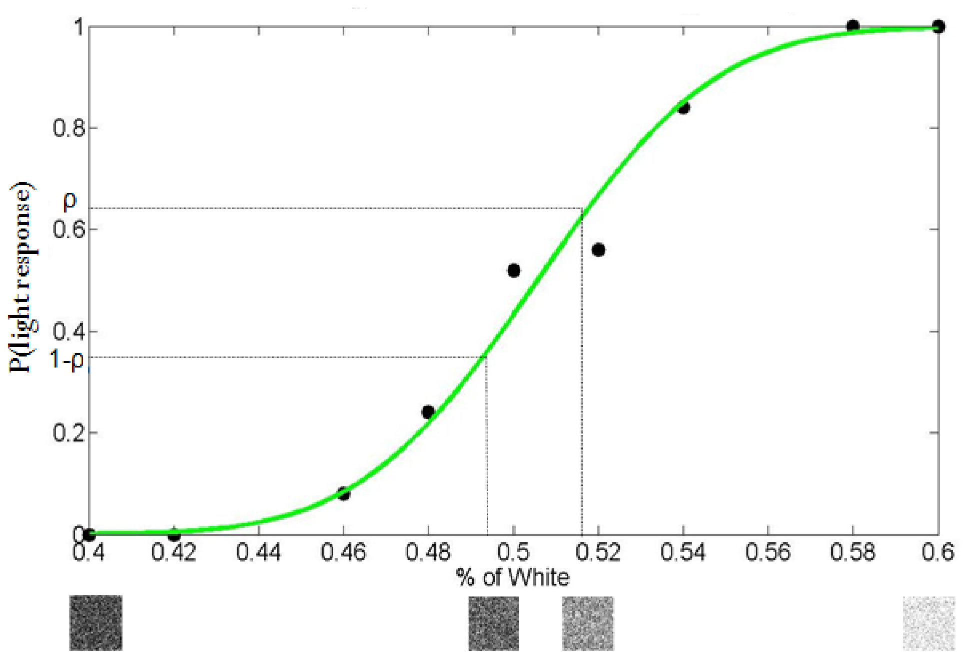


Figure 2.3: Example data and fitted psychometric function. Frequency of light responses as a function of the proportion of white dots for nine contrast levels. On the x-axis, as the percentage of white dots increases, the stimulus would appear lighter to participants. The green curve shows the fitted psychometric function. The dashed lines illustrate how the ρ-thresholds for both the light and dark states are computed using the the fitted function.

Learning phase

This phase contained 200 trials, and similarly to the first phase, each subject saw the stimulus for 250ms, but was not told about the correct mapping between light or dark states and the two response options. In order to avoid any interference from the preliminary phase, the new response options were circle and square instead of rectangle and triangle. Each trial provided a reward opportunity: in trials where the subject chose the correct response, she received a positive reward feedback (£0.025), which lasted for 1.5s; in trials where the incorrect response was selected, no reward feedback was given and the next trial was presented immediately after the response and the inter-trial interval of 500ms. Subjects were assigned the task of maximizing their rewards. Finally, without telling our participants from the test group, we switched the mapping rules at the 101th trial, so that if, for example, the correct response for the light state (resp., dark state) before the switch was circle (resp., square), after the switch, the correct response would become square (resp., circle). Figure 2.4 presents the sequence in one trial.

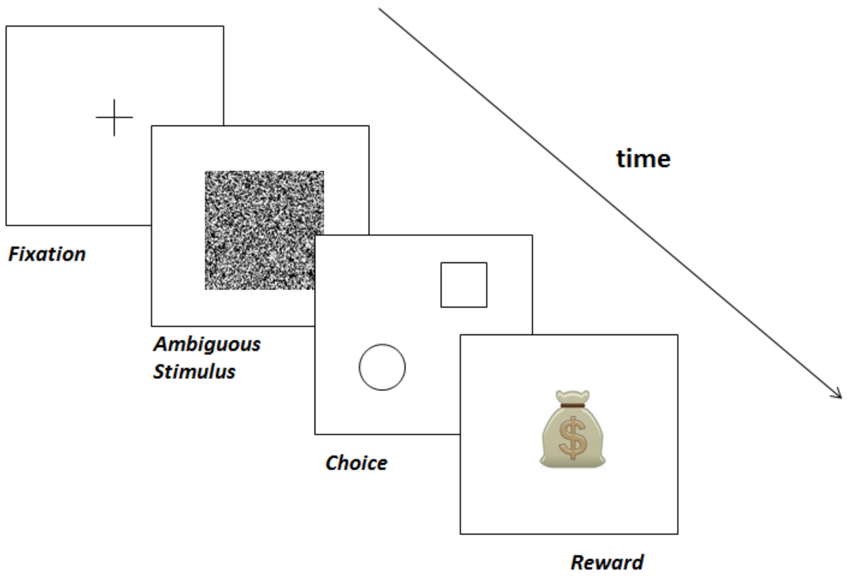


Figure 2.4: The sequence in one trial from the learning phase looked similar to that from the preliminary phase. The only difference here is that participants received a positive reward feedback (winning £0.025) if they responded correctly.

2.2.4 Learning curves

To evaluate participants' learning performance and their ability to adapt to the switch in the mapping rules, we calculated a learning curve for each participant by computing their moving average of accuracy with a window size of 25 (i.e., the averaging was performed for every 25 trials) during the course of the learning phase.

2.2.5 Computational models

To explain participants' performance on the experiment, we compared two computational models: the first, a Bayesian temporal-difference-based model, which shows how a learner, who is aware of the level of state uncertainty, could use the reward feedback to resolve the uncertainty (Larsen et al., 2010); the second is a simple temporal-difference-based model in which the learner ignores the state uncertainty, and updates only the Q-value of the state that she believes is the true state (this corresponds to "the winner takes all" model described in Larsen et al., 2010). Each model was fitted to participants' trial-by-trial responses to shed light on the possible computations carried out by the human brain in RL tasks with expected and unexpected uncertainty.

The PWRL model

PWRL (Posterior Weighted Reinforcement Learning; Larsen et al., 2010) shows how to augment temporal difference learning to deal with state uncertainty. The model allocates reward to each state according to its posterior probability—that is, after observing the reward—of being the true state. The basic idea behind this scheme is that rewards carry information about the true state as do observations, and hence should be used to estimate state probabilities. More precisely, the state-action values $Q_t(s, a)$, which here estimate the expected reward for action a (circle or square) in state s (light or dark) at time t (i.e., the discount factor $\gamma = 0$), are updated as follows:

$$Q_{t+1}(s, a_t) = Q_t(s, a_t) + \alpha \mathbb{P}(S_t = s | r_t, a_t, i_t, \boldsymbol{Q_t})(r_t - Q_t(s, a_t)) \tag{2.1}$$

where updates are for all possible states and only the chosen action a_t. α is the learning rate, i_t is the state that is identified by the learner (it might be different from the true state s_t), $\boldsymbol{Q_t}$ is the vector of all state action values (actually we could simply condition on the Q-values for the chosen action a_t, but we chose not to do so in order to avoid making notation cumbersome), and $\mathbb{P}(S_t = s|r_t, a_t, i_t, \boldsymbol{Q_t})$ is the posterior probability that the true state is s once the reward r_t has been observed, which using Bayes rule, can be re-written as:

$$\mathbb{P}(S_t = s|R_t = r_t, a_t, i_t, \boldsymbol{Q_t}) = \frac{\mathbb{P}(R_t = r_t|s, a_t, \boldsymbol{Q_t})\rho_t(s, i_t)}{\sum_{s'} \mathbb{P}(R_t = r_t|s', a_t, \boldsymbol{Q_t})\rho_t(s', i_t)} \tag{2.2}$$

where $\rho_t(s, i)$ is the probability with which the learner identifies i as the true state, such that her identification i is correct with probability ρ. More precisely:

$$\rho_t(s, i) = \mathbb{P}(I_t = i|S_t = s) = \begin{cases} \rho & \text{if } i = s, \\ 1 - \rho & \text{otherwise.} \end{cases} \tag{2.3}$$

Note that to obtain Equation 2.2, we assumed that *a priori* both states are equally likely to be the true state and to be identified by the learner as the true state (i.e., $\mathbb{P}(S_t = s) = \mathbb{P}(I_t = i) = 1/2$), and so we have $\mathbb{P}(S_t = s|I_t = i) = \mathbb{P}(I_t = i|S_t = s)$. $\mathbb{P}(R_t = r_t|s, a_t, \boldsymbol{Q_t})$ is the probability that the reward is r_t, given that the current state is s, the chosen action is a_t and the vector containing the current estimates of the average reward values for all state-action pairs is $\boldsymbol{Q_t}$. Since we are treating the simple case of a deterministic reward R_t (wining 2.5 pennies or nothing), R_t can be considered as a Bernoulli random variable, and hence the probability of each outcome can be estimated using $Q_t(s, a_t)$, the expected value of R_t, as follows:

$$\mathbb{P}(R_t = r_t|s, a_t, \boldsymbol{Q_t}) = \begin{cases} Q_t(s, a_t)/2.5 & \text{if } r_t = 2.5, \\ 1 - Q_t(s, a_t)/2.5 & \text{if } r_t = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

Using this expression, we can re-write Equation 2.1 in a more explicit way as:

$$Q_{t+1}(s,a_t) = \begin{cases} Q_t(s,a_t) + \alpha \frac{Q_t(s,a_t)\rho_t(s)}{\sum_{s'} Q_t(s',a_t)\rho_t(s')}(2.5 - Q_t(s,a_t)) & \text{if } r_t = 2.5, \\ Q_t(s,a_t) + \alpha \frac{(2.5-Q_t(s,a_t))\rho_t(s)}{\sum_{s'}(2.5-Q_t(s',a_t))\rho_t(s')}(0 - Q_t(s,a_t)) & \text{if } r_t = 0. \end{cases} \tag{2.5}$$

Until now, we have shown how the PWRL model updates the state-action values using a modified version of temporal difference learning that takes into account the uncertainty in the state identification. There remains the question of how the learner selects her actions given the estimated Q-values. Here, we assume that she does it according to the softmax selection rule (see Section 1.1.2). That is:

$$\mathbb{P}(A_t = a | I_t = i_t) = \frac{\exp(Q(i_t,a)/T)}{\sum_{a'} \exp(Q(i_t,a')/T)} \tag{2.6}$$

where T is the temperature parameter, which determines the degree of stochasticity in action selection. Note that Equation 2.6 shows how to compute the distribution of action selection given an identified state i_t, and not the true state s_t. In fact, the true state s_t is not known to the learner with certainty, but can only be correctly identified with probability ρ. In order to get the probabilities of selecting each action a marginalising out the true state s_t, we average out Equation 2.6 according to the probability of detecting each state given the true state, $\rho_t(s_t,i') = \mathbb{P}(I_t = i'|S_t = s_t)$. This gives:

$$\mathbb{P}(A_t = a|S_t = s_t) = \sum_{i'} \frac{\exp(Q(i',a)/T)}{\sum_{a'} \exp(Q(i',a')/T)} \rho_t(s_t,i') \tag{2.7}$$

where we assume that this is the softmax distribution that the learner uses to select actions.

To sum up, the steps followed by the learner in each trial under this model are as follows: First, she observes a noisy input from the environment (whose state can be correctly identified with probability ρ). Second, she selects an action according to the softmax choice function (see Equation 2.7). Third, the reward feedback is used to calculate the posterior probability of each state being the correct state (see Equation 2.2). Fourth, she updates the state-action values using Equation 2.5. This cycle is repeated with each new stimulus.

Although this model makes optimal use of reward feedback to get information about the current state before updating the state-action values, it can suffer from difficulties when a change in mapping rules is introduced as shown in Larsen et al. (2010). In fact, consider the following example, which is taken from the same paper and adapted to our case. Let us first denote by s_L and s_D the light and dark state respectively, and by a_L and a_D the correct "light" and "dark" response before the switch. Prior to the switch—in our experimental setting—the reward for choosing a_L in state s_L is 2.5, whereas it is 0 if a_D is selected in s_L. In trial 101, the rewards switch, so that $R_t(s_L, a_L) = 0$ and $R_t(s_L, a_D) = 2.5$, for $t > 100$. Assume that by $t = 100$, the model has learned the correct state-action values, so that $Q_{100}(s_L, a_L) = 2.5$ and $Q_{100}(s_L, a_D) = 0$. Suppose also that the true state at $t = 101$ is s_L, so that the probabilities of detecting each state as the true state are: $\rho_t(s_L, s_L) = 0.65$ and $\rho_t(s_L, s_D) = 0.35$ (from now on, we consider t to be equal to 101). Let's say the learner chooses at t what she believes to be the best action, i.e., a_L. Because the mapping rules have switched, her action would result in a null reward. Thus, the posterior probability that the state is s_L is given by (from Equations 2.2):

$$\begin{aligned}\frac{\rho_t(s_L, s_L)\,(2.5 - Q_t(s_L, a_L))}{\rho_t(s_L, s_L)\,(2.5 - Q_t(s_L, a_L)) + \rho_t(s_L, s_D)\,(2.5 - Q_t(s_L, a_D))} &= \frac{0.65 \times 0}{0.65 \times 0 + 0.35 \times 2.5} \\ &= 0 \qquad (2.8)\end{aligned}$$

This means that the learner would believe with certainty, after observing the null reward, that the true state is s_D rather than s_L, and hence would not change $Q_t(s_L, a_D)$, thereby continuing to use the outdated state-action mapping.

Note that, in this example, we have assumed that the estimated state-action values have converged to the correct ones to illustrate why the PWRL model might get stuck with using the same mapping after the switch. The model can potentially overcome this, for example, by using a very low value for the learning rate to avoid convergence.

The WTA-RL model

One simple approach to solve RL tasks with state uncertainty is to ignore the uncertainty and simply update the Q-value of the identified state in each trial. This is the basic idea behind the "the winner takes all" reinforcement learning (WTA-RL) model presented by Larsen et al. (2010). More specifically, the model assumes that, in trial t, the learner identifies the true state as i_t (via a sampling process that correctly identifies the true state with probability ρ), then updates the Q-value for only that identified state. Thus, in our experimental setting, the learner would update the right state-action value, in average, with frequency ρ. Under this scheme, the updating equation for the state-action values is given by:

$$Q_{t+1}(s,a) = \begin{cases} Q_t(s,a) + \alpha\,(r_t - Q_t(s,a)) & \text{if } s = i_t, a = a_t, \\ Q_t(s,a) & \text{otherwise.} \end{cases} \tag{2.9}$$

Using the state-action values, the learner selects actions according to the softmax selection rule defined in Equation 2.6. This model is hence similar to a SARSA model (see section 1.1.2) with the difference being that it updates $Q_t(i_t, a_t)$ instead of the Q-value for the current state s_t, $Q_t(s_t, a_t)$.

One advantage of this model is that it adapts well to a switch of contingencies, as it does not suffer from the problem of self-confirming beliefs encountered with the PWRL model. However, the WTA-RL model is not assured to converge to the correct state-action values (Larsen et al., 2010). Nevertheless, learning incorrect Q-values does not necessary result in incorrect actions, and hence the model can still be applied to solve learning tasks with state uncertainty.

Model evaluation

To compare the models in terms of how well they explained participants' choices, we estimated the free parameters of each of the two models separately for each participant from both the control and test group, using the maximum likelihood estimation procedure. More specifically, we selected model parameters that maximised the log-likelihood of observed actions conditioned on the previously encountered observations and rewards (Daw, 2011). The analysis of participants' learning curves indicated that a number of participants performed at a significantly higher level than 0.65, suggesting that their real level of state uncertainty was probably higher than it ought to be (i.e., $\rho = 0.65$). Therefore, we fitted the models with and without ρ as a free parameter, and found that considering ρ as a free parameter produced a better fit for both models and conditions, as judged by the BIC scores (see below for more details). Hence, we only present the fitting results with ρ as free parameter, along with the learning rate α and the exploration parameter T (the fitting results with ρ fixed at 0.65 are presented in Appendix A). We implemented the fitting procedure in Matlab using the constrained optimisation function fminsearchbnd (D'Errico, 2005), where ρ was constrained to lie between 0.5 and 1, and α between 0 and 1. To avoid finding local optima, we run the fitting procedure for each model with 30 randomly generated initial parameter values.

For the PWRL model, choice likelihoods could be computed directly using Equation 2.7. However, for the WTA-RL model, which includes a sampling step (when the learner identifies the current state), we did not have an explicit formula for computing the likelihoods. Instead, we had to estimate the probability of each observed response by simulating a one-step ahead model prediction of that choice a number of times, then recording the proportion of times the model accurately predicted that choice. More precisely, to approximate the likelihood of a participant's choice at time t, we provided the model with the sequence of the participant's responses, rewards and observations until $t-1$ along with the observation at time t, and used that to generate a model response. We repeated this 100 times, then computed the proportion of times the observed choice was accurately predicted by the model.

Finally, to compare the goodness of fit of the two models, we computed the Bayesian information criterion (BIC) for each subject and model using the model's best fitting parameters for that subject (Schwarz et al., 1978):

$$BIC_j = -2 \cdot ln(L_j) + k \cdot ln(N) \tag{2.10}$$

where L_j is the likelihood of the model with the best fitting parameters for the j^{th} subject, N is the number of trials (200 in our case) and k is the number of free parameters in the model (equal to 3 here). Note that the lower the BIC score of a model, the better it fits the data.

2.3 Results

We first analysed the behavioural results to see (1) how well participants learned the task in the control condition (and test condition prior to the switch) given the expected state uncertainty, and (2) whether they managed to detect the unexpected uncertainty in the test condition despite the presence of state uncertainty. We then fitted the behavioural data to each of the two computational models to determine which model best accounted for participants' choices across the control and test condition.

2.3.1 Learning performance under expected state uncertainty

Figure 2.5 shows that participants from both the control and test (before the switch) group performed, in general, at the level of the stimulus ambiguity, and that they were able to learn the correct state-action mapping despite the presence of state uncertainty.

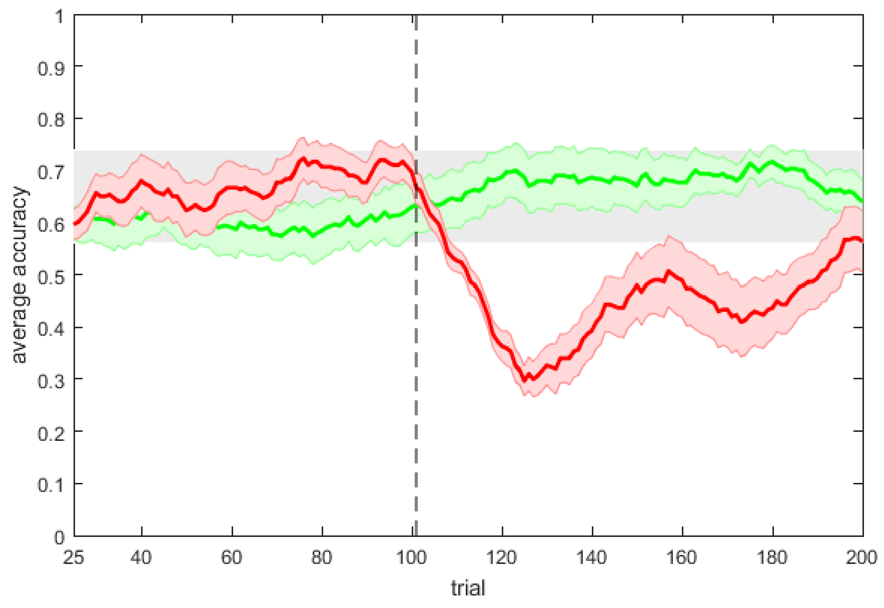


Figure 2.5: Running average score of the “average subject” in the control (solid green line) and test group (solid red line) with a window size of 25. The shading along each curve represents the standard error of the mean, while the light grey region represents a 95% confidence interval of the moving average of a Bernoulli variable with probability of success equal to 0.65 (this is to give an indication of the normal range of performance of a subject that has learned the task rules). The vertical dashed line indicates the trial where the switch occurs in the test condition.

This was confirmed at the individual level when we compared the proportion of correct responses on the first block of 20 trials to the proportion of correct responses on the last block of 20 trials preceding the "switch trial" (i.e., the 101th trial in both conditions). In fact, as can be seen from Figure 2.6A, before the "switch trial", 20 out of 24 participants from both groups showed improved performance by the end of the first 100 trials, as indicated by points above the equality line in the figure (note also that the performance of one participant did not increase in the last 20 trials because her accuracy was already above 75% in the first 20 trials). Also, 20 participants reached an accuracy above chance in the last block of 20 trials preceding the switch. This suggests that the majority of our participants, not only managed to learn the task (without the switch), but also were able to do it within 100 trials.

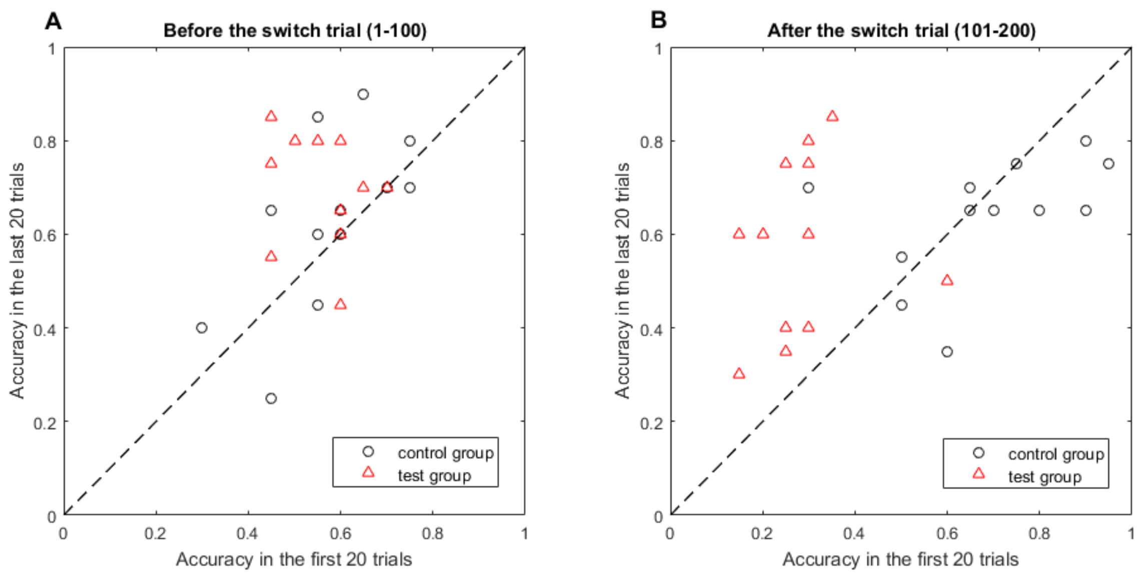


Figure 2.6: Comparison of average accuracy in the first 20 trials and last 20 trials before the "switch trial" (A) and after it (B), for both the control and test group.

2.3.2 Learning performance under unexpected uncertainty

Participants were also able, in general, to adapt to the unsignaled change in state-action mapping as illustrated in Figure 2.5 (solid red learning curve). As expected, accuracy dropped to the level $1 - \rho$ after the switch, and seems to have recovered slowly after it to arrive again at a level near of the level of state uncertainty $\rho = 0.65$. Figure 2.6B indicates that most participants from the test group (11 out of 12) showed improved performance in the last 20 trials in comparison with the first 20 trials following the switch; however, only 8 participants reached a performance above chance by the end of the testing sequence, and hence could be counted as successful in dealing with unexpected uncertainty.

Moreover, there were four different switch-response patterns (identified by eye; see Figure 2.7): (1) there were participants who could not recover at all from the switch (top left figure), (2) those who quickly adapted to the switch (i.e., within about 30 trials; see top right figure), (3) those who slowly adapted (i.e., did not recover until the very end of the test period; see bottom left figure) and (4) those who showed an oscillating learning behaviour (i.e., multiple relapse/recovery; see bottom right figure).

2.3.3 Comparing data with models

Let us now see which model explains better participants' choices in each condition (model fitting results are summarised in Table 2.1). We first compared the participants overall learning curve with that of each of the two fitted models (see Figure 2.8). To construct the learning curve of each model, we took the obtained fitted parameters for each subject and simulated the model 100 times. We then averaged those results to get a learning curve for each subject. We finally averaged across all subjects to get the overall learning curve of the fitted model. As can be seen from the figure, both models produced a good fit to participants' data. The WTA-RL model seems to give the closest overall fitted learning curve to that of participants in the control condition, whereas the PWRL model seems to be better in the test

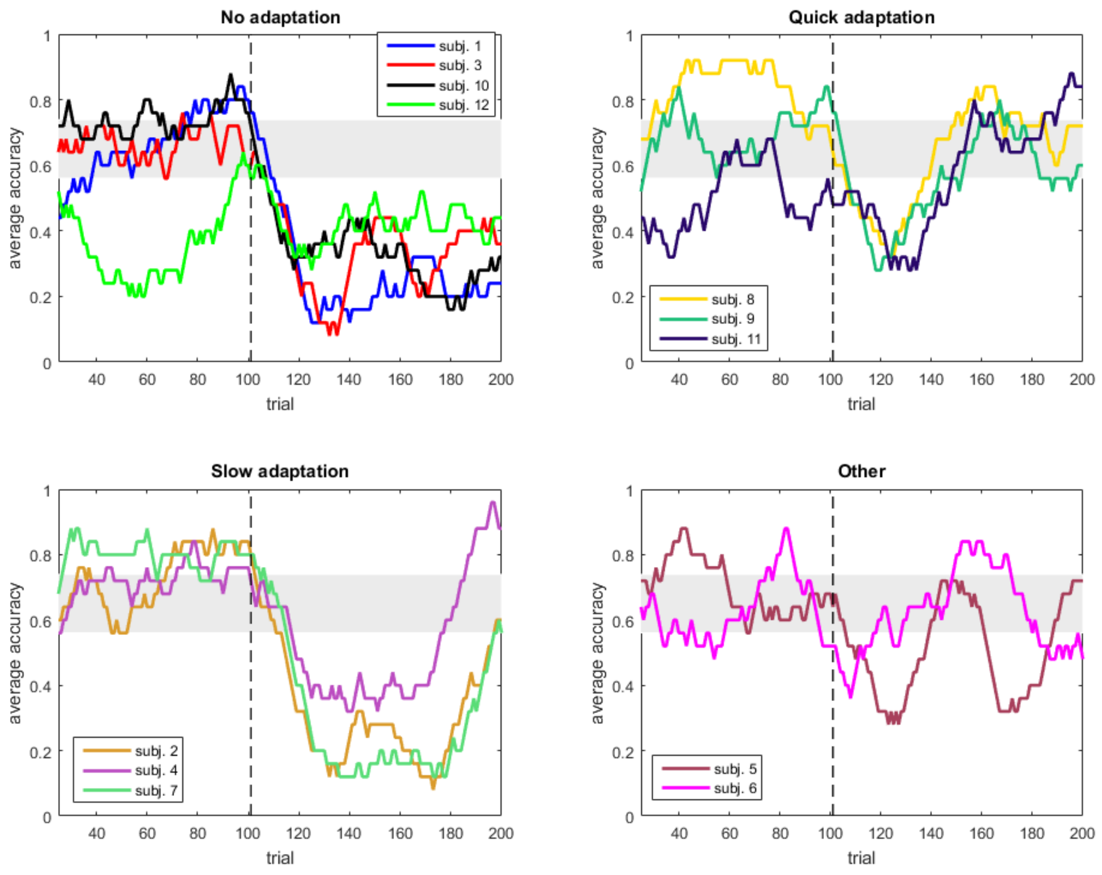


Figure 2.7: Individual learning curves categorised by the pattern of adaptation to the switch. The light grey region represents a 95% confidence interval of the moving average of a Bernoulli variable with probability of success equal to 0.65. The vertical dashed line indicates the trial where the switch occurs.

condition, especially in capturing the late recovery from the switch (see also fitted learning curves for each participant in Appendix A).

Table 2.1: The mean ± S.D. best-fitting parameter values by model and condition, along with the total BIC scores.

Model	Parameter	Constraint	Fit value (S.D.)		Total BIC	
			Control	Test	Control	Test
PWRL	α	$0 \leq \alpha \leq 1$	0.43 (0.39)	0.31 (0.35)	3049	3071
	T	$0 \leq T < \infty$	0.54 (0.88)	1.22 (1.20)		
	ρ	$0.5 \leq \rho \leq 1$	0.69 (0.15)	0.79 (0.17)		
WTA-RL	α	$0 \leq \alpha \leq 1$	0.29 (0.37)	0.23 (0.36)	2997	3086
	T	$0 \leq T < \infty$	0.67 (0.30)	0.72 (0.35)		
	ρ	$0.5 \leq \rho \leq 1$	0.79 (0.16)	0.84 (0.11)		

We then compared the BIC scores of the two models. Figures 2.9A-B show that WTA-RL was the model that explained participants' choices the best in both the control (8 subjects out of 12) and test (7 subjects out of 12) condition; however, the average BIC score favoured the PWRL model in the test condition. A more detailed analysis looking at the BIC scores by pattern of adaptation (Figure 2.9C-D)—based on the four-way classification of the test group participants presented in Figure 2.7—revealed that the PWRL model best fitted mainly those subjects who did not adapt (3 subjects out of 4) or slowly adapted to the switch (2 subjects out of 3), and none from the other two categories of learners.

We also checked the estimated parameter values to see if they can predict when a participant would adapt to the switch and how quickly (Table 2.2). We did not include the fitting results for one subject as the fitted value of their state uncertainty level ρ using both models was close to 0.5, meaning that the models considered that

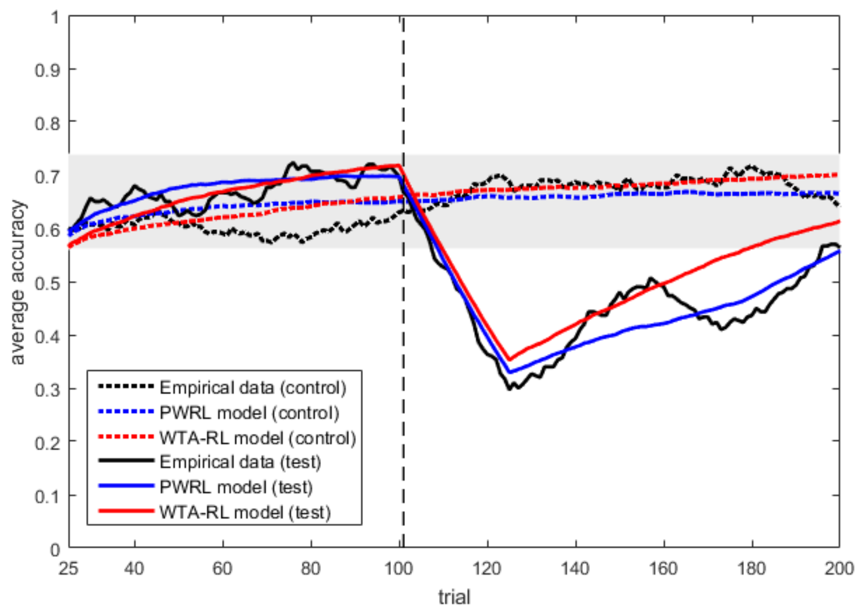


Figure 2.8: Comparison of learning curves of participants and fitted models. To draw model learning curves, we first constructed a learning curve for each subject by averaging over 100 simulations with her fitted parameters, then averaged across participants to get the overall learning curve. The light grey region represents a 95% confidence interval of the moving average of a Bernoulli variable with probability of success equal to 0.65. The vertical dashed line indicates the trial where the switch occurs in the test condition.

this subject was identifying states at random, and hence the resulting fitted learning rate and exploration parameter would be meaningless. The first striking finding is the large estimate values of the state uncertainty level ($\rho \geq 0.90$) obtained with the PWRL model for the "fast-adapter" group, and with the WTA-RL model for the "non-adapters" and "slow-adapters". Assuming that the psychometric function that we fitted in the preliminary phase is reliable, these large estimated values for ρ are unlikely to represent the true state uncertainty level of those participants. Instead, they are probably due to the two models having difficulties to fit those participants,

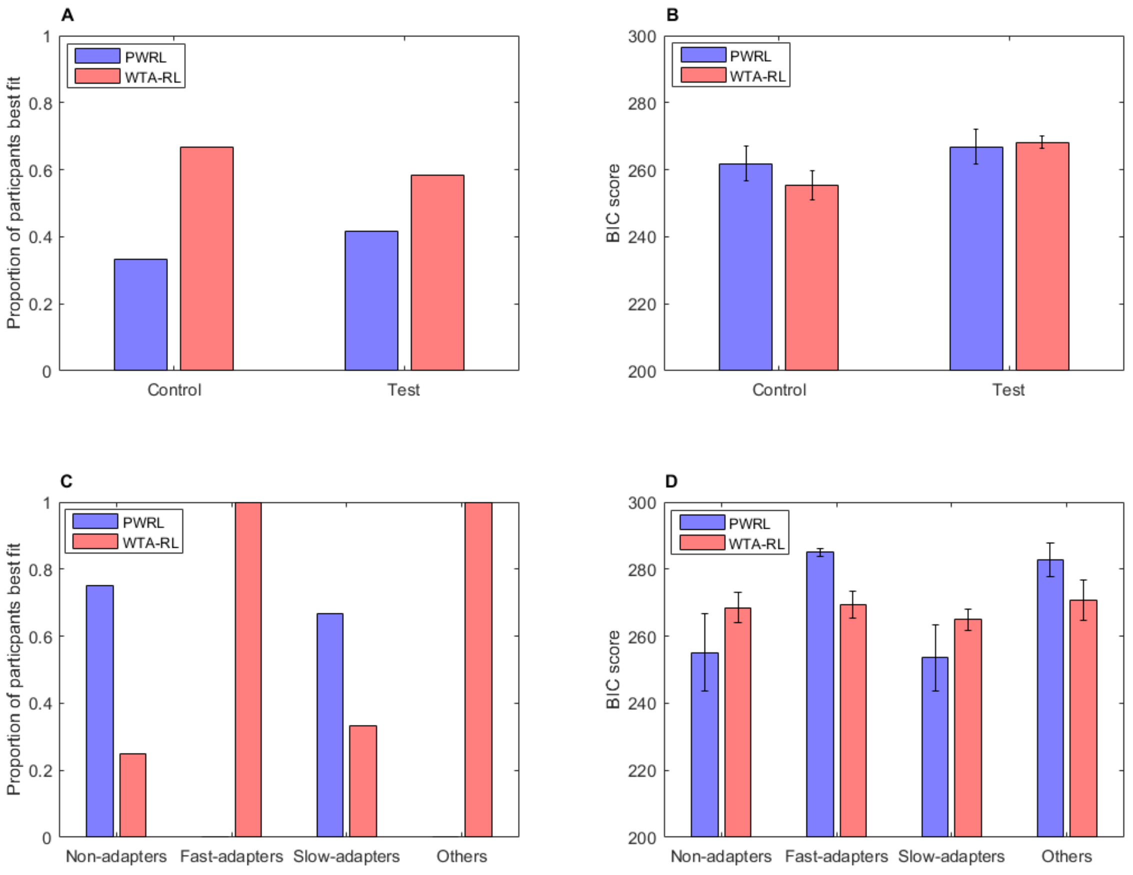


Figure 2.9: Best fit comparison. **(A)** Proportion of participants that each model fitted the best in both the control and test condition. **(B)** BIC scores by model and condition, with lower scores indicating a better overall fit. **(C)** Proportion of participants from the test group that each model fitted the best, by pattern of adaptation. **(D)** BIC scores by model and pattern of adaptation.

as our previous BIC scores analyses revealed—that is, PWRL is worse at fitting "fast adapters", whereas WTA-RL is worse at fitting the "non-adapter" and "slow-adapter" subjects (see also participants' individual learning curves in Figure A.3).

Another finding that is worth mentioning is the low values of the estimated learning rate α and exploration parameter T for the non-adapters in comparison with the slow-adapters (using PWRL—the best fitting model). The fact that the

Table 2.2: The mean ± S.D. best-fitting parameter values by model and category of adapter.

Model	Parameter	Non-adapters mean (S.D.)	Slow-adapters mean (S.D.)	Fast-adapters mean (S.D.)
PWRL	α	0.001 (0.001)	0.16 (0.10)	0.19 (0.038)
	T	0.002 (0.003)	0.22 (0.19)	2.56 (0.78)
	ρ	0.74 (0.068)	0.78 (0.016)	0.97 (0.019)
WTA-RL	α	0.017 (0.003)	0.027 (0.015)	0.092 (0.004)
	T	0.77 (0.11)	0.34 (0.14)	0.66 (0.27)
	ρ	0.96 (0.019)	0.90 (0.024)	0.81 (0.029)

learning rate was small for those who did not adapt is surprising since one would expect that a low learning rate means that participants are less likely to reach a point where they get stuck in the first place. In particular, it should help avoid the self-confirming belief problem that we highlighted in Section 2.2.5 (see Equation 2.8) by avoiding convergence to the true Q-values. However, a slow learning rate also means that the Q-values cannot be easily adjusted after the switch. The low value of T simply allows the learner to select the correct mapping despite the small difference between the Q-values for the two possible actions in each state (due to the small value of α).

2.4 Discussion

The results of our experiment suggest that the majority of the participants from the control group managed to learn the mapping between states of the world (light or dark) and actions (circle or square), under state uncertainty conditions (with level of state uncertainty at 65%). This tendency was confirmed when we looked at the learning performance prior to the switch of the test group participants. Furthermore, most participants managed to adapt to the switch (8 out of 12), but some did it very slowly. It is clear that in our experiment the expected uncertainty represented by the ambiguity underlying the states made it difficult for the participants to adapt to the unexpected uncertainty. These results are in line with previous studies showing that people are in general capable of dealing well with both expected and unexpected uncertainty (Behrens et al., 2007; Nassar et al., 2010; Wilson and Niv, 2011); however these studies used reward uncertainty instead of state uncertainty.

In an attempt to understand the underlying processes that human participants might have used to solve the tasks used in the present study, we compared two candidate models: a simple temporal-difference-based RL model (WTA-RL) and a Bayesian temporal-difference-based RL model (PWRL). Model fitting and model comparison favoured the simple RL model as a characterisation of participants' behaviour, though the PWRL model seemed to give a better overall account of data in the test condition. The PWRL model, as expected, mainly explained the choices of the participants who could not adapt to the switch. It also fitted the data of those who slowly adapted to the switch better than the WTA-RL model.

Qualitatively, the main difference between the two models is that the Bayesian RL model uses reward feedback to resolve the state uncertainty, whereas the simple RL model does not, and merely relies on the learner's perceptual system to correctly identify the state. Thus, one possible explanation for why some participants did not adapt—whose choices in the present experiment were best explained by the Bayesian RL model—is that they were using reward feedback to adjust their belief about the true state. On the other hand, those who quickly adapted to the switch probably did not use the reward feedback to gain information about the ambiguous state.

The fact that the estimated learning rate α by the PWRL model was small for the participants who did not adapt in comparison with those who adapted (it was also the case for the WTA-RL model, but this should be taken with precaution since the estimated state uncertainty level parameter was unexpectedly large) is consistent with Behrens et al.'s (2007) finding, which is that α should be large in volatile environments in order to perform well. Further investigation is still needed to discern between these two types of learning and study their effect on the ability of participants to learn under expected and unexpected uncertainty.

A body of research that is closely related to the present work comes from the study of the partial reinforcement extinction effect, or more explicitly, how using random rewards can affect overall learning performance and the persistence on using an option that is no longer the best option after extinction (Humphreys, 1939; Nevin and Grace, 2000; Hochman and Erev, 2013). For example, to test this effect, Hochman and Erev (2013) used an experiment with two phases: a training and an extinction phase, in both of which participants were asked to choose between two options (exactly like in a two-armed bandit task): a promoted option—one that produced the highest expected reward in the training phase, and which the experimenters wanted to test how long the participants would keep using during the extinction phase—and a demoted option, which becomes the most rewarding option in the extinction phase. The promoted option was rewarded by 1 or 17 with equal probability if selected during training (i.e., with mean reward of 9), and always rewarded by 1 during extinction, whereas the demoted option produced a reward of 8 when selected during both phases. Clearly, there are several similarities between this paradigm and ours. In fact, the described experiment also includes the two types of uncertainties studied in the present work: it has expected uncertainty due to the use of partial reinforcements, and unexpected uncertainty, which occurs at the transition from the training to the extinction phase. The promoted option corresponds, in our case, to the correct mapping prior to the switch, and the demoted option is the correct mapping after the switch. The main difference between the two paradigms is that we used state uncertainty whereas they used reward uncertainty, as in the previous studies that looked at expected and unexpected uncertainty (Behrens et al., 2007; Nassar et al., 2010; Wilson and Niv, 2011). Also, we had different states whereas Hochman and Erev (2013) had no context—only two actions to choose from.

This analogy opens a new set of possibilities using the predictions from the literature about the partial reinforcement extinction effect. One prediction that is likely to hold in our case is that the presence of expected state uncertainty should slow down the adaptation to the switch in comparison with the absence of expected uncertainty. This prediction comes from the observed positive effect of partial reinforcements (e.g., when the promoted option is rewarded probabilistically) on extinction relative to full reinforcements (when the promoted option is rewarded deterministically)—that is, the promoted option is found to be selected more often during extinction in a partial reinforcement schedule than in a full reinforcement schedule. In the present experiment, we studied the effect of unexpected uncertainty (via a switch in state-action response mapping) on learning performance under expected state uncertainty. Instead, to study the effect of state uncertainty on the ability to detect unexpected uncertainty, our experiment should be modified so that in both the control and test condition, the switch is introduced in the middle of the learning process, and in the control condition, the state uncertainty is removed.

Another factor that was found to affect the ability and speed of switch detection in Hochman and Erev's (2013) study is the relative attractiveness of the promoted option in relation to the attractiveness of the demoted option. It would be interesting to test if this would hold in our case—that is, if we can speed up the adaptation to the switch by increasing the reward value.

These two predictions from the partial reinforcement extinction effect literature (i.e., the presence of expected uncertainty or small relative reward between the promoted and demoted option slows down the adaptation to the switch) were well accounted for by a sampling model, the contingent sampling, which assumes that people choose options that produce the highest rewards in similar past experiences (Hochman and Erev, 2013). An exciting avenue for future work would be to test if this model (or other sampling models) can also explain the effects observed in our experiment better than the RL models.

Our work showed that humans are capable of learning under state uncertainty and that their performance was comparable to that of RL methods. However, other studies that employed more complicated partially observable tasks than ours reported that people can have difficulties in dealing with state uncertainty. For instance, Doshi-Velez and Ghahramani (2011) tested subjects on two classic tasks from the partially observable Markov decision process literature—the tigerworld task and gridworld task—and found that subjects learned slower and performed lower than three different RL models, even when subjects were given contextual information about the problem (i.e., when meanings were attached to observations and actions, rather than using abstract symbols). Moreover, all the models that they considered failed to account for subjects' choices, suggesting that human subjects were employing different strategies than the RL models. Stankiewicz et al. (2004) even showed that human participants were not able to learn a navigation task with state ambiguity due to perceptual aliasing (see Section 1.2.1). The fact that we found a different result is very likely due to the fact that we used a simpler partially observable task, with only two possible states and actions, rather than maze tasks that have at least a dozen of different states. In addition, the state uncertainty in those maze tasks was different in nature than ours as their state uncertainty arose from perceptual aliasing, and hence their uncertainty level was probably different as well.

To sum up, the present study provides evidence that humans can solve partially observable RL tasks when the partial observability is due to state uncertainty. Moreover, the state uncertainty does not seem to prevent humans from detecting an unexpected environmental change and to adjust their learning in response to it, confirming the previous findings that have looked at the same question from the angle of reward uncertainty. In the rest of the thesis, we turn to the second case of partial observability, which is when the state depends on past information, and hence some sort of memory is needed.

PART II:

Partial observability due to memory dependence

Chapter 3

Actor-Critic gating model: limitations and possible enhancements

3.1 Introduction

It is common in the real world that humans and animals have to learn to act in environments where the optimal action depends not only on the current environmental state, but also on information encountered in the past. For example, to understand what we are reading, we often need to maintain recently processed information to connect it with the most recent information from the text. We also need to hold the gist of what we are reading to construct an overall understanding of the text (Swanson and O'Connor, 2009). Animals searching for food have to retain the location of bad patches that they visited to avoid returning to them again (see for example Olton and Samuelson's (1976) experiment). Several models have suggested that working memory (WM) supports reinforcement learning (RL) in such partially observable Markovian environments (Braver and Cohen, 2000; O'Reilly and Frank, 2006; Zilli and Hasselmo, 2008; Lloyd et al., 2012). These models could learn what information is important for subsequent behaviour (to maximise reward), so needs to be actively maintained in WM, and what information is irrelevant, so should be discarded, while at the same time learning to behave in the external world.

One such model is the Actor-Critic gating model of Todd et al. (2009), which has the advantage of being psychologically plausible—as a temporal difference-based model inspired by the gating framework— while being less complex than the classical neural-network-based gating models. More interestingly, Todd et al. (2009) showed that their model was not only capable of solving challenging WM learning tasks like the 12-AX, but it also exhibited limitations similar to those observed in humans on an artificial grammar learning task.

However, one apparent problem with their model—and gating models in general— is that they take a long time to train. For example, Todd et al. (2009) reported that they run the 12-AX task for twenty millions time steps to make sure to get convergence, and that the model took several hundreds of thousands of trials to reach their learning criterion. This questions its plausibility as a model of human learning (there is also a practical issue as the model usually takes a long time to simulate, which might discourage other researchers from using it). Hence, there is a need to improve this model if we want to test it against human performance, and to achieve that, we need to first identify what makes the model learn so slowly.

In this chapter, we take that first step by identifying three major problems with the current framing of the Actor-Critic gating model. We also propose for each identified problem either a complete solution, or introduce briefly a possible way to overcome it, in which case we pursue it in more details in a separate chapter. This chapter can hence be seen as an introduction to the next chapters, each of which proposes a modification or a new approach to improve the Actor-Critic gating model. Most of the proposed enhancements can also be used with other gating models, especially that we take care that they make the Actor-Critic gating model more psychologically and neurally plausible, or at least do not affect its plausibility as a model of human behaviour.

3.2 General methods

3.2.1 The 12-AX task

The 12-AX has been one of the most popular tasks for testing WM-based RL models (Frank et al., 2001; O'Reilly and Frank, 2006; Todd et al., 2009). In this task (Figure 3.1), digits are presented (1 or 2), interspersed with letters (A, B, C, X, Y or Z). The task is structured such that responses to the letters X or Y are rewarded depending on the most recent digit 1 or 2, and the most recent letter A, B or C. The rules of the task to be learned are as follows: If the last digit was 1 (resp. 2), the last letter from the set $\{A, B, C\}$ was A (resp. B), and the current observation is X (resp. Y), then the correct response is R, which is rewarded with 1. In all other cases, the correct response is L, rewarded 0.25. Reward is 0 for incorrect responses.

The sequence of observations in the task is generated as follows: we start by sampling equiprobably one digit 1 or 2. This marks the beginning of an outer loop, which defines the subsequence of observations delimited by two digits (includes only the first digit). Each outer loop is composed of one to four inner loops—pairs of letters A, B or C followed by X, Y or Z, such that at least half of the inner loops are potential targets AX or BY. The remaining pairs are drawn randomly from the 9 possible inner loops (e.g. AY or CZ). The number of outer loops to use depends on the number of observations needed.

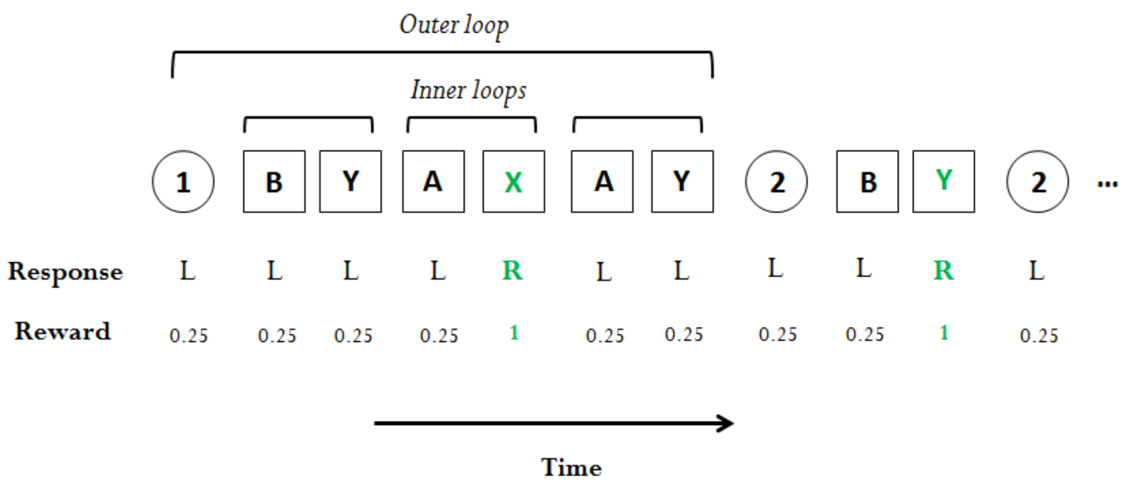


Figure 3.1: A sequence of observations from the 12-AX task. The correct response and the associated reward is given below each observation. The green colour highlights the end of the target sequences ("1-A-X" and "2-B-Y"), where the learner should select the 'R' response.

To perfectly solve this task, the learner must keep track of the last digit 1 or 2 and the last letter A, B or C independently. Here we assume that the learner does this using her WM (the assumption that WM can be used to keep track of such cues will be tested in a behavioural experiment in Chapter 6).

3.2.2 Actor-Critic gating model

To solve the 12-AX and other WM learning tasks considered in this chapter, we use the WM-based RL model of Todd et al. (2009). This model—as already mentioned in Section 1.2.2—was inspired by both the early idea of supporting RL with memory to solve partially observable/non-Markovian tasks from the machine learning literature (see for example, Littman, 1994), and the more recent biologically plausible gating models of type PBWM (Prefrontal cortex Basal ganglia Working memory; O'Reilly and Frank, 2006), which show how WM gating functions are learned in the brain.

Todd et al.'s (2009) model uses WM to track past states, so as to make the most relevant ones part of the present learning context when making decisions. More importantly, WM functions such as what to store, when to store and how to use the stored information in WM are all assumed to be learned via an RL process, along with the motor actions (e.g., lever press, verbally report an item). For example on the 12-AX task, since keeping track of the digits is rewarding in the long-term as it informs about which response to select, the model would potentially learn to store them in WM whenever they are encountered and maintain them until a new digit is presented.

Formally, the model is based on the Actor-Critic architecture with eligibility traces, and uses softmax rule for action-selection (see Section 1.1.2). As such, the model has three major components (Figure 3.2):

Environment. The environment is assumed to provide the learner, at each time, with an observation (e.g., a letter or a digit as in the 12-AX) and a reward feedback. To remain faithful to the unified RL formalism, WM is also assumed to be part of the environment. This does not mean that WM is considered external to the learner; it only means that the information residing in WM is now part of the state that the learner sees at each time step. WM is viewed as an array composed of a fixed number of slots (or cells), each of which can contain one observation. The state is hence defined by the current observation o and the current WM content $(m_1, m_2, \dots, m_C)$, where m_i is the item held in slot i and C is the total number of WM slots (to simplify we refer to C as WM capacity).

Actor. The role of this module is to choose actions via two types of agents: one motor agent, which is responsible for choosing motor actions, and C gating agents, each of which controls one WM cell. These gating agents act independently of each other and can choose one of two possible gating actions: Maintain (M) the content of its WM cell, or Update (U) it with the current observation o (i.e. replace its content with o). Actions are selected based on the state-action values of each agent. Here, they are selected probabilistically following a softmax rule (described in more details below), such that actions with the highest values are selected more often.

Critic. This component evaluates the performance of the actor by calculating an error signal δ—that is, the deviation between the expected reward and the obtained reward— after each reward feedback. This signal is received by the actor, which uses it to re-estimate the state-action values.

More specifically, at time t, the steps followed by the learner in this model are as follows (for a summary see Table B.1 in the appendix):

First, the learner observes the current state, s_t, then selects a motor action a_t and C gating actions $g_{1,t}, \ldots, g_{C,t}$, each corresponding to one WM cell. The motor and gating actions are chosen probabilistically, based on the motor and gating state-action values Q^M and $Q^G_1, \ldots, Q^G_C$ respectively, and in accordance with the softmax selection rule. More precisely, the corresponding probabilities are given by:

$$p(s,a) = \mathbb{P}(a_t = a|s_t = s) = \frac{\exp(Q^M(s,a)/T)}{\sum_{a'} \exp(Q^M(s,a')/T)} \tag{3.1}$$

$$p(s,g_i) = \mathbb{P}(g_{i,t} = g_i|s_t = s) = \frac{\exp(Q^G_i(s,g_i)/T)}{\sum_{g'_i} \exp(Q^G_i(s,g'_i)/T)}, \quad \forall i = 1,2,\ldots,C \tag{3.2}$$

where T is the exploration parameter. To avoid redundancy and to simplify our equations, henceforward we will write the equations only for one gating agent; the equations for the other gating agents being the same. For example, if we use a gating action g in a given equation, this will refer to all gating actions $g_1, \ldots, g_C$.

Once the agent has chosen a motor action and gating action, these actions are carried out, and the reward r_t and next state s_{t+1} are observed (note that the reward

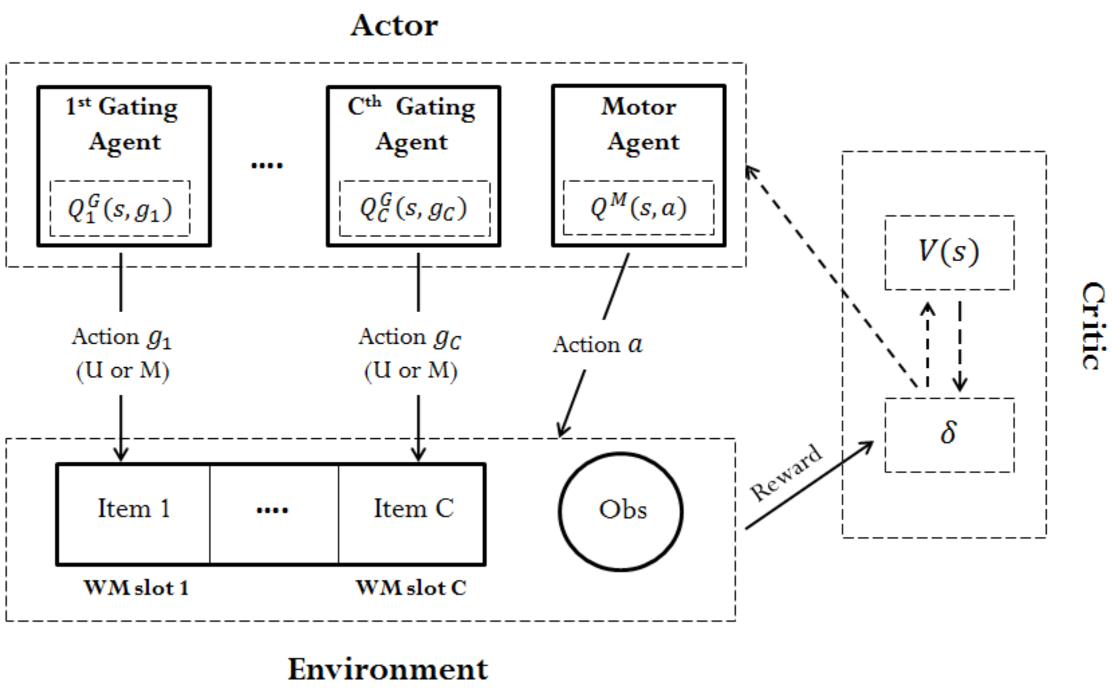


Figure 3.2: Todd et al.'s (2009) gating model based on the Actor-Critic architecture. The actor component is where the actions (gating and motor) are chosen based on the state-action values Q^M (motor) and Q^G's (gating). After actions have been taken, the environment sends a reward signal to the critic, which calculates an error signal δ. The actor then uses this error signal to update the motor and gating state-action values.

is dependent only on the selected motor action), which are then used to calculate the temporal-difference error signal δ_t:

$$\delta_t = r_t + \gamma V_t(s_{t+1}) - V_t(s_t) \tag{3.3}$$

where γ is the discount-rate parameter, and $V_t(s_t)$ and $V_t(s_{t+1})$ are respectively the estimated values of state s_t and s_{t+1} at time t.

Third, the learner updates the motor, gating, and state eligibility traces e^M, e^G and e^V as follows:

$$e^V(s) = \begin{cases} \gamma\lambda e^V(s) + 1 & \text{if } s = s_t, \\ \gamma\lambda e^V(s) & \text{otherwise.} \end{cases} \tag{3.4}$$

$$e^M(s,a) = \begin{cases} \gamma\lambda e^M(s,a) + 1 & \text{if } s = s_t, a = a_t \\ 0 & \text{if } s = s_t, a \neq a_t \\ \gamma\lambda e^M(s,a) & \text{otherwise.} \end{cases} \tag{3.5}$$

$$e^G(s,g) = \begin{cases} \gamma\lambda e^G(s,g) + 1 & \text{if } s = s_t, g = g_t \\ 0 & \text{if } s = s_t, g \neq g_t \\ \gamma\lambda e^G(s,g) & \text{otherwise.} \end{cases} \tag{3.6}$$

where $0 \leq \lambda \leq 1$ is the decay parameter, where we have used an accumulating trace function in which the trace of the non chosen action was set to 0 (this manipulation was introduced by Singh and Sutton, 1996, in the context of a replacing trace to improve learning performance). Todd et al.'s (2009) paper, on the other hand, used the following trace function, which is not common in the RL literature:

$$e^M(s,a) = \begin{cases} 1 - p(s,a) & \text{if } s = s_t, a = a_t \\ -p(s,a) & \text{if } s = s_t, a \neq a_t \\ \gamma\lambda e^M(s,a) & \text{otherwise.} \end{cases} \tag{3.7}$$

where $p(s,a)$ is the softmax probability of selecting action a, the gating trace function has exactly the same form, and the state eligibility trace has an accumulating trace form. Our experimentations with different types of trace function—including a replacing trace—suggest that the choice of the trace function has only a small effect on learning performance. Note, however, that it is important here to use eligibility traces to be able to successfully learn to select the gating actions, whose consequences can only be measured further down the line (reflected by the amount of reward that will be received).

As a final step, the state and state-action values are updated using the calculated temporal-difference error, eligibility trace values, and the learning rate α :

$$V(s) = V(s) + \alpha\delta_t e^V(s) \tag{3.8}$$

$$Q^M(s,a) = Q^M(s,a) + \alpha\delta_t e^M(s,a) \tag{3.9}$$

$$Q^G(s,g) = Q^G(s,g) + \alpha\delta_t e^G(s,g) \tag{3.10}$$

Let us now show how the 12-AX can be solved using the presented framework. As presented in Table 3.1, one possible solution is to reserve two WM cells for the task: one cell to remember the last digit (1 or 2) and the other cell to remember the last letter (A,B or C). For example, if the first cell is used to keep track of the last digit, then the first gating agent updates this cell whenever it observes a digit. After that, it maintains this knowledge until it encounters a new digit. This ensures that the digit stored in this WM cell represents the last observed digit. Concerning the motor actions, they should be selected following the rules of the task (for example, with (1,A) in WM and X being the current input, the learner should select 'R'). This is the most obvious solution to the 12-AX task, so we will refer to it as the expected policy.

Table 3.1: Expected solution for the 12-AX task using the Actor-Critic gating framework. Gating and motor actions are given for each state (current observation and WM contents). $m1$ and $m2$ refer to the memory items that are stored respectively in the first and second WM cell, and which could represent any of the possible observations.

Observation	*WM state*	*1st Gating action*	*2nd Gating action*	*Motor action*
1, 2	**m1 | m2**	**U**	**M**	**L**
A,B,C	**m1 | m2**	**M**	**U**	**L**
X	**1 | A**	**M**	**M**	**R**
Y	**2 | B**	**M**	**M**	**R**
X,Y	**m1 | m2**	**M**	**M**	**L**

3.2.3 Model settings

For all the simulation results presented below, we show the model behaviour under the best set of parameters (α, T and λ) on each task. In most cases, to find the set of optimal parameters for each model, we did a grid search over a wide range of possible parameters values, and selected those parameters that maximised the asymptotic reward rate (i.e., the average reward in the last n trials, where we varied n depending on the time required to run the task) across 30 simulations. However, we sometimes conducted the parameter search using as low as one simulation in cases where the simulations took a long time to run (for example, it took us about one hour to complete each simulation run using the original model of Todd et al. (2009) on an Intel Core i7 computer). Also in each case, we started by doing a quick grid search in the parameter space to get an idea about the region containing the set of best parameters. Once we got closer to the optimum, we varied the parameters on a finer scale and ran the 30 simulations per parameter setting. Note that when the asymptotic reward of two parameter sets were the same (for example, when the model could reach perfect accuracy for many sets of parameters), we used a learning speed measure called time-to-criterion to select the best set of parameters, as explained below. In all simulations, we set the discount parameter γ to 0.95, and initially set all Q-values to 0. Also, we randomly initialised the content of each WM cell to one of the possible observations (we did this to avoid adding an extra symbol to represent an empty WM cell, which would have increased further the dimensionality of the problem). Table B.2 lists the parameters that we used to simulate each model on each task.

3.2.4 Analyses

Learning curves

The main method that we used to assess performance was to construct learning curves. We plotted them differently from the previous chapter; here, we computed the average accuracy over blocks of k trials, where k was chosen in each case, so that there were exactly 20 disjoint blocks in total. One advantage of using learning curves is that they not only give a measure of model accuracy, but also give an idea about how fast the model learned the task.

Time-to-criterion

Another way to evaluate model learning speed is to calculate the number of time steps necessary to reach a certain level of accuracy, or what is often referred to as time-to-criterion (see for example, Krueger and Dayan, 2009). In most cases, we used the criterion of 0.95 (i.e., proportion of correct responses) in 500 successive target trials X or Y. We mainly used time-to-criterion in the parameter selection process, although we also used it in the last section of this chapter.

3.3 Results

3.3.1 The problem of using independent gating agents

Simulation results using Todd et al.'s (2009) model

In an attempt to replicate Todd et al.'s (2009) result that their Actor-Critic gating model could successfully learn the 12-AX, we simulated their model on the 12-AX using the same parameters given in their paper. That is:

- WM capacity: $C = 2$ (we also simulated the model without WM and with a WM capacity of one for comparison purposes).
- Learning rate: $\alpha = 0.1$.
- Discount rate: $\gamma = 0.94$.
- Decay rate: $\lambda = 0.75$ (according to the paper, the model could learn the task with at least two other values of λ, but they found that learning was faster with $\lambda = 0.75$, as judged by the time-to-criterion).
- Exploration parameter: $T = 10$ (the exact value of T used in the original paper was not given, so we tried different values of T, then chose the best value).
- Number of time steps: $N = 2 \times 10^7$.

We found that the model could not reach optimal performance in any of our simulations using these parameters, neither could it reach the criterion of 300 successive correct responses that Todd et al. (2009) used in their paper. So why haven't we been able to reproduce their results, although we used exactly the same parameters as in the original study?

Before answering this question, we first tested the possibility that there are other parameters that can make the model perfectly learn the 12-AX. Thus, we did a parameter search to try to find a set of parameters that can solve the task, but this time using a number of trials N equal to 10^7 (this is mainly because the simulations were very slow to run and that this number should be enough to allow the model to fully learn the task, since the original paper reported a time-to-criterion of less than 3×10^5 trials). Figure 3.3 depicts the learning curves obtained using the parameters that produced the highest performance (i.e., largest reward rate) for the models with zero, one and two WM cells. The two-cell gating model performed better than the pure RL model (i.e., without WM) as reported in Todd et al.'s (2009) paper, but could not reach the maximal accuracy level, and even performed lower than the one-cell gating model. This was not only the case in average, but also true in all the 30 individual simulations.

Given that we could not find any set of parameters that can get the Actor-Critic gating model to perform optimally as reported in the original paper, we tested the possibility that the problem might come from the framing of the model itself. In fact, one key aspect of Todd et al.'s (2009) model that might explain our findings is that it has two independent gating agents, each of which controlling one WM cell. Thus, the model needs to learn, among other things, to coordinate between the two gating agents (for example, to not update both WM cells at the same time, as it will record the same information twice). To see if the model was successful in doing so, we tracked WM contents in the last 20 trials of each simulation run as illustrated in Table 3.2.

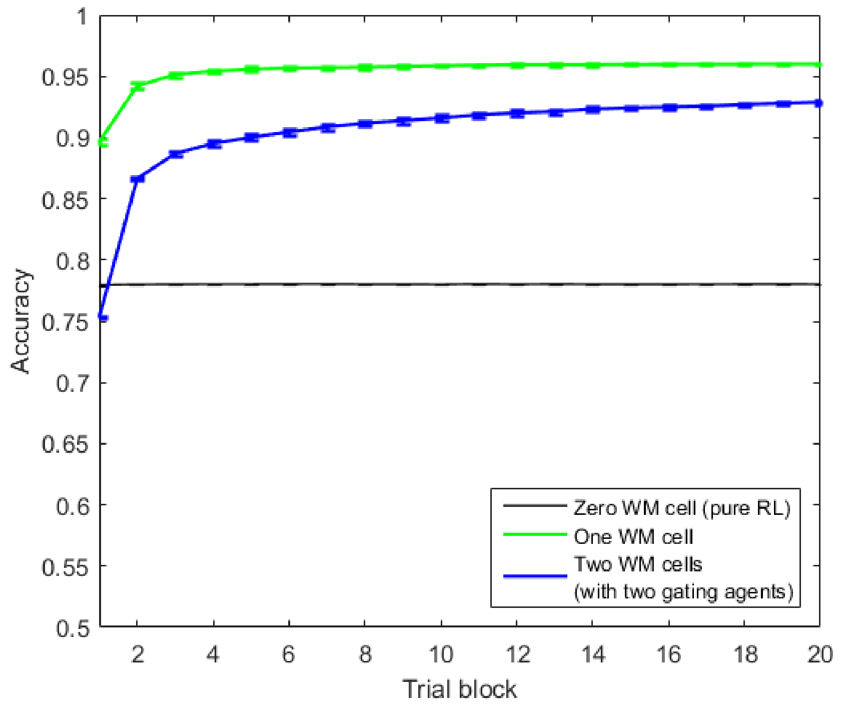


Figure 3.3: Learning curve of the Actor-Critic gating model with two independent gating agents (blue line). The accuracy was averaged over blocks of 5×10^5 trials and over 30 runs. The figure also includes the learning curves of the pure RL model (black line) and the one-cell Actor-Critic gating model (green line). Error bars represent standard errors.

Table 3.2: WM snapshot from one simulation with the two-cell Actor-Critic gating model to illustrate the coordination problem between the two gating agents.

WM cell 1	B	Y	2	C	Y	A	X	2	2	X	C	X	B	Y	C	C	A	2	2	C
WM cell 2	B	Y	2	C	Y	A	X	2	2	X	C	X	B	Y	C	B	A	A	A	C

We found that, in the 30 simulations completed, the mean number of trials in which both WM cells contained the same item during the last 20 trials was 19.33 and the standard deviation was 1.31 (see Figure 3.4). Moreover, in all simulations, WM contained redundant information during at least 15 out the last 20 trials, and contained redundant information during all the 20 trials in 18 simulations. This clearly indicates that the two gating agents learned, most of the time, to store the same information in their respective WM slots, suggesting that they failed to coordinate their updating actions.

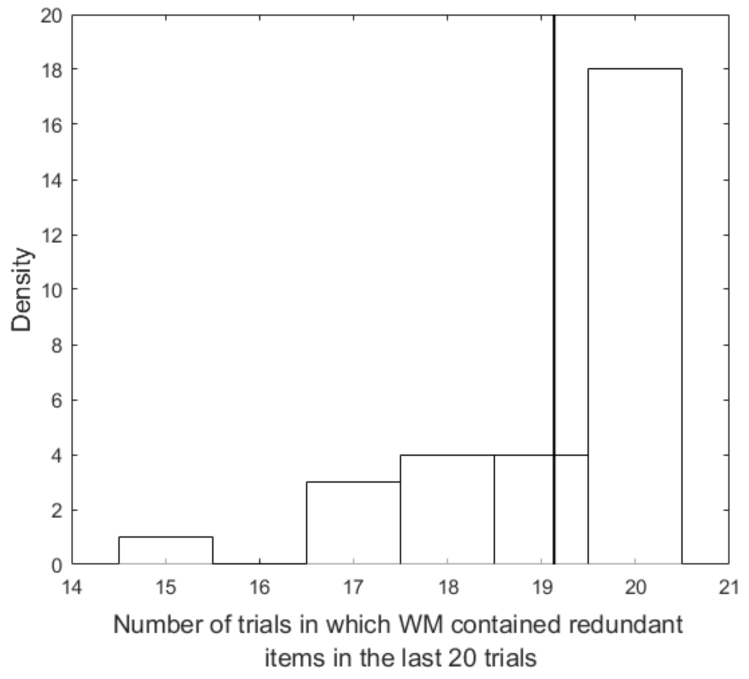


Figure 3.4: Histogram of the number of trials in which WM contained redundant items in the last 20 trials for the Actor-Critic model with two gating agents. The thick line represents the mean.

Another finding that is worth highlighting here is that the model performed very well with a WM capacity of one (> 95%). This might explain why the two-cell model mimicked the learning of a one-cell model, by using its two WM cells as if it only had one cell. In fact by doing so, the two-cell model could learn very good suboptimal policies. One such policy consists of compressing the information about the two cues (1,A) (resp., (2,B)) into A (resp., B), thereby allowing the agent to keep track of both cues using only one WM slot. Basically under this policy, the gating agent stores the digit cue 1 (resp., 2) once it has encountered it (i.e., choose "Update" response). Then, it updates again only if the letter cue is A (resp., B). Therefore, A (resp. B) in WM would mean that the last digit and letter cue are 1 (resp. 2) and A (resp. B). Any other item in WM would mean that the sequence is not a target one, and hence the "L" response needs to be selected. This was possible because the occurrence of the cue letter A (resp. B) is only relevant if 1 (resp. 2) was the last digit, and not 2 (resp. 1).

In the following, we will propose a modification of the current Actor-Critic gating model, capable of solving the coordination problem between the two gating agents, which should allow the model to perfectly learn the 12-AX task.

Alternative model: use of one gating agent

Our simulations showed that the Actor-Critic gating model as described in Todd et al. (2009) could not reach optimal performance when applied to the 12-AX task. We identified one problem that might be the cause of this restriction, which is the failure of the two gating agents to coordinate their updating actions, thereby resulting sometimes in redundant information in WM. To overcome this, we propose here to use a single gating agent—instead of multiple ones—that controls all WM cells and only updates one WM cell at a time.

In this new version of the Actor-Critic gating model (Figure 3.5), the possible gating actions become: Update the j^{th} WM cell while maintaining the other cells (U_j; $j = 1 \cdots C$), or Maintain all WM cells (M), for a total of $C + 1$ possible gating actions. This ensures that the model will never store the same item in two different

WM cells. Note that this modification of the Actor-Critic gating model does not change the general form of its updating equations; the only difference is that the updates should now be applied to one gating agent with three possible actions, instead of having one updating equation for each agent.

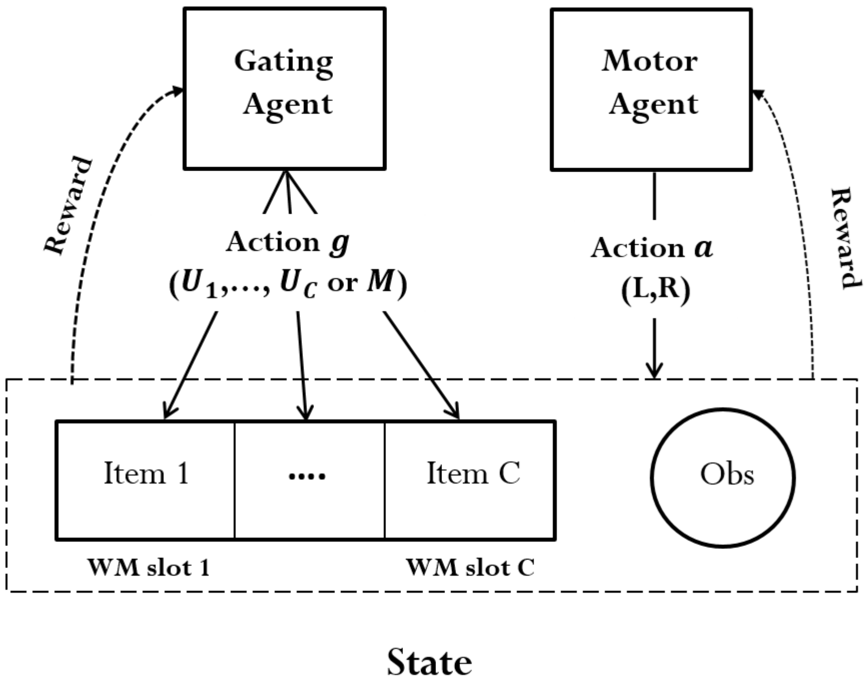


Figure 3.5: Actor-Critic gating model with one gating agent (referred to as the RLWM model) controlling a WM capacity of C.

Figure 3.6 compares the learning curves of the two architectures (one versus two gating agents). The one-gating version of the model reached almost perfect accuracy in the last block (approximately 99.8%), which was substantially better than the accuracy that we obtained using the original two-gating model of Todd et al. (2009) (approximately 92.9%). Thus from now on, we will use this new version of the Actor-Critic gating model (i.e., with one gating agent), which we will call the RLWM (Reinforcement Learning with Working Memory) model.

A noteworthy finding from our simulations is that the pure RL outperformed

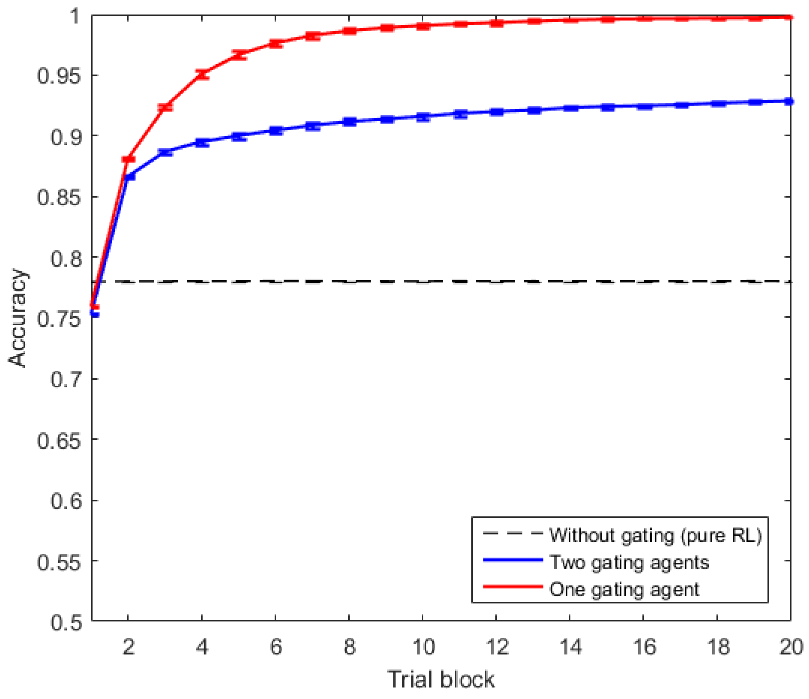


Figure 3.6: Comparison of the learning curves of the RLWM model with two independent gating agents and the RLWM with one gating agent. The accuracy was averaged over blocks of 5×10^5 trials and over 30 runs. Error bars represent standard errors.

both versions of the Actor-critic gating model with two WM slots during the early stage of the 12-AX task. For example, the pure RL model was more accurate than the RLWM model for at least 250000 time steps (see Figure 3.6). So, considering the lifetime of most animals (humans included), which only allow them to do limited practice, the 12-AX does not seem to be diagnostic of the involvement of WM in real environments. However, this only happens because of the nature of the 12-AX, in which good performance can be achieved using simple Markovian strategies, such as always choosing the "L" response (for example, an agent that responds "L" all the time is expected to get about 63% of possible expected reward and to achieve an expected accuracy of about 87%). We may well design tasks that more strongly demand the learner to learn to use latent states in WM than the 12-AX.

This is exactly what we do next. More precisely, we introduce two simplified versions of the 12-AX: the first, called the 12-XY, has only four observations (1, 2, X or Y), and only requires a WM capacity of one to be solved. The second task is very similar to the 12-AX, except that it does not contain the distractor stimuli C and Z and does not require a response on cue trials 1, 2, A or B. We refer to this second task as the 12-AB-XY. The two tasks are structured such that a model without access to information from previous observations would perform at around chance level, and hence, we can easily assess the usefulness of adding a WM component to the Actor-Critic model. In addition, we would expect people to learn the new tasks relatively quickly, and hence these tasks might be better candidates to be used in human experiments. Also from a technical perspective, they make simulations easier to run and analyse, whilst still maintaining the challenging partially observable nature of the 12-AX.

3.3.2 The problem of using a fixed working memory capacity

Most previous demonstrations of WM-based RL considered a fixed WM capacity that is set before the start of the learning task and kept constant over the course of the task (see for example, Zilli and Hasselmo, 2008; Todd et al., 2009; Lloyd et al., 2012). In addition, the tendency in those studies was to show that the proposed WM-based RL models can achieve high asymptotic performance. Here, we are not just concerned with final performance, but also with learning speed, and how performance and speed are affected by the WM capacity allocated to the task at hand. For that, we simulate the RLWM model with different WM capacities using two simplified versions of the 12-AX, and analyse the effect of WM capacity on the accuracy and speed of learning.

Effect of working memory capacity on learning in the 12-XY task

As a first step to evaluate the effect of WM capacity on learning with the RLWM model, we used a very basic version of the 12-AX task, where we kept only the observations 1, 2, X and Y (Figure 3.7). The task is structured such that responses to the letters are rewarded depending on the last digit seen. More specifically, the rules to be learned are as follows: if the last digit was 1 (resp. 2) and the current letter is X (resp. Y), then choose the 'R' response. In all other cases, choose 'L'. Reward is either 1 for correct responses or 0 for incorrect responses. (No reward was given in response to actions when the current observation was a digit). This task is, hence, very similar to the classical AX-CPT task (Cohen et al., 1999b) in that the correct response to the letters X and Y depends on one set of cues (here the digits 1 or 2).

Note that in this task, we do not need to use the notion of inner loop since there are no cue letters; inner loops are simply the letters X or Y instead of pairs of letters. We also use the same definition of outer loop—that is, an outer loop consists of one to four letters X or Y, along with the digit that precede them.

Given that the 12-XY requires the learner to retain only one cue at a time, we can solve it by using a WM capacity of one. As shown in Table 3.3, a good policy to learn (expected policy) is to keep track of the digit cues—that is, update WM content only when the current observation is 1 or 2. The motor actions should then follow the rules of the task (e.g., choose 'R' if the current observation is X and WM contains 1).

We simulated the new task using the one-cell RLWM model with $N = 50000$ trials per simulation (Figure 3.8). We also included simulation results using the

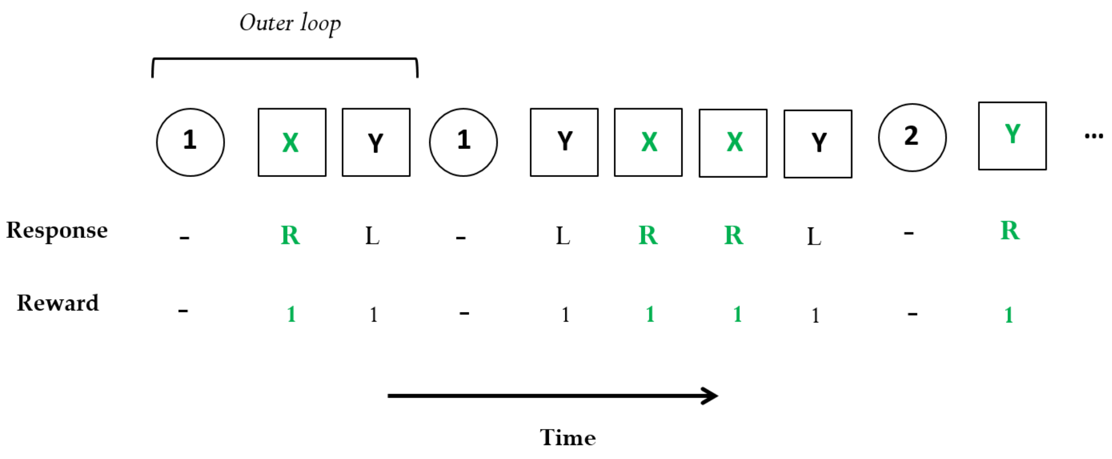


Figure 3.7: A sequence of observations from the 12-XY task. The correct response and the associated reward is given below each observation. Here, outer loop size is equal to the number of letters X and Y between two successive cues plus one (to count the first cue). The coloured letters X and Y are used to show where the 'R' action is required.

Table 3.3: Expected solution for the 12-XY task using the RLWM with one cell. Gating and motor actions are given for each observation and WM content. m refers to the memory item stored in WM, and which could represent any of the possible observations.

Observation	*WM state*	*Gating action*	*Motor action*
1, 2	$\boxed{m}$	**U**	**-**
X	$\boxed{1}$	**M**	**R**
X	$\boxed{2}$	**M**	**L**
Y	$\boxed{1}$	**M**	**L**
Y	$\boxed{2}$	**M**	**R**

RLWM model with two WM cells to see whether the model can still solve the task with an extra WM cell, and how increasing WM capacity affects learning speed (We did not include the pure RL model since there was no Markovian policy that could solve the task). Although, in many WM tasks, a larger WM capacity is assumed to be beneficial, here we predict that with a larger WM capacity, learning becomes slower, because the state-action space becomes larger, which results in more contingencies between states and actions to learn.

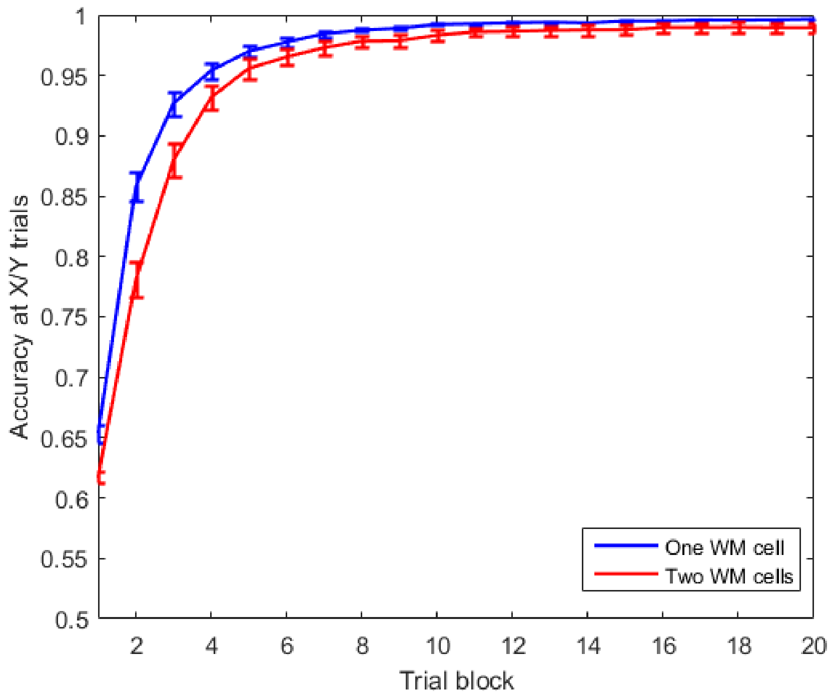


Figure 3.8: Comparison of the learning curves of the two-cell RLWM model and the one-cell RLWM model. The accuracy was averaged over blocks of 2500 trials and over 30 runs. Error bars represent standard errors.

As expected, the one-cell RLWM learned to perform optimally in the task, and did it in a significantly shorter time in comparison with the original 12-AX task (around forty thousands trials as opposed to more than eight millions of trials). More importantly, it substantially performed above chance during the whole task, and hence performed better than the pure RL. This gives us a case where learners would perform poorly if they rely on contingencies that depend only on the present, but would do better if they use WM to take into account past observations. With the larger WM capacity, the model also reached optimal accuracy, but took longer as we expected. It is clear that there is no advantage of using more WM resources as the one-cell model performed better during the whole task. However, this is not always the case as we will see when we will introduce the other modified version of the 12-AX.

Effect of working memory capacity on learning in the 12-AB-XY task

Now that we evaluated the effect of WM capacity on a basic WM learning task with one class of cues to keep track of (digits 1 or 2), let us examine this effect on a task that requires at least two WM cells to be perfectly solved. We ask the same question as in the previous analysis: how does the overall performance (taking into account both accuracy and speed) of the one-cell RLWM model compare to that of the two-cell model?

The rules of the new task are similar to the original 12-AX, except that, again, there is no action required and no reward given in cue trials (Figure 3.9). We also used a similar structure to the original task—that is an outer loop size that varied between one and four.

As with the previous task, we simulated this one using a WM capacity of one and two, with each simulation run consisting of $N = 10^6$ trials (we increased the number of trials to allow enough time for learning). Figure 3.10 compares the learning curves of the one-cell and the two-cell RLWM model. With two WM cells, the model had a better asymptotic average accuracy than with one WM cell (0.99 versus 0.9); however the one-cell model reached its top accuracy level faster, and

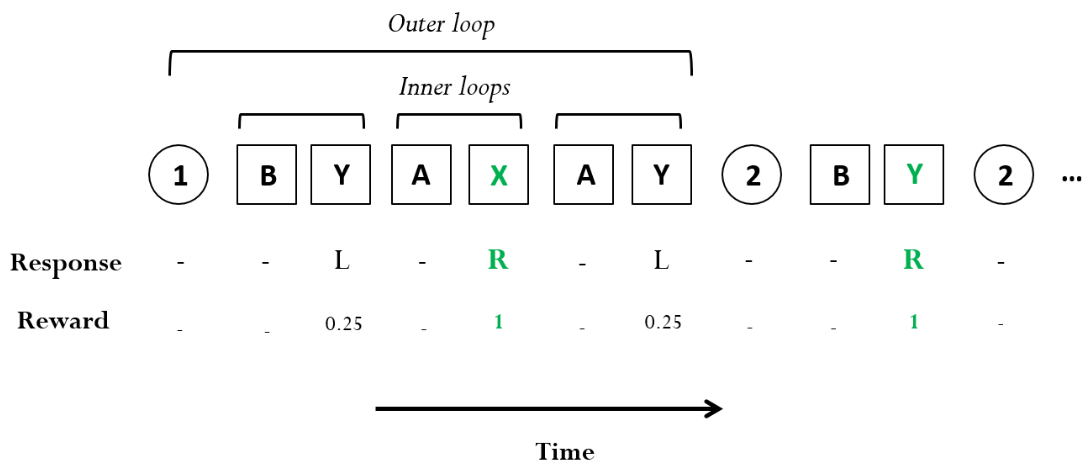


Figure 3.9: A sequence of observations from the 12-AB-XY task.

more interestingly, outperformed the two-cell model during the first 300 thousand trials.

This suggests that learners in the 12-AB-XY task are faced with a trade-off between learning speed and learning accuracy; depending on what is more important—speed or accuracy—they should either use a lower WM capacity if speed is more important or a larger WM capacity if accuracy is more important. But what if both speed and accuracy are important to the learner?

The current assumption of a fixed WM capacity does not seem to allow the learner to meet both objectives. We have seen that in both simulated tasks—the 12-XY and 12-AB-XY—learning is faster when the WM capacity is chosen to be small, but might be incomplete as in the 12-AB-XY. On the other hand, the learner could achieve optimal accuracy by choosing a larger WM capacity, but at the expense of slowing down learning.

Another issue with using a fixed WM capacity is how the learner would know the right WM capacity to use at the beginning of the task. Even worse, the fixed WM capacity assumption does not allow gating models to smoothly generalise to Markovian tasks. In fact, it would be nonsense to use the RLWM model with a WM component in a task that has Markovian rules. In fact, although the model might end up learning the correct rules, learning would certainly be substantially slower than with a simple RL model.

The most obvious alternative to the assumption of a fixed WM capacity is to allow the learner to flexibly use its WM resources. One way to do that is to gradually allocate WM resources to the task at hand; the learner would start with a low WM capacity—even as low as zero (i.e., use a pure RL mechanism)—then would increase the WM capacity as learning progresses in the task depending on some measure of learning success. This is the basic idea behind a model that we will introduce in the next chapter.

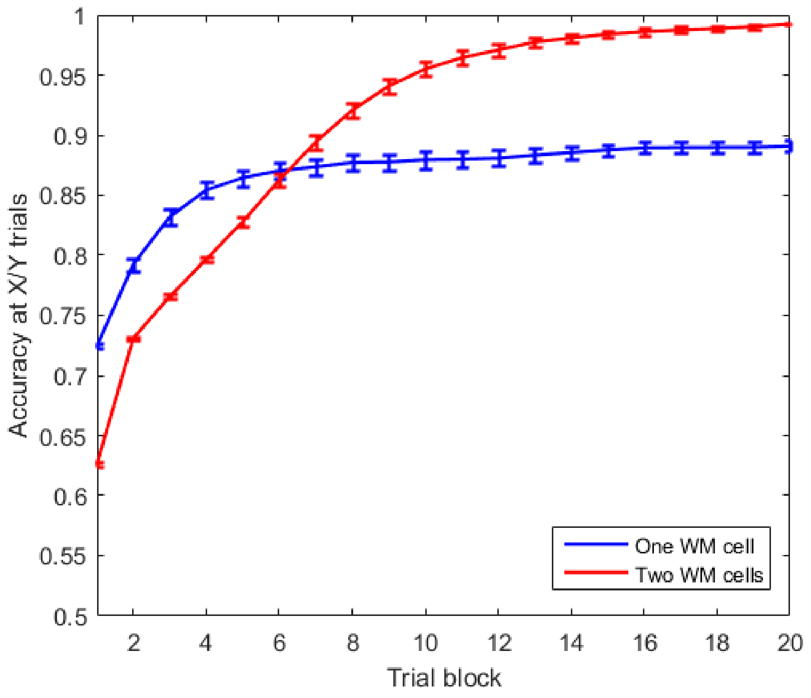


Figure 3.10: Speed-accuracy trade-off in learning the 12-AB-XY task. The figure compares the learning curve of the one-cell RLWM model with that of the two-cell model. Accuracy is averaged over blocks of 5×10^4 trials and over 30 simulation runs.

3.3.3 The problem of non-coordination between the gating and motor agent

Since the Actor-Critic gating model has multiple agents (several gating agents in addition to the motor agent), it suffers from the same problems and challenges of multiagent RL systems. Among them are the issues of stability of an agent's learning and adaptability to changes in other agents' behaviour (Busoniu et al., 2008). For instance, the gating agent(s) needs to provide the motor agent with a stable context (via the gating policy) in order for the motor agent to be able to learn the appropriate behaviour. Similarly, learned gating actions have no value if they are not accompanied with good motor responses. Thus, it is central to coordinate the behaviour of the different agents in order to successfully learn with gating models.

In section 3.3.1, we showed how to avoid the problem of coordination between the gating agents by simply using one gating agent that controls all WM cells instead of multiple gating agents. A related issue is how to structure the learning between the gating and motor agent in the RLWM model. Our purpose here is to simply introduce the coordination problem between the gating and motor agent, and to demonstrate to what extent this problem can slow down learning with the RLWM model (and gating models in general). Possible solutions to this problem will be discussed in the "Discussion" section.

To give an idea about how much this problem negatively affects learning under the RLWM model, we compared the learning time—using time-to-criterion—of the current version of the RLWM model (without any coordination between the gating and motor agent) on the 12-AX against the cumulative time required to learn the gating policy alone—the correct motor policy being given and equal to the expected optimal motor policy (see Table 3.1)—and to learn the motor policy alone—the gating policy being given and equal to the expected optimal gating policy. We used such cumulative learning time because it would reflect learning speed if learning were perfectly structured between the two agents. In fact, the ideal scenario would be that the model first learns one policy—gating or motor—then once it has been done, starts learning the other policy. Although this cannot be perfectly done in practice, it indicates how much improvement in learning speed can still be achieved by appropriately structuring the learning between the gating and motor agent. Note here that we simulated the separate learning of the gating policy and that of the motor policy using the same parameter values as those used for the two-cell RLWM model (see Table B.2).

Figure 3.11 shows that there is a major shortfall in learning speed due to the coordination problem between the gating and motor agent; an ideal gating model where learning is structured between the two agents would be at least 7 times faster on the 12-AX than the current version of the RLWM model. The comparison results also show that learning the motor policy was substantially faster than learning the gating policy. Thus, one idea to coordinate the learning of the two agents would be to fix the gating policy (for example, by randomly generating it at the starting of the task), while trying to learn the motor policy. The learner would then turn to adjust the gating policy whenever it finds that the learned motor policy cannot achieve optimal performance. The advantage of using a similar approach is to avoid updating the wrong type of policy by updating both agents' policies. In fact, a lack of reward can be wrongly attributed to a correct motor policy rather than an incorrect gating policy and vice versa.

3.4 Discussion

Techniques based on RL have been used to solve a wide range of learning problems. However, most current RL techniques work only in Markovian settings and hence,

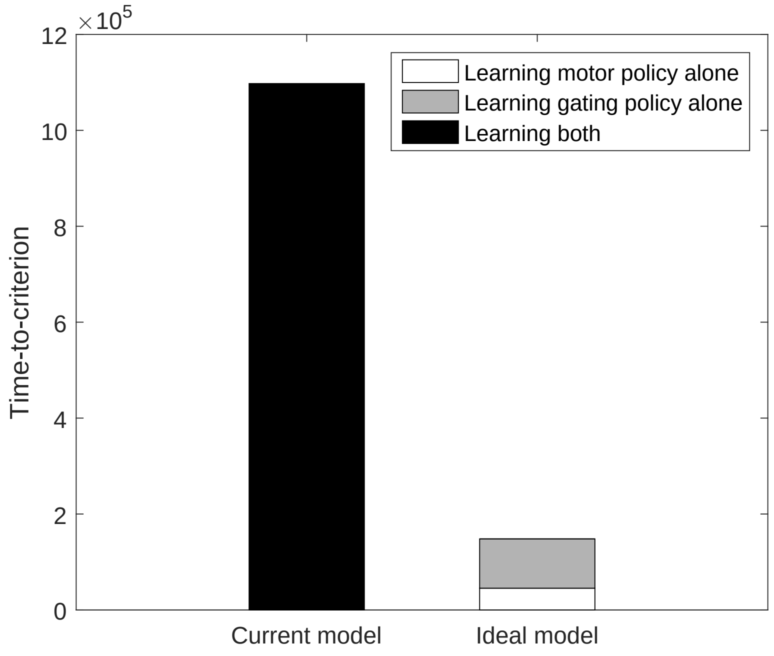


Figure 3.11: Margin of improvement in learning speed that can be achieved by coordinating the gating and motor agent. Time-to-criterion represents the number of trials required to reach an accuracy level of 0.95% in 500 successive X/Y trials. All models were simulated using the best parameters that we found for the RLWM model in Section 3.3.1.

cannot be applied directly to solve the learning problem of WM learning tasks. WM-based RL models such as the one proposed by Todd et al. (2009) propose a solution to this problem by transforming the non-Markovian interface between the learner and the environment into a partially observable Markovian one.

This solution is compelling both from a psychological and a mathematical perspective. Psychologists have been interested in these models because they propose an explanation for how people learn to use their WM (under the gating framework; Braver and Cohen, 2000; Rougier et al., 2005; O'Reilly and Frank, 2006). From a mathematical perspective, they can be used to solve partially observable Markov decision processes in general, with the possibility to maintain the convergence properties of the original RL methods. However these models suffer from several shortcomings; we identified at least three in this chapter: two are related to the multi-agent nature of WM-based models and one to the use of a fixed WM capacity.

The first problem is due to the use of multiple gating agents, all acting in a parallel and modular fashion, and thus ending up acting redundantly. As a way to overcome this problem, we introduced a variant on Todd et al.'s (2009) framework in which all WM actions are carried out by a single agent. This new assumption not only solved the redundancy problem, but also seems more plausible from a neural and psychological perspective. In fact, the modification is in accordance with the central bottleneck view of WM (Garavan, 1998; Rohrer et al., 1998). For instance, Oberauer (2002) proposed that WM is controlled by the focus of attention system, which is limited and cannot perform parallel operations as updating WM cells independently.

The second problem arises because the model uses reward feedback to update both motor and gating state-action values at the same time. This is ineffective, because lack of rewards can be wrongly attributed to a correct motor policy rather than an incorrect gating policy and vice versa. A better approach would be to allocate the observed reward to one agent only following some defined criteria for detecting gating and motor errors, then update only the state-action values of that agent. A second way to deal with the coordination problem between the gating and motor agent, which was proposed by Lanzi (2000) (see also Lanzi, 1998), would be to reduce the gating exploration relative to that of the motor agent. Although they showed that this solution improved learning in perceptual aliasing problems, they did not explain why this manipulation produced a better performance. One possible explanation is that a low internal memory exploration ensures that the motor agent has a WM context stable enough to allow the motor agent to learn the appropriate motor responses. Another possible solution, which is similar to the approach we took to resolve the redundancy problem in WM, would be to consider a single agent that simultaneously takes both gating and motor actions. This is somewhat similar to the assumption used by Zilli and Hasselmo (2008) in their gating model; however, they only allowed the agent to take either a gating or a motor action at a time.

The third issue that we identified is that WM capacity is assumed to stay constant throughout the learning task and that is must be chosen at the beginning of it. This deals poorly with the tradeoff between learning speed and learning performance. In fact, while in some WM learning tasks, a small WM capacity is enough to support optimal learning during the whole task (as in the 12-XY), in other tasks, the use of a smaller capacity is better only at the initial stage of learning, and hence more WM resources are necessary to perform optimally (as in the 12-AB-XY), but at the expense of slowing down learning.

We expect that a learner that gradually allocates its resources can better deal with this trade-off: the agent starts with few WM resources, then increases them if it cannot improve any more its performance using the current resources. The next chapter will explore this idea by putting in place a model that allocates WM resources gradually rather than using a fixed WM capacity during the whole task. We expect people to use a similar strategy, as it has lower cognitive costs, but at the same time, boosts initial learning.

Our simulations also showed that the 12-AX task is not particularly diagnostic of the involvement of WM, since good performance could be achieved without the use of WM (i.e., with a pure RL model) or with just one WM slot. The fact that the WM-based model performed almost optimally with a WM capacity of one (accuracy higher than 0.95%) is surprising since the 12-AX is often presented as a task that requires a WM of at least capacity two (Todd et al., 2009; Krueger and Dayan, 2009)—being an extension of the CPT-AX task to actually create more demands on WM by requiring the learner to separately maintain the digit (1 and 2) and letter (A and B) cues against interference rather than just the letter cues. Our analyses of the obtained policies with a one-cell WM revealed that the RLWM model was able to achieve such a high accuracy by compressing the information about the two cues (1,A) (resp., (2,B)) into a single item, and hence was able to keep track of both cues using only one WM slot (see Section 3.3.1).

As alternative tasks, we used two simplified versions of the 12-AX, where a pure RL model performed at chance level, while the RLWM model performed optimally. These tasks not only highlighted the importance of WM in supporting RL, but also were easier to learn than the original task, and hence are better candidates to be tested with humans. However, even on the simplified versions of the 12-AX, learning with the RLWM model was very slow, especially when the task required more than one WM state. Other factors that could slow down leaning are the number of possible observations and the number of cues. Hence for example, the 12-XY task was the easiest to learn, since it has only four possible observations (1,2,X and Y) and only two cues (1 and 2) (see Gorski, 2012, for a a detailed analysis about the effect of task characteristics, such as the number of observations and the distance between successive cues, on the learning performance of WM-based RL models).

Note that the Actor-Critic gating model has other shortcomings that may not be addressable by simply modifying its parameters or settings. For example, the RLWM model has a lot of parameters that need to be set carefully in order to get convergence to an optimal policy. In particular, it is crucial to allow the model to explore enough but not excessively (i.e., balancing the trade-off between exploration and exploitation). Solving the exploration-exploitation trade-off is especially critical in gating models, as the addition of a WM component to the agent results in a sizeable state-action policy space. Bayesian RL techniques propose a principled way of dealing with this trade-off and often have less parameters to set up, and hence can be used as an alternative to the classical RL approach. Given these advantages, Chapter 5 will introduce a Bayesian model-free RL method capable of solving WM learning tasks, and compare its performance with classical WM-based RL models (see also Ez-zizi et al., 2015).

In summary, one noticeable feature of all WM-based RL models of which we are aware—including the ones considered here—is that they take a very long time to learn. This questions their plausibility as realistic models of human or animal performance. In particular, these models are very incremental, and are slow to adapt on the basis of feedback about the structure of the environment. This also means that these models will not perform well in situations where the environment is dynamic. One potential solution to be explored in the next chapter is to use a WM with an adaptive capacity instead of a fixed one.

Chapter 4

Working memory-based reinforcement learning with adaptive working memory capacity

4.1 Introduction

Working memory-based reinforcement learning (RL) models such as the Actor-critic gating model of Todd et al. (2009) often assume that working memory (WM) capacity remains fixed over the course of WM learning tasks. This deals poorly with the trade-off between learning speed and learning accuracy as we highlighted in the previous chapter (see Section 3.3.2), and limits the learner, in some cases, to either optimise learning speed by allocating a low WM capacity to the task, or to optimise learning accuracy by allocating a large WM capacity. Another problem with this rigid assumption is that the WM capacity to use should be selected manually before the onset of the task. However, it seems implausible that humans would make their mind about the WM capacity to allocate throughout the whole task before embarking on it.

To balance the speed-accuracy trade-off, we suggested in the previous chapter to add more flexibility to the RLWM model, by allowing it to increase its allocated WM capacity while learning, instead of using the same WM capacity throughout the whole task. The present chapter sets forth to test this idea by constructing a new version of the RLWM model that has an adaptive WM capacity, and comparing its performance with the original RLWM model. The main underlying assumption behind this new version of the RLWM model is that using WM is costly, and people would not allocate high WM resources unless it helps them improve performance. In addition, a learner that uses too much WM resources might be at a disadvantage since learning can become slow, as we saw in the previous chapter.

This is not without precedent as other studies have pointed out the advantage of starting small in computational models of learning. For example, Elman (1993) implemented a similar approach under a recurrent neural network model to simulate language acquisition in young children, and showed that his model could not fully learn the rules of a complex artificial grammar learning task unless the model's memory was gradually enhanced. However, his model was based on supervised learning (i.e., learning from a set of training examples, each including the input along with desired output) rather than RL, and did not have a separate memory component to store inputs from the task (the memory in his model simply arose from the recurrent connections in the hidden layers of the neural network; for more details about recurrent neural networks, see for example, Lipton, 2015).

Krueger and Dayan (2009) highlighted the importance of the gradual allocation of memory resources in the context of shaping in RL. The need to increase the memory of their neural network arose as the agent was presented with new elements of the learning task. To ensure a successful transition to a new subtask, they proposed two allocation methods: manual and automatic. In the manual case, they added one fresh memory block to the network at the onset of each new task, but this implied that the learner either knows when a new task starts or relies on an external supervisor to know when to allocate more memory resources. In contrast, in the automatic case, they introduced an automatic criterion for detecting the need for extra memory allocation, based on the learner's current error rate. Thus in this version, the learner can add more than one memory block within a single subtask.

Along similar lines, we also propose here a manual and an automatic version of resource allocation under the RLWM model. However, unlike Krueger and Dayan (2009), we are interested in applying our model even in cases where the environment is stationary, as opposed to only environments where rules change, for example, due to shaping. In the manual allocation, we assume that the learner increases its WM resources at specific times (could be either predetermined by the learner before the task or indicated by an external supervisor). We used manual allocation for the same purpose as in Krueger and Dayan (2009), which is to focus on the potential benefits of gradual WM resource allocation without worrying too much about implementation details, and to build upon it to construct the final version of the model—the automatic resource allocation model. In this final version, the learner is left to decide how to handle its WM resources; the learner might decide to increase them as a way to refine its current behavioural policy, or keep them fixed if it is satisfied with its current performance level.

In the following, we provide details about how we implemented the gradual resource allocation approach under the RLWM model, and apply the resulting model to the 12-AX task. We chose to use the 12-AX here rather than, for example, the 12-XY or 12-AB-XY task, because it better illustrates the trade-off between speed and accuracy, as there are benefits and costs in terms of learning time and long-term accuracy to using each level of WM capacity, including a null capacity. We hence expect the resource allocation model to take advantage of the benefits of learning with each WM capacity in the 12-AX. Our initial findings show the potential of the proposed approach to improve learning performance in WM-based RL models.

4.2 General methods

4.2.1 Resource allocation model: RLWM with adaptive working memory capacity

To overcome the aforementioned limitations of using a fixed WM capacity under the RLWM model, we extend the model by allowing WM capacity to vary over the course of a learning task. The general idea of our approach is that the learner gradually allocates its WM resources to the task based on its current accuracy level, starting with no allocated WM resources—that is, relying on a pure RL mechanism—then, increasing the allocated WM capacity step-by-step to improve performance. More specifically, the agent first tries to learn the task without the use of WM. If it fails, it then transitions to an RL mechanism with one WM cell. Again if the learner cannot solve the task, the currently allocated WM capacity is increased by a second WM cell. This process is repeated until the agent has completely learned the task. To apply this approach, we need to define a specific criterion to decide when to increase WM resources. For that, we follow Krueger and Dayan's (2009) procedure, and try two methods for resource allocation: manual and automatic.

In the manual case, we assume that WM resources are augmented at specific times—that is, the first increase happens on a certain trial N_1, the second on trial N_2, and so on. This might, for example, represent a case where an external supervisor tells the learner (e.g. a robot) when it is time to increase WM resources (for example by monitoring the learner's performance), or a case where an individual has prior information about the task at hand, so plans the number of time steps she needs to spend learning with each WM capacity. The real purpose, however, behind using this simple allocation procedure is to focus on the effect of gradually allocating WM resources on learning performance, without worrying to much about the details of how to decide when to increase the WM capacity.

In the automatic case, we introduce an accuracy-based criterion for when to augment the WM resources. More precisely, we assume that the learner keeps track of its current accuracy rate Acc_n—that is, the average accuracy in the current block n of the previous w trials (for example, with $w = 1000$, the learner would compute the average accuracy after every 1000 trials). Then, at the starting of a new block $n + 1$, the learner computes the new accuracy rate Acc_{n+1}, and decides to augment its allocated WM resources if the difference between the new and previous accuracy rate is lower than a small number ϵ (for example with $\epsilon = 0.01$ and $w = 1000$, the learner would need to check if the accuracy has not changed by more than 1% during the previous 1000 trials). That is:

$$|Acc_{n+1} - Acc_n| < \epsilon \tag{4.1}$$

Put simply, the learner uses this equation to tell whether performance has levelled out or not under the current WM resources.

When this criterion is satisfied (in the case of automatic allocation) or when it is simply time to augment WM resources (in the case of manual allocation), the next step for the agent—in addition to increasing the allocated WM capacity—is to transfer what has been learned so far with the lower WM capacity to the new augmented policy. In this way, the agent avoids relearning the task from scratch. For instance, if on the 12-AX task, the agent has learned to choose "L" when the current observation is "Z" during stage 0 (to simplify, we refer to the stage in which the agent uses i WM cells as stage i), then it should keep using this rule when the allocated WM capacity is increased to one (i.e., choose "L" when the current observation is "Z", regardless of the content of the allocated WM cell). One way to accomplish this is to retain the same previously estimated state/state-action values after the transition to a higher WM capacity. Thus for example, if in the previous stage, learning was performed without WM (i.e., stage 0), then at the starting of stage 1, the state and motor state-action values are copied under the augmented policy as follows:

$$V^{(1)}(m,o) = V^{(0)}(o), \quad \forall m, o \tag{4.2}$$

$$Q_M^{(1)}(m,o,a) = Q_M^{(0)}(o,a), \quad \forall m, o, a \tag{4.3}$$

where $V^{(i)}$ and $Q_M^{(i)}$ refer respectively to state and motor state-action values during stage i, and m, o and a represent respectively a possible content of the one-cell WM, a possible current observation and a possible motor action.

The same procedure is used when the WM capacity switches from one to two, or generally from C to $C+1$, with now the learned gating state-action values also needing to be transferred along with the motor Q-values. However, the transfer of the gating Q-values is not as straightforward as with the motor Q-values, since when the learner switches to a WM capacity of $C+1$, the number of possible gating actions becomes $C+2$ (i.e., U_1–update the first slot, U_2–update the second slot,$\cdots$, U_{C+1}–update the $(C+1)^{th}$ slot, or M–maintain the contents of all slots), instead of $C+1$ gating actions with a WM capacity of C (i.e., U_1, U_2,$\cdots$, U_C, or M). To overcome this challenge, we base the transfer of the gating policy on the simple idea that, at the start of stage $C+1$ (which corresponds to an allocated WM capacity of $C+1$), the gating agent would keep controlling its first C cells in the same manner as it used to control them when it allocated exactly C cells to the task (note that here we refer only to the period that comes directly after augmenting the WM resources, as the learner might change the gating policy later during the new stage). Thus, as illustrated in Figure 4.1, if during stage C, the best gating action was g given the WM contents $(m_1, \cdots, m_C)$, then g remains the best action to choose at the starting of stage $C+1$ given the same contents of the first C WM cells, regardless of the content of the new cell $C+1$. For example in the 12-AX task, if by the end of stage 1, the agent has learned to keep track of the digit cues using the single allocated WM cell, the procedure of transfer ensures that the learner will keep storing the digit cues in that same WM cell after adding a second WM cell (it would then presumably remain for the learner to learn how to control the second WM cell, ideally reserving it to keep track of the letter cues). Formally, this can be achieved by copying the gating Q-values in the following manner:

$$Q_G^{(C+1)}(m_1, \cdots, m_C, m_{C+1}, o, U_i) = Q_G^{(C)}(m_1, \cdots, m_C, o, U_i), \quad \forall i = 1 \cdots C \tag{4.4}$$

$$Q_G^{(C+1)}(m_1, \cdots, m_C, m_{C+1}, o, U_{C+1}) = Q_G^{(C)}(m_1, \cdots, m_C, o, M) \tag{4.5}$$

$$Q_G^{(C+1)}(m_1, \cdots, m_C, m_{C+1}, o, M) = Q_G^{(C)}(m_1, \cdots, m_C, o, M) \tag{4.6}$$

where here we biased the "Update" action of the new WM cell to have the previous value of the "Maintain all" action to ensure that the first C WM cells are updated (or potentially updated to be more precise since there is also the exploration noise to take into account), as in the previous stage by preserving the same ranking of their Q-values.

Finally, we assume that the learner stops increasing its WM resources when the accuracy rate threshold of 0.99 is reached (i.e., when the proportion of correct responses in one block of length w becomes higher than 99%). Note that this modified version of the RLWM model, which we will call from now on the resource allocation model, uses exactly the same mechanisms to update the state and state-action values and to select actions during each stage, as the original RLWM model (see Section 3.2.2). The only difference is the possible transition to a larger WM

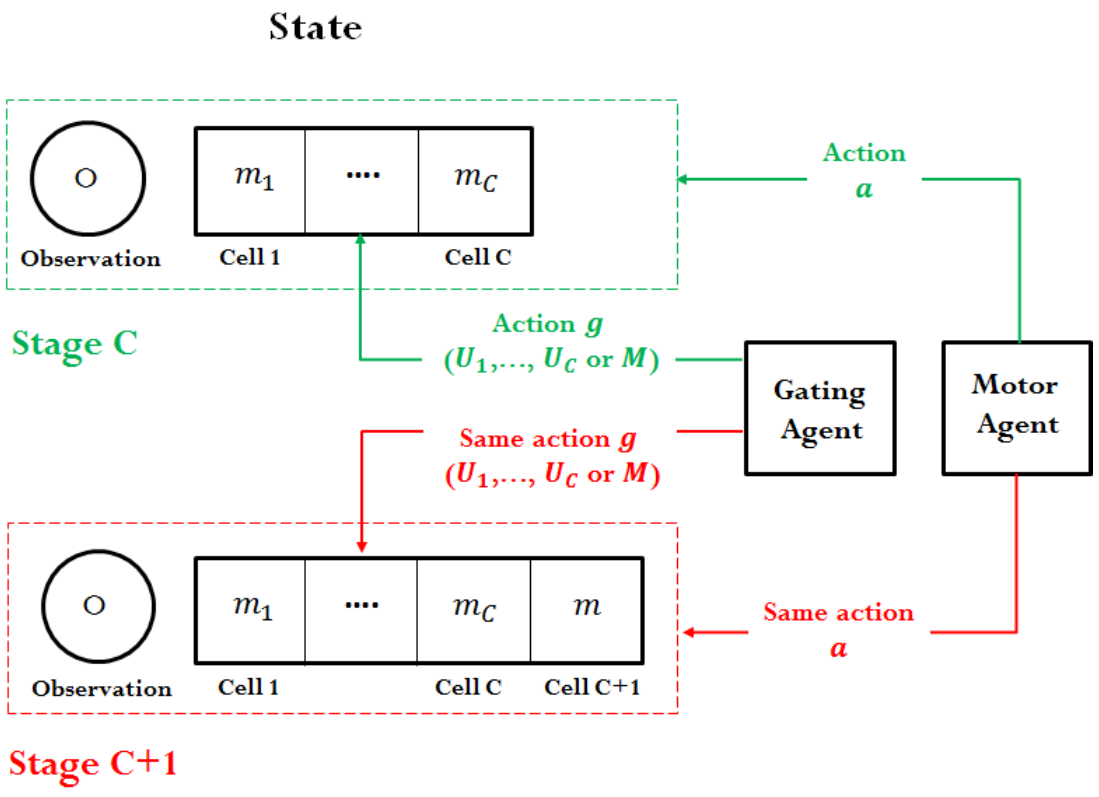


Figure 4.1: Transfer of learning after increasing WM resources under the resource allocation model. When transiting from stage C to stage $C+1$, the previous look up tables containing the motor and gating Q-values are copied to the new augmented look up tables such that the gating and motor actions that had the highest Q-values (marked in green) remain with the highest Q-values at the starting of the new stage (marked in red).

capacity and the transfer of learning required after each transition (i.e., the copying of the state and state-action values).

4.2.2 Model settings

We simulated the resource allocation model on the 12-AX task with both manual and automatic allocation, and compared their learning curves with those of the RLWM model with different fixed WM capacities (zero, one and two). To simplify simulations, we constrained the adaptive WM capacity to not go beyond two—a capacity that is enough to perfectly solve the 12-AX.

We encountered two main challenges when simulating the resource allocation models: (1) how to choose the learning rate to use in each learning stage, and (2) how to choose the exploration parameter. In particular, after augmenting WM resources, it is important that the agent explores enough to learn the contingencies that it has not been able to learn during the previous stages, but not too much so that the agent does not unlearn the useful contingencies that have been acquired, and also to not decrease short-term accuracy. The same thing is true for the learning rate; the learner needs to use a high learning rate at the beginning of a new stage in order to unlearn the no longer useful strategies (for example in the 12-AX, a good strategy to use in stage 0 is to always select "L" when the current observation is "X"; however, in stage 1, the learner would need to slightly reshape this strategy, for example, by continuing to select "L" when "X" is encountered, except when the stored item in WM is either 1 or A), but the learning rate should not be so high as to not converge to a suboptimal policy.

To overcome the challenge of setting the learning rate after each WM capacity transition, we introduced a learning rate that automatically tunes itself according to the other parameters, using Dabney and Barto's (2012) work. The idea behind their adaptive learning rate is simple: in each trial t, the learning rate α_t is adjusted such that the update of the state value V_t (see Equation 3.8) will necessary improve—or at least not hinder—learning. What this means is that if we were to compute the temporal-difference error again, this time using the just updated state-value V_{t+1}, the new temporal-difference error δ'_t would be closer to 0 than the actual temporal-difference error δ_t—that is, we would have $|\delta'_t| < |\delta_t|$ and $sign(\delta'_t) = sign(\delta_t)$. Dabney and Barto (2012) proposed one possible α_t that satisfies this criterion, which is defined by:

$$\begin{cases} \alpha_t = min\{\alpha_{t-1}, |\gamma e^V_{t-1}(s_t) - e^V_{t-1}(s_{t-1})|\} \\ \alpha_0 = 1 \end{cases} \tag{4.7}$$

where s_t refers to the state at time t and e^V_{t-1} to the state eligibility trace function at time $t-1$.

We found that using this adaptive learning rate substantially improved performance even for the RLWM model with fixed WM capacity, as we shall see in the next section. Therefore, we used this new form of the learning rate in all our simulations with the RLWM model. This also guarantees a fairer comparison against the resource allocation models (manual and automatic).

Concerning the choice of the exploration rate, we assumed that the learner uses a different exploration rate T_C during each learning stage C, and followed a different strategy for setting the exploration rate depending on the allocation type. In the manual case, as our main goal was to simply show the benefit of the approach of gradually increasing WM resources without worrying too much about parameter details, we used the best exploration rate that we found for each fixed capacity in the corresponding learning stage (for example, if we were to find that the best exploration rates for the RLWM model with zero, one and two WM cells are respectively $T_0 = 1$, $T_1 = 3$ and $T_2 = 4$, we would use those same exploration rates respectively in learning stage 0, 1 and 2 when simulating the manual resource allocation model). Also, to promote plasticity when transiting to a new learning stage, we reduced the state-action values by a factor of 2 before copying them into the new augmented look-up value table (note that we did this only in the manual case, as the same effect could be achieved by using a decreasing exploration rate, as presented below).

In order to balance exploration-exploitation in the automatic allocation case, we used a decreasing exploration rate with the number of visits to the current state.

More precisely, at time t, we set the exploration rate as follows:

$$T_t = \frac{1}{\log(a \cdot n_t(s_t) + b)} \tag{4.8}$$

where s_t is the state at time t, and $n_t(s)$ is the number of visits to state s at time t. The constants a and b are free parameters, which determine the starting exploration rate (with b) and the rate of decrease of exploration (with a). Note that this form of exploration is close to the one used by Singh et al. (2000), with the main difference being that they considered an exploration parameter dependent also on the state-action values. As with the fixed exploration rates in the manual allocation case, we also allowed the parameters a_C and b_C to change in each learning stage C.

For the most part, we set the parameters of our models as in the previous chapter. We used a discount factor $\gamma = 0.95$, and initially set all Q-values to 0. We chose all other parameters by doing an extensive grid search to find the best parameters (i.e., the parameters that maximise the asymptotic accuracy across 30 simulations). Also, as learning was generally faster than in the previous chapter, we run all models for $N = 10^6$ trials, instead of 10^7 trials. Finally, we set the parameters that control the transition to a larger allocated WM capacity as follows:

- for manual allocation, the trials where the first and second WM capacity transitions happened were respectively $N_1 = 10^4$ and $N_2 = 3 \times 10^5$. These values were chosen visually based on the learning curves of the zero and one cell RLWM models, so that the allocation model would have enough time to converge in each stage.

- for automatic allocation, we used a different parameter ϵ and w for each stage to detect if accuracy has levelled out: for stage 0, $\epsilon_0 = 0.005$ and $w_0 = 1000$ (i.e., allocate one WM cell if the number of errors made in the current block of 1000 trials in comparison with the last block does not change by more than

5 errors), while for stage 1, $\epsilon_1 = 0.0025$ and $w_1 = 10000$ (i.e., allocate a second WM cell if the number of errors made in the current block of 10000 trials in comparison with the last block does not change by more than 25 errors). These parameters were chosen somewhat arbitrarily after trying a few possible values, but the main idea here was that the learner would need to spend more time learning with higher WM capacities than with lower capacities as the dimensionality of the learning problem would be greater in the first case, so the criterion for switching the WM capacity needs to be stricter when learning with a larger WM capacity. Although there are certainty better ways to capture this idea, we did it here by using different switching parameters ϵ and w for each learning stage.

Table C.1 lists the chosen parameters for each model.

4.3 Results

4.3.1 Manual allocation

We first applied the manual resource allocation model to the 12-AX (see Figure 4.2A-B) in order to provide a basic evaluation of the benefits—if there are any—of the gradual allocation of WM resources over the fixed allocation approach. One finding that is worth highlighting before we present the results of the resource allocation model simulations is that the adaptive learning rate improved substantially the learning speed of all three RLWM models. In fact, for example, the one-cell and two-cell RLWM models with the adaptive learning rate reached the accuracy level of 0.95 in 1000 successive trials, in respectively about 2.7×10^5 and 4.4×10^5, as opposed to 5.7×10^5 and 1.3×10^6 when simulated with fixed learning rates (based on the simulations presented in the previous chapter).

More importantly, allowing the RLWM model to change its allocated WM capacity, although in a very basic manner, made the model learn faster than with a fixed WM capacity of two cells, which is the minimum capacity required to perfectly solve the 12-AX. Not only that, the new model also reached a slightly better asymptotic accuracy than the two-cell RLWM model. In addition, the manual resource allocation model performed, during each learning stage, almost the same as the RLWM model with the same allocated WM capacity, with the allocation model performing lower only at the beginning of each new learning stage. Thus, by augmenting step-by-step its WM capacity, the agent made the most of all the three capacities: learning was accelerated with zero and one WM cell, then higher long-term performance was achieved after switching to two WM cells.

However, one apparent issue that prevented the manual allocation model from learning even faster is the drop in accuracy after each switch to a higher WM capacity, which the model did not recover from rapidly (see for example, in Figure 4.2B, the drop in accuracy after switching from no allocated WM resources to one allocated WM cell on trial $N_1 = 10^4$). The obvious reason behind these decreases in accuracy is that we increased the exploration rate (from $T_0 = 2$ to $T_1 = 3$, then $T_2 = 4$) and reduced the state-action values by a factor 2, each time the model transitioned to a new stage to promote plasticity. One possible way to overcome this problem would be to use a decreasing exploration rate, which would also remove the need to discount the learned Q-values (for example, by making sure that the exploration rate is large enough at the starting of a new stage). Although we did not re-simulate the manual resource allocation model using a decreasing exploration rate, we used it directly with the automatic resource allocation model, as we shall see next.

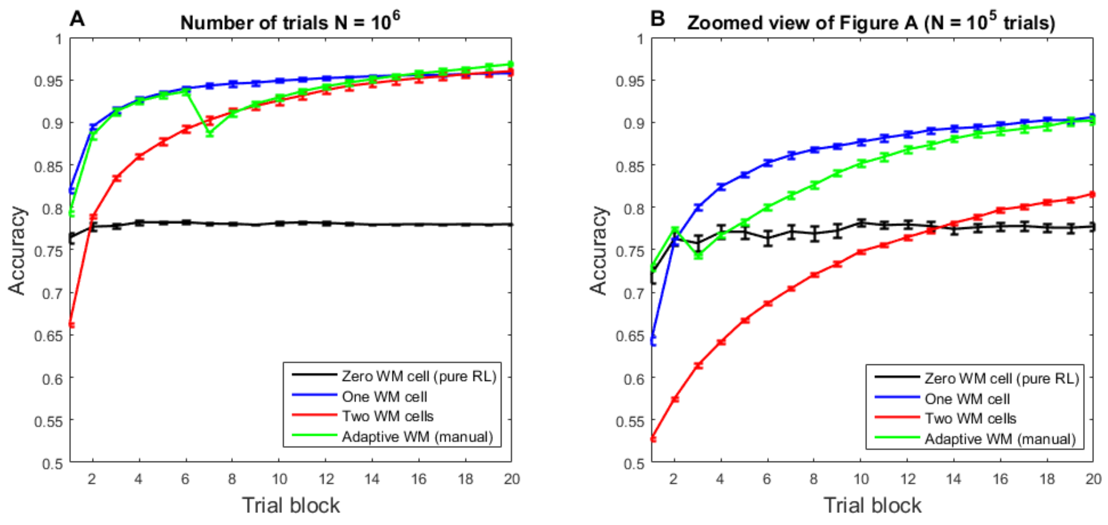


Figure 4.2: Learning curve on the 12-AX task using adaptive working memory capacity with manual allocation criterion (green). Learning curves obtained using fixed working memory capacity of zero (black), one (blue) and two (red) are also shown. Each model was run for 10^6 trials, and accuracy was averaged over 30 runs. (A) Here, we show the full learning curves, which were obtained by computing the average accuracy over blocks of 5×10^4 trials. (B) Zoomed view of the learning curves shown in Figure A. Here accuracy was averaged over blocks of 5000 trials.

4.3.2 Automatic allocation

We also simulated the automatic resource allocation model on the 12-AX, and compared its performance against the RLWM model with fixed WM capacities of zero, one and two. In addition to automatically deciding when to increase WM resources, we introduced two other modifications to the resource allocation model in comparison with the manual case. First, we used a decreasing exploration rate (see Equation 4.8) to prevent as much as possible the decline in learning accuracy after switching to a larger WM capacity. The second modification consisted of no longer discounting the state-action values (i.e., dividing the Q-values by a factor of two) before copying them at the transition step, and only relying on the choice of the exploration rate parameters to create the necessary plasticity after transiting to a new learning stage.

Figure 4.3A compares the learning curves of the RLWM model with adaptive WM capacity and with different fixed WM capacities. The results show that the automatic allocation model learned the 12-AX almost perfectly by the end of the task (reached an accuracy of approximately 0.99 in the last block of 50 thousand trials as opposed to about 0.96 with the two-cell RLWM model). Moreover, the automatic version of the resource allocation model recovered faster from the decrease in accuracy after augmenting the allocated WM capacity than the manual version (notice in Figure 4.3B, the relatively quick recovery of the automatic allocation model after the first transition, which happened during the second block). Another difference between the automatic and manual case is that the automatic resource allocation model augmented its WM resources generally earlier (in average, the first and second change in WM capacity happened respectively in trials $N_1 = 6500$ and $N_2 = 73000$, as opposed to the manually chosen trials $N_1 = 10^4$ and $N_2 = 3 \times 10^5$ for the manual allocation model).

4.4 Discussion

This chapter addresses the idea, proposed in Chapter 3, of allowing the learner under the RLWM model to gradually increase its allocated WM resources, while performing a WM learning task. Two criteria were used for deciding when to increase the learner's allocated WM capacity: a basic approach based solely on the time spent learning with each WM capacity, which we entitled the manual resource allocation approach, and a more adaptive approach that depends on the learner's current accuracy rate, which we referred to as the automatic resource allocation approach.

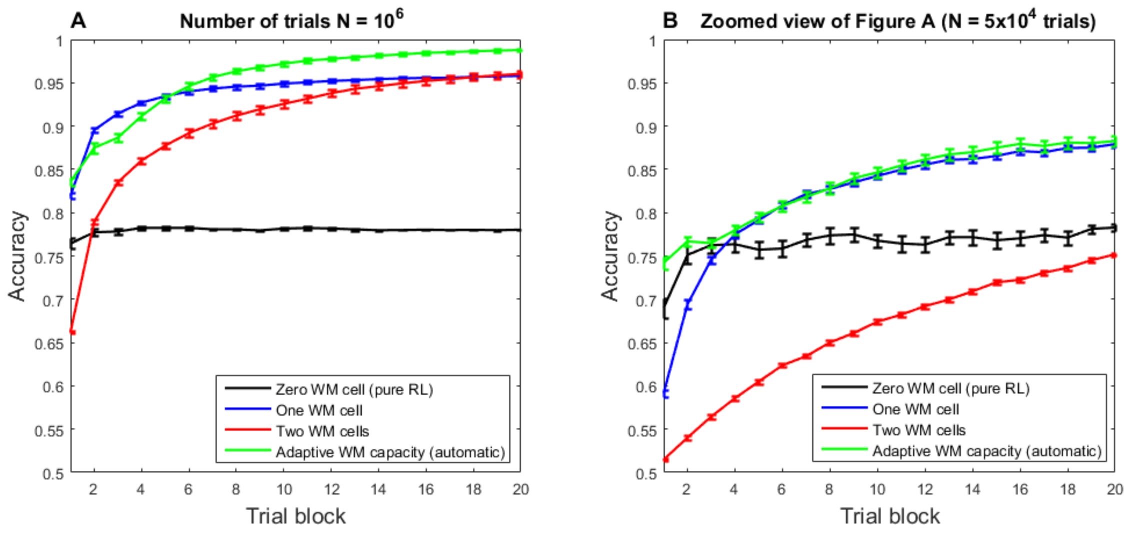


Figure 4.3: Learning curve on the 12-AX task using adaptive working memory capacity with automatic allocation criterion (green) in comparison with learning curves obtained using fixed working memory capacities of zero (black), one (blue) and two (red). Each model was run for 10^6 trials, and accuracy was averaged over 30 runs. (A) Here, we show the full learning curves, which were obtained by computing the average accuracy over blocks of 5×10^4 trials. (B) Zoomed view of the learning curves shown in Figure A. Here accuracy was averaged over blocks of 2500 trials.

The results of our simulations on the 12-AX task with both these approaches show the superiority of the flexible resource allocation model over the RLWM model, with a bigger advantage for the automatic allocation approach.

These findings, although preliminary and need to be replicated on other tasks, confirm the usefulness of starting small when performing complex WM learning tasks, as highlighted in previous computational modelling works (see Hertwig and Todd, 2003, for a brief review). One of the pioneer works promoting the 'start small' idea is that of Elman (1993), which investigated the ability of a recurrent neural network model to learn grammatical rules, such as noun-verb agreement in embedded and non-embedded clauses. His model succeeded only when he either gradually increased the complexity of the sentences that the network was exposed to (as in shaping), or gradually increased the memory power of the network (as in our resource allocation model). Elman (1993) connected his second finding to language acquisition in young children; they initially have a very low WM capacity, which allows them to focus on the basic rules of language. Once these have been learned, more complex language acquisition can take place, especially when their WM capacity expands more. Here, we propose that people may voluntarily gradually allocate their WM resources to help them learn the WM learning tasks they encounter, rather than being forced into it due to their natural cognitive development (although the manual resource allocation approach can also be potentially used to model the same type of situations that Elman (1993) studied, which involve the interaction between learning and WM development).

A growing body of evidence from the visual WM field suggests that people can indeed voluntarily control their WM resource allocation, and update it as the salience or goal relevance of the stimuli to be stored changes (Ma et al., 2014; Bays et al., 2011; Gorgoraptis et al., 2011). For example, Gorgoraptis et al. (2011) showed that the precision of recall was higher for cued items (i.e., items that were indicated to participants as being more likely to be probed for recall) than uncued items or when no cues were given. Human ability to strategically allocate WM resources has not only been observed in visual WM tasks, but also in tasks similar to the 12-AX. For instance, Chatham and Badre (2013) suggested that people might track the utility of information in WM for maximising future reward, and decide to reallocate their WM resources if the utility of stored information declines later during the task. They also proposed a Q-learning model for how people might decide to allocate their WM resources. However, the notion of reallocation of resources that Chatham and Badre (2013) addressed is more general than how we defined it here. Their notion alludes not only to the function of deciding how many items to store in WM (although implicitly), but also to other WM functions such as when to store and when to remove an item from WM. Their main argument is that people would only retain or keep retaining an item in WM if it has a high utility in guiding future behaviour.

This leads us to the question of how we might extend the model to allow for an even more flexible resource allocation than what has been proposed in this chapter. One possible idea is to consider the allocation of WM resources as an active gating action per se, and assign a value, $V_A(c)$, for each possible allocated WM capacity, c. We could then assume that the learner chooses the number of WM cells to use based on the allocation values ${V_A}'$s, and according to the softmax selection rule. One advantage is that we can obtain the allocation values directly from the maintained state values, so we do not need to independently keep track of them. Here is how one might proceed with this computation. Let's assume that the learner keeps track of a separate state value function $V^{(c)}$, for each possible number of allocated WM cells, c (to use when the state is determined by the contents of c WM cells along with the current observation). We could then define $V_A(c)$, the allocation value of c, as the average state value $V^{(c)}$ over all visited states in the last n trials. For example, the allocation value of $c = 2$ would be given by:

$$V_A(2) = 1/n \sum_{(m_1,m_2,o) \text{ in the last } n \text{ trials}} V^{(2)}(m_1, m_2, o) \tag{4.9}$$

where m_1 is the item stored in the first cell, m_2 is the item stored in the second cell, and o is the current observation. The sum is over all visited states in the previous n trials. Once the learner has computed the allocation values for all possible WM capacities, it would select the capacity to allocate using softmax rule.

Obviously this is just a starting idea that needs further development. In particular, it is important to allow for knowledge to be transferred across the state values from different dimensions. In other words, if in the last n trials, the agent was learning with c WM cells, we need to decide how we should transfer the learned values from $V^{(c)}$ to the other value functions $V^{(0)}, V^{(1)}, \cdots, V^{(c-1)}, V^{(c+1)}, \cdots$.

Another interesting question is how to test whether or not people use the gradual resource allocation approach when performing WM learning tasks. One possibility is to fit participants' data to the manual or automatic resource allocation model. This, however, will be very tedious as we will not have access to participants' WM contents, and also because both models have many free parameters to fit. A better approach would be to simply try to find if there is an overall pattern of gradual allocation in participants' data. This could be achieved by measuring their WM span while performing the main WM learning task, and see if the span is generally decreasing throughout the task, as predicted by the resource allocation model (in fact, the WM span task would measure their remaining WM resources after they have allocated a certain capacity to solve the task, so if the allocated WM resources are increasing, the remaining resources will be decreasing), or fixed as assumed in most of the WM-based RL models. This is the approach that we take in Chapter 6 to test how people solve WM learning tasks and how they utilise their WM while performing them.

However, before that, the next chapter explores the Bayesian approach to try to solve some of the issues that we encountered in this chapter, such as how to automatically set the exploration when switching to a different WM capacity. The adaptive learning rate that we used worked well in our simulations on the 12-AX, but the decreasing exploration rate only partially solved the problem. In particular, it added two extra parameters to the model to set in each learning stage, so that, for example, with two WM cells allocated in the final stage, there will be six exploration parameters to set. The Bayesian approach has the advantage of automatically tuning the exploration rate. It also has the advantage of providing estimates of the uncertainty around the Q-values estimates (i.e., precision), which could be used to introduce an enhanced criterion for deciding when to increase WM resources based on uncertainty rather than the current accuracy rate. The idea is that the learner would change its allocated WM capacity when a high level of precision in the Q-values estimates has been achieved with the current WM resources (which is an indication of learning convergence), but the performance is still not satisfactory to the learner.

In conclusion, the gradual WM resource allocation is a plausible and simple, yet powerful approach for improving performance in WM-based RL models. Starting with low WM resources allows the learner to build the foundation that she can rely on to do more complex learning. As such, this process is similar to changing gears when driving; the driver—like the agent under the resource allocation model—uses the first and second gear to build up speed, before switching to higher gears for a smoother driving.

Chapter 5

Bayesian working memory-based reinforcement learning [1]

5.1 Introduction

The working memory-based reinforcement learning (RL) models that we have explored so far to solve working memory learning tasks are slow to learn and require setting the learning parameters carefully in order to reasonably expect convergence to an optimal policy. In particular, it is crucial to allow the models to explore enough but not excessively (i.e., balancing the trade-off between exploration and exploitation). Solving the exploration-exploitation trade-off is especially critical in these models, as the addition of a working memory (WM) to the agent results in a sizeable state-action policy space.

Thompson sampling has received increasing interest in recent years as a method for balancing exploration and exploitation in bandit problems (May et al., 2012).

[1] A version of this chapter was published as A. Ez-zizi, S. Farrell, and D. Leslie (2015). Bayesian Reinforcement Learning in Markovian and non-Markovian Tasks. In *IEEE Symposium Series on Computational Intelligence*, pages 579-586.

It is self-tuning and automatically anneals exploration–exploitation in response to the level of information in the value estimates. The general idea behind Thompson sampling is to choose an action according to its probability of having the highest value. As such, it requires the estimation of posterior distributions of action values instead of simply action values. Thus a Bayesian approach to RL is required.

Despite its elegance, the Bayesian approach has only been used occasionally in modern RL. Among the few model-free Bayesian RL models that have been proposed in the literature are the SARSA variant of the Kalman temporal difference model (KTD-SARSA) and its extended version (XKTD-SARSA), both of which use online value function approximation along with Kalman filtering to estimate the state-action value function of a given policy (Geist and Pietquin, 2010). A related algorithm is the SARSA-based Gaussian process temporal difference (Engel et al., 2005), which can be seen as a special case of the KTD-SARSA model in stationary environments. These models consider state-action values to be random processes and compute their posterior distribution given the observed rewards and transitions.

In this chapter, we describe a modified version of the XKTD-SARSA model that does not use value function approximation, and show that it has a similar form of updates to the standard SARSA(λ), but with time- and state-dependent step-sizes that are governed by the learned properties of the system instead of arbitrary learning rates and eligibility traces. We then show how to implement XKTD-SARSA for a non-Markovian model incoropating a WM module, following Todd et al.'s (2009) approach. The advantages of introducing the Bayesian approach are seen in the existence of a native mechanism for sharing information across state-action pairs, replacing eligibility traces, and a native mechanism for balancing exploration and exploitation, removing the need to use an appropriately parameterised action-selection method. Finally, we compare the performance of the Bayesian WM-based

RL with that of the RLWM model on the 12-XY and 12-AB-XY—two simplified versions of the 12-AX task (the RLWM model and these two tasks were introduced in Chapter 3).

This chapter mainly demonstrates the potential of using Bayesian techniques to solve WM learning tasks. Thus, for example, we leave the question of how to implement the gradual WM resource allocation idea (see Chapter 4) under the Bayesian approach for future investigations.

5.2 Background and notation

The purpose of this section is to present the Markovian version of the Bayesian RL model (RL) that we use to construct our Bayesian WM-based RL model. We start by offering a quick reminder of the SARSA method, since the Bayesian model uses the same temporal-difference equation as SARSA (a full description of SARSA can be found in Section 1.1.2). This will also give us the opportunity to introduce the necessary notations that are used throughout the chapter. We then present an initial version of the Bayesian model, called KTD-SARSA, which is compatible only with Markovian tasks with deterministic transitions. Finally, we present a more general version of the Bayesian model, entitled XKTD-SARSA, which can work with both deterministic and stochastic transitions.

5.2.1 SARSA (vectorial form)

Let $z_t = (s_t, a_t)$ and r_t be respectively the state-action pair and reward at time t. We consider that there are p possible state-action pairs denoted by z^1,...,z^p. Under SARSA, the state-action values are updated, at each time step t, according to the following equation:

$$Q_{t+1}(z_t) = Q_t(z_t) + \alpha\delta_t \tag{5.1}$$

where α is the learning rate, and δ_t is the temporal difference error signal, which is given by:

$$\delta_t = r_t - (Q_t(z_t) - \gamma Q_t(z_{t+1})) \tag{5.2}$$

with γ being the discount factor. Denote the vector of all state-action pairs by $\boldsymbol{Q_t}$:

$$\boldsymbol{Q_t} = \begin{pmatrix} Q_t(z^1) \\ \vdots \\ Q_t(z^p) \end{pmatrix}$$

Thus, Equation 5.2 can be written into a vectorial form as:

$$r_t = \boldsymbol{F_t}\boldsymbol{Q_t} + \delta_t \tag{5.3}$$

where, $\boldsymbol{F_t} = (0 \ \cdots \ 0 \ 1 \ \cdots \ -\gamma \ 0 \ \cdots \ 0)$, which has all its elements equal to 0 except the j_1^{th} and j_2^{th} element such that $z^{j_1} = z_t$ (hence the value 1) and $z^{j_2} = z_{t+1}$ (hence the value $-\gamma$). This vectorial formulation will be useful when we present the Bayesian RL models.

Under SARSA(λ)—the generalisation of SARSA—all state-action values can be potentially updated using a modified version of Equation 5.1:

$$Q_{t+1}(z) = Q_t(z) + \alpha\delta_t e_t(z) \tag{5.4}$$

where $e_t(z)$ is the eligibility trace of the state-action pair z, which decays over time by a decay parameter $\lambda \in [0, 1]$.

5.2.2 Kalman temporal difference framework for Markovian tasks

The Bayesian RL models that we use in this chapter are based on the Kalman temporal difference (KTD) framework (Geist and Pietquin, 2010). The general idea behind this framework is to assume that the Q-values are random variables, and then use the Kalman filter framework (Welch and Bishop, 2000) to compute posterior distributions over the Q-values after each newly observed reward. More specifically, we assume that the relation between $\boldsymbol{Q_t}$ and r_t can be represented by a Kalman filter type system, where $\boldsymbol{Q_t}$ is the state vector (i.e., vector to estimate) and r_t is the observed variable (i.e., measurable variable that the estimation is based upon):

$$\begin{cases} \boldsymbol{Q_t} &= \boldsymbol{Q_{t-1}} + \boldsymbol{V_t} \\ r_t &= \boldsymbol{F_t Q_t} + \delta_t \end{cases} \tag{5.5}$$

such that we assume that the state vector $\boldsymbol{Q_t}$, and the noises $\boldsymbol{V_t}$ and δ_t satisfy the conditions of a standard Kalman filter system (see for example, Welch and Bishop, 2000). We denote the covariance matrix of $\boldsymbol{V_t}$ and variance of δ_t respectively by $\boldsymbol{P_{V_t}}$ and P_{δ_t}—that is, we have $\boldsymbol{V_t} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{P_{V_t}})$ and $\delta_t \sim \mathcal{N}(0, P_{\delta_t})$.

Next, we will present the original version of the KTD-SARSA model, which assumes that the transitions of the underlying Markov decision process (MDP) are deterministic. Then, we will present the extended version of the KTD-SARSA (XKTD-SARSA), which can handle MDPs with stochastic transitions.

KTD-SARSA: A Bayesian reinforcement learning model for deterministic Markov decision processes

In a deterministic MDP, the noise δ_t can be simply assumed to be white (i.e., δ_t's are uncorrelated, have mean zero, and a finite variance), in which case the Kalman estimates of the Q-values vector and its uncertainty can be derived in a straightforward manner and are given by:

$$\begin{cases} \boldsymbol{\hat{Q}_t} &= \boldsymbol{\hat{Q}_{t-1}} + \boldsymbol{K_t}(r_t - \boldsymbol{F_t \hat{Q}_{t-1}}) \\ \boldsymbol{\hat{P}_t} &= (\boldsymbol{\hat{P}_{t-1}} + \boldsymbol{P_{V_t}}) - \boldsymbol{K_t}[\boldsymbol{F_t}(\boldsymbol{\hat{P}_{t-1}} + \boldsymbol{P_{V_t}})\boldsymbol{F_t^T} + P_{\delta_t}]\boldsymbol{K_t^T} \end{cases} \tag{5.6}$$

where:

$$\begin{aligned}
\hat{\boldsymbol{Q}}_{\boldsymbol{t}} &= \mathbb{E}(\boldsymbol{Q}_{\boldsymbol{t}}|r_0, \cdots, r_t) \\
\boldsymbol{K}_{\boldsymbol{t}} &= \frac{(\hat{\boldsymbol{P}}_{\boldsymbol{t-1}} + \boldsymbol{P}_{\boldsymbol{V_t}})\boldsymbol{F}_{\boldsymbol{t}}^T}{\boldsymbol{F}_{\boldsymbol{t}}(\hat{\boldsymbol{P}}_{\boldsymbol{t-1}} + \boldsymbol{P}_{\boldsymbol{V_t}})\boldsymbol{F}_{\boldsymbol{t}}^T + P_{\delta_t}} \\
\hat{\boldsymbol{P}}_{\boldsymbol{t}} &= \mathbb{E}[(\boldsymbol{Q}_{\boldsymbol{t}} - \hat{\boldsymbol{Q}}_{\boldsymbol{t}})(\boldsymbol{Q}_{\boldsymbol{t}} - \hat{\boldsymbol{Q}}_{\boldsymbol{t}})^T]
\end{aligned}$$

$\boldsymbol{K}_{\boldsymbol{t}}$ is the Kalman gain and $\hat{\boldsymbol{P}}_{\boldsymbol{t}}$ corresponds to the variance associated with the estimation $\hat{\boldsymbol{Q}}_{\boldsymbol{t}}$ (posterior mean) of $\boldsymbol{Q}_{\boldsymbol{t}}$.

Let us now write these equations into a scalar form and compare them with those of the SARSA(λ) algorithm. First, denote $k_t(z, z') = Cov(Q_t(z), Q_t(z'))$ for any state-action pairs z and z'. Also, to simplify expressions, let us assume that $\boldsymbol{P}_{\boldsymbol{V_t}} = 0$ (this is a valid assumption if we are dealing with stationary environments as the ones captured by the tasks that we are using in this chapter). By developing the equations in 5.6, we can write the posterior mean of each state-action value and its covariance with any other state-action pair as follows:

$$\hat{Q}_{t+1}(z) = \hat{Q}_t(z) + \alpha_t(z)\hat{\delta}_t^{TD} \tag{5.7}$$

$$\hat{k}_{t+1}(z, z') = \hat{k}_t(z, z') - \alpha_t(z)\left[\hat{k}_t(z', z_t) - \gamma\hat{k}_t(z', z_{t+1})\right] \tag{5.8}$$

where:

$$\begin{aligned}
&\hat{Q}_t(z) = \mathbb{E}(Q_t(z)|r_0, \cdots, r_t) \\
&\hat{k}_t(z, z') = \mathbb{E}[(Q_t(z) - \hat{Q}_t(z))(Q_t(z') - \hat{Q}_t(z'))] = \mathbb{E}[Cov(Q_t(z), Q_t(z')|r_0, \cdots, r_t)] \\
&\hat{\delta}_t^{TD} = r_t + \gamma\hat{Q}_t(z_{t+1}) - \hat{Q}_t(z_t) \\
&\alpha_t(z) = \frac{\hat{k}_t(z, z_t) - \gamma\hat{k}_t(z, z_{t+1})}{\hat{k}_t(z_t, z_t) - 2\gamma\hat{k}_t(z_t, z_{t+1}) + \gamma^2\hat{k}_t(z_{t+1}, z_{t+1}) + P_{\delta_t}}
\end{aligned}$$

The updating equation for the posterior mean of a state-action value looks similar to that obtained using a standard SARSA(λ) (see Equation 5.4), but with time- and state-dependent learning rate $\alpha_t(z)$ (here the learning rate and eligibility trace terms have been replaced with a variable learning rate). Here also, all state-action values could potentially be updated depending on their covariance (-estimate) with the current expected reward. In fact, $\alpha_t(z)$ can be re-written in terms of variances and covariances as:

$$\alpha_t(z) = \frac{\mathbb{E}[Cov(Q_t(z), \overline{r}_t | r_0, \cdots, r_t)]}{\mathbb{E}[Var(\overline{r}_t | r_0, \cdots, r_t)] + P_{\delta_t}} = \frac{\widehat{Cov_t}(Q_t(z), \overline{r}_t)}{\widehat{Var_t}(\overline{r}_t) + P_{\delta_t}}$$

where $\overline{r}_t = Q_t(z_t) - \gamma Q_t(z_{t+1})$ is the expected reward at time t, and where we denoted $\mathbb{E}[Cov(X, Y | r_0, \cdots, r_t)]$ as $\widehat{Cov_t}(X, Y)$ (estimator of the covariance between X and Y at time t), and $\mathbb{E}[Var(X | r_0, \cdots, r_t)]$ as $\widehat{Var_t}(X)$.

The formula for $\alpha_t(z)$ means that the more positively correlated $Q_t(z)$ and $\overline{r}_t$ are, the bigger is the step-size $\alpha_t(z)$ used to update $Q_t(z)$. This is more true when both the estimated variance of $\overline{r}_t$ and true variance of δ_t are small (i.e., when the estimation of the expected reward is more precise and the system is less noisy). This is comparable to how eligibility trace works—the more recently visited a state-action pair, the more updated its corresponding Q-value. Similarly, here, the more correlated a state-action value to the current expected reward, the more it is updated.

Likewise, Equation 5.8 can be re-written into a more interpretable way as:

$$\begin{aligned}\hat{k}_{t+1}(z, z') &= \widehat{Cov_{t+1}}(Q_{t+1}(z), Q_{t+1}(z')) \\ &= \widehat{Cov_t}(Q_t(z), Q_t(z')) - \frac{\widehat{Cov_t}(Q_t(z), \overline{r}_t)\widehat{Cov_t}(Q_t(z'), \overline{r}_t)}{\widehat{Var_t}(\overline{r}_t) + P_{\delta_t}}\end{aligned}$$

which, in the case of variances, becomes:

$$\widehat{Var_{t+1}}(Q_{t+1}(z)) = \widehat{Var_t}(Q_t(z)) - \frac{\widehat{Cov_t}(Q_t(z), \overline{r}_t)^2}{\widehat{Var_t}(\overline{r}_t) + P_{\delta_t}}$$

This means that the uncertainty in a state-action value estimate decreases as its covariance with the expected reward increases, and that this is more true when the estimate of the expected reward is more precise or when the system noise is small. The interesting thing is that this decrease in uncertainty does not depend directly on the observed reward r_t as in the updating equation (5.7) of the Q-value means.

Action-selection via Thompson sampling

To finish the specification of the KTD-SARSA model, we must determine how to select actions based on the estimated Q-values and their underlying uncertainty. Given that we have the whole distribution of the state-action values, one sensible approach, known as Thompson sampling (Thompson, 1933), is to sample action a with the probability it has the highest Q-value in the current state. That is, at time t, if the state is $s_t = s$, we compute for each action a the probability $\mathbb{P}(argmax_{a'} \, Q(s, a') = a \,|\, \text{observations to date})$. The resulting probability distribution can then be used to sample an action. This can be efficiently implemented by simply sampling Q-values from their posteriors for each of the possible actions in the current state, and then choose the action with the highest sampled Q-value.

XKTD-SARSA: A generalisation of KTD-SARSA

The previous model is only suitable for tasks where the transitions of the underlying MDP are deterministic. In the stochastic case, the δ_t's can no longer be assumed independent. In fact, Geist and Pietquin (2010) showed that ignoring this would result in biased Kalman estimates. To deal with this, they considered the δ-noise to be a moving average of two white noises (see also Engel et al., 2005, for a motivation for this choice):

$$\delta_t = -\gamma u_t + u_{t-1} \quad \text{with} \quad u_t \sim \mathcal{N}(0, \sigma^2)$$

Under this new assumption, the Kalman estimates cannot be obtained directly as was done in the KTD case. A solution for extending the Kalman filter to moving average noises is given in Geist and Pietquin (2011). The basic idea of their solution is to transform the moving average noise into a vectorial autoregressive noise by adding an auxiliary random process w_t that stores the occurrences of u_t at each time step. More specifically, with $w_t = u_t$, we can write:

$$\begin{pmatrix} w_t \\ \delta_t \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} w_{t-1} \\ \delta_{t-1} \end{pmatrix} + \begin{pmatrix} 1 \\ -\gamma \end{pmatrix} u_t$$

The Kalman filter problem can then be easily solved using the resulting autoregressive noise, by re-writing Equation 5.5 as:

$$\begin{cases} \boldsymbol{X_t} &= L\boldsymbol{X_{t-1}} + \boldsymbol{V_t'} \\ r_t &= \boldsymbol{F_t'}\boldsymbol{X_t} \end{cases} \tag{5.9}$$

where:

$$\boldsymbol{X_t} = \begin{pmatrix} \boldsymbol{Q_t}^T & w_t & \delta_t \end{pmatrix}^T, \boldsymbol{V_t'} = \begin{pmatrix} \boldsymbol{V_t}^T & u_t & -\gamma u_t \end{pmatrix}^T, \boldsymbol{F_t'} = \begin{pmatrix} \boldsymbol{F_t} & 0 & 1 \end{pmatrix}$$

$$\boldsymbol{L} = \begin{pmatrix} \boldsymbol{I_p} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0^T} & 0 & 0 \\ \boldsymbol{0^T} & 1 & 0 \end{pmatrix}, \boldsymbol{P'_{V_t}} = \begin{pmatrix} \boldsymbol{P_{V_t}} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0^T} & \sigma^2 & -\gamma\sigma^2 \\ \boldsymbol{0^T} & -\gamma\sigma^2 & \gamma^2\sigma^2 \end{pmatrix}$$

The posterior mean and covariance of the augmented vector $\boldsymbol{X_t}$ are given by:

$$\begin{cases} \boldsymbol{\hat{X}_t} &= \boldsymbol{L\hat{X}_{t-1}} + \boldsymbol{K_t'}(r_t - \boldsymbol{F_t'\hat{X}_{t-1}}) \\ \boldsymbol{\hat{P}'_t} &= (\boldsymbol{L\hat{P}'_{t-1}L^T} + \boldsymbol{P'_{V_t}}) - \boldsymbol{K_t'}[\boldsymbol{F_t'}(\boldsymbol{L\hat{P}'_{t-1}L^T} + \boldsymbol{P'_{V_t}})\boldsymbol{F_t'^T}]\boldsymbol{K_t'^T} \end{cases} \tag{5.10}$$

where $\boldsymbol{\hat{X}_t} = \begin{pmatrix} \boldsymbol{\hat{Q}_t}^T & \hat{w}_t & \hat{\delta}_t \end{pmatrix}^T$ and $\boldsymbol{K_t'}$ is the Kalman gain of the new system. The estimates of the mean and covariance matrix of the vector of state-action values $\boldsymbol{Q_t}$ can be obtained from $\boldsymbol{\hat{X}_t}$ and $\boldsymbol{\hat{P}'_t}$ by respectively extracting the first p elements of $\boldsymbol{\hat{X}_t}$ and the first $p \times p$ matrix block of $\boldsymbol{\hat{P}'_t}$. Notice also that δ_t and w_t are now part of the state to estimate.

We can extract the scalar updating equations for the posterior mean and covariances of the Q-values from the system of equations 5.10, as we did with KTD-

SARSA:

$$\hat{Q}_{t+1}(z) = \hat{Q}_t(z) + \alpha_t(z)(\hat{\delta}_t^{TD} - \hat{\delta}_t) \tag{5.11}$$

$$\hat{k}_{t+1}(z, z') = \hat{k}_t(z, z') - \frac{\widehat{Cov_t}(Q_t(z), \overline{r}_t + w_t)\widehat{Cov_t}(Q_t(z'), \overline{r}_t + w_t)}{\widehat{Var_t}(\overline{r}_t + w_t) + \gamma^2\sigma^2} \tag{5.12}$$

where:

$$\begin{aligned}
\hat{\delta}_{t+1} &= \hat{\delta}_t + \alpha_t(\delta)(\hat{\delta}_t^{TD} - \hat{\delta}_t) \\
\alpha_t(z) &= \frac{\widehat{Cov_t}(Q_t(z), \overline{r}_t + w_t)}{\widehat{Var_t}(\overline{r}_t + w_t) + \gamma^2\sigma^2} \\
\alpha_t(\delta) &= \frac{\widehat{Cov_t}(w_t, \overline{r}_t + w_t) + \gamma^2\sigma^2}{\widehat{Var_t}(\overline{r}_t + w_t) + \gamma^2\sigma^2}
\end{aligned}$$

The updating equations 5.11 and 5.12 for computing the posterior mean and covariance of the Q-values are not as different from those obtained in the deterministic case (see Equations 5.7 and 5.8) as it might look like. In fact, we expect that the posterior mean $\hat{\delta}_t$ would converge toward 0 (the true mean of δ_t). Also, given that w_t is equal to the white noise u_t, the posterior variance $\widehat{Var_t}(w_t)$ and posterior covariance $\widehat{Cov_t}(w_t, Q_t(z))$ are expected to converge respectively towards σ^2 (the true variance of u_t) and 0 (the true covariance between $Q_t(z)$ and u_t, because u_t is *a priori* independent from all state-action values). Hence, Equations 5.11 and 5.12 should become very similar to Equations 5.7 and 5.8 in the long run, and the differences between XKTD-SARSA and KTD-SARSA are in the early phases of learning. Given that the early stages of learning are often the very stages we are interested in here since they determine learning speed, we will use XKTD-SARSA, the extended version of the Bayesian model.

In the following, we will present the Bayesian RLWM model, which we constructed based on XKTD-SARSA, before assessing its performance on the 12-XY and 12-AB-XY—two WM learning tasks that we encountered in Chapter 3.

5.3 General methods

5.3.1 Bayesian RLWM model

To allow XKTD-SARSA to be applied effectively to WM learning tasks, we follow Todd et al. (2009) and supplement the agent with WM (as outlined in Chapter 3). As with the RLWM model, we assume that WM has C slots, and that each slot can store one of the previously encountered observations from the environment. The agent can modify the content of WM slots at each time step via the gating actions, which allow the agent to either store the new observation in one of the slots while maintaining the content of the other ones, or maintain the content of all memory slots. The gating actions are chosen by the agent along side the motor actions.

The inclusion of the additional WM state does not change the main dynamics of XKTD-SARSA—that is, the updating equations in 5.10 remain the same. There are only two slight modifications: the first one consists of the current state s_t now being composed of the current observation o_t and the current memory content $m_t = (m_t^1, \dots, m_t^C)$, where m_t^j is the observation held in the j^{th} WM slot at time t. The second modification is that action a_t now includes a motor action (a_t^M) and a gating action (a_t^G). Hence, each state-action pair z can be written $z = (m, o, a^M, a^G)$, where m, o, a^M, and a^G are respectively the current memory content, observation, motor action and gating action. The steps followed by the agent in the augmented XKTD-SARSA model, which we refer to as the Bayesian RLWM can be summarised as follows:

1. Choose the prior mean and covariance of the Q-value vector.

2. Observe current state $s_t = (m_t, o_t)$.

3. Select a motor action a^M and a gating action a^G based on state-action values

Q(s,.): $(a^M, a^G) \leftarrow Thompson(Q, s)$.

4. Observe reward r_t and go to next state s_{t+1}.

5. Update the mean and covariance matrix of the Q-values using the updating equations in 5.10 .

6. Repeat steps 2-5 until termination.

There is one main difference between the framing of the Bayesian RLWM and the original RLWM (defined in Chapter 3): In the Bayesian version, we consider that there is only one state-action value function that controls both motor and gating actions, as opposed to having a separate Q-value function for for each type of action. We chose to use one state-action value function here to maintain all information about the covariances between different state-action pairs, as some of this information might be lost if we break the link between the motor and gating state-action pairs.

5.3.2 Model settings

To evaluate the Bayesian WM-based RL model presented here, we applied it to the two simplified versions of the 12-AX task that we introduced in Chapter 3 (see Section 3.3.2): the 12-XY and 12-AB-XY. We simulated the model on each task for 30 times, each run consisted of $N = 10^4$ trials on the 12-XY, and $N = 5 \times 10^5$ on the 12-AB-XY.

We initialised the prior mean $\boldsymbol{\hat{X}_0} = \mathbf{0}$ and prior covariance $\boldsymbol{\hat{P}'_0} = 10I_{d+2}$ (where I_{d+2} is the identity matrix of dimension $d+2$, such that d is the number of possible state-action pairs in the task to solve). We set the covariance of the state noise $\boldsymbol{P_{V_t}} = 10^{-10}I_d$ (although the learning problems that we used are stationary, we did not use a null matrix to avoid numerical stability issues). Also, we chose $\gamma = 0.75$

(note that this is lower than the value that we used in the previous chapters, which was equal to 0.95. We lowered the value of γ because the Bayesian model could not successfully learn the tasks with such a high value). The only parameter in the Bayesian model that we allowed to vary was σ^2, the variance of the white noise u_t. In the case of the 12-XY, we did an extensive grid search to obtain the best value for σ^2, which was equal to 0.5, whereas on the 12-AB-XY, we chose $\sigma^2 = 0.1$ after trying a few possible values. We did not perform an extensive grid search for the 12-AB-XY task, because it took about four hours to complete each simulation run, mainly due to the large covariance matrices involved (we run most of the simulations on an Intel Core i7 computer, and implemented the Bayesian model using sparse vectors and matrices, which made the running time even faster). Hence, other values of σ^2 could be potentially found to make the Bayesian model perform better on the 12-AB-XY task.

Finally, we compared the Bayesian RLWM model against the standard RLWM model (see Chapter 3) that is enhanced with Dabney and Barto's (2012) adaptive learning rate (see Section 4.2.2). The standard RLWM has two free parameters T, the exploration rate, and λ, the decay parameter of the eligibility trace function (γ being equal to 0.75). Our extensive grid search produced $T = 1$ and $\lambda = 0.7$ on the 12-XY task, and $T = 1$ and $\lambda = 0.9$ on the 12-AB-XY task. Table D.1 summarises the parameter values used for each model in each learning task.

5.4 Results

5.4.1 12-XY task

We first simulated the Bayesian and standard RLWM, both with one WM cell on the 12-XY (see Figure 5.1). Results show that the Bayesian version outperformed the non-Bayesian version in terms of the time required to reach their top performance. For example, the Bayesian model could reach the criterion of 95% response success rate in 100 trials after about 735 trials, whereas the conventional RLWM model reached the same learning criterion after about 1308 trials. In terms of the final average accuracy, both models could converge toward the maximal average accuracy by the last block of 500 trials.

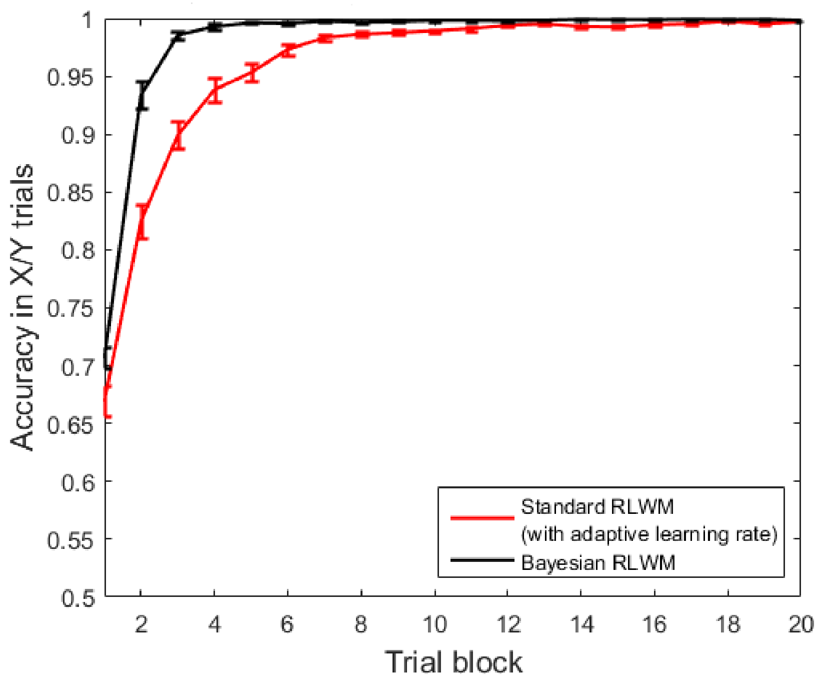


Figure 5.1: Learning curves for the standard and Bayesian RLWM model, both with one WM cell, on the 12-XY task. Accuracy was averaged over blocks of 500 trials and over 30 simulation runs. Error bars represent standard errors.

5.4.2 12-AB-XY task

On the 12-AB-XY task, the Bayesian RLWM model performed slightly worse than the non-Bayesian version both in terms of speed and final average accuracy (Figure 5.2). The Bayesian model reached an average accuracy of 0.98 in the last block of 25000 trials, with a 95% time-to-criterion approximately equal to 47600. In contrast, the standard RLWM had a final average accuracy of 0.99, reaching the 95% criterion, in average, after about 33300 trials. However, as we previously mentioned, we performed only a light grid search in the parameter space in the Bayesian case in comparison with the non-Bayesian case, due to the slow running speed of the Bayesian model.

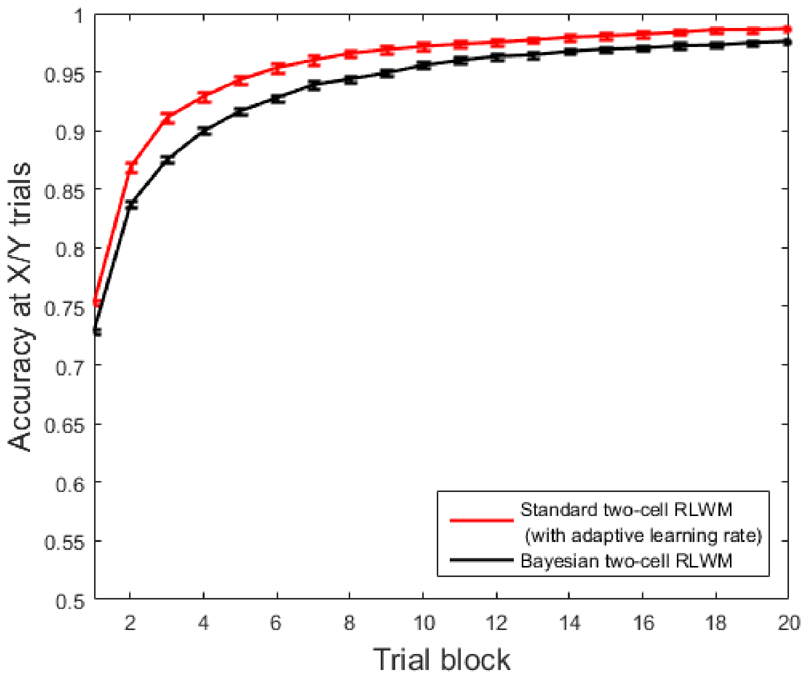


Figure 5.2: Learning curves for the standard and Bayesian RLWM model, both with two WM cells, on the 12-AB-XY task. Accuracy was averaged over blocks of 25000 trials and over 30 simulation runs. Error bars represent standard errors.

5.5 Discussion

We have investigated the Kalman temporal difference (KTD) approach to Bayesian RL, and its extension XKTD. We have shown that they are closely related to the standard SARSA(λ) algorithm. We noted that the full Bayesian framework results in a natural replacement for eligibility traces, and allows the use of Thompson sampling, which also removes the need to parametrise the action-selection mechanism. We then tested the XKTD model on two WM learning problems, after enhancing it with a WM module as in Todd et al. (2009), Zilli and Hasselmo (2008) and Lloyd et al. (2012). Our preliminary results demonstrate the potential of the approach, especially that it simply requires two free parameters to set, which are σ^2, the variance of the white noise, and γ, the discount parameter.

Simulation results showed that the proposed Bayesian RLWM model outperformed the conventional RLWM with adaptive learning rate on the 12-XY—a moderately challenging WM learning task. In contrast, on the more complex 12-AB-XY task, the Bayesian model resulted in a slightly lower performance. However, we were limited by its large computational requirements for updating the state-action values (specifically, the updating of the covariance matrix the Q-values), which prevented us here from searching the parameter space more exhaustively to maximise learning performance. Hence, the results on the 12-AB-XY do not reflect the full potential of the Bayesian model, but instead show its potential as an alternative approach to conventional WM-based RL methods.

The question of how to alleviate the computational burdens that are often associated with the use of Bayesian and other types of RL algorithms is an active area of research (see for example, Geramifard et al., 2006). One of the main solutions that have been proposed in the literature is to use value function approximation, where the Q-value functions are represented in parameterised functional forms, and where the number of parameters is considerably lower than the number of state-action pairs. This is, for example, what was used in the original version of the XKTD model (Geist and Pietquin, 2010). Here, we did not use parameterised Q-value functions to simplify the exposition and interpretation of the algorithms (to allow for the possibility to use the models to explain human behaviour in future studies). A solution that might not hinder the plausibility of the Bayesian model would be to update only a few covariance values, since in most learning domains, only some state-action pairs will be related. For example, one might to choose to update only the n largest covariances (e.g., $n = 100$).

Future research should also compare the Bayesian RL model with standard RL models, such as the SARSA(λ) algorithm in memoryless partially observable problems (i.e., those that do not require the addition of memory to be solved). This would test whether the updating scheme of the Bayesian model—based on the covariances between the state-action values—is a good alternative to eligibility traces, which are often used to solve memoryless partially observable tasks (see for example, Loch and Singh, 1998).

The Bayesian approach potentially offers a more natural way of specifying the adaptive resource allocation model, because the Bayesian RLWM model carries information about the precision of the Q-values estimates (i.e. the variances). But how could we augment the Bayesian RLWM using the gradual allocation approach? There are three points to take into consideration for this purpose: (1) which criterion to use to decide when to increase WM resources? (2) how to transfer the means of the Q-values after each capacity switch? and (3) how to transfer the covariances of the Q-values? We can base the criterion for switching to a higher WM capacity on the precisions of the Q-value estimates (given by the variances of the Q-values). For example, we can assume that the learner switches to a higher WM capacity if the sum of the variances becomes lower than a small number (suggesting that the algorithm has converged with the currently allocated WM resources). Concerning learning transfer, we can copy the means and covariances of the Q-values similarly to how we copied the point estimates of the Q-values under the flexible resource allocation model (see section 4.2.1). A more conservative approach would be to copy just the mean values and leave the model to relearn the covariances between the different state-action values.

This chapter concludes our modelling work on WM-based RL using the Actor-Critic gating model of Todd et al. (2009). The modifications and enhancements that we introduced to the model has significantly improved its learning speed, taking it from a learning time that is measured in millions of time steps in complex non-Markovian tasks, such as the 12-AX, to a learning time that is measured in hundreds of thousands of trials. Given our simulation results, one task that seems suitable for testing human performance is the 12-XY. In the next chapter we will use an episodic version of this task to investigate the role of WM in non-Markovian RL settings. The purpose of the study will be to check whether people rely on WM to solve non-Markovian RL tasks (or on another memory system), and if it is the case, to find out how people use WM when performing this type of learning tasks.

Chapter 6

The role of working memory in human reinforcement learning of non-Markovian tasks

6.1 Introduction

Working memory-based reinforcement learning models under the gating framework take for granted the assumption that working memory (WM) supports reinforcement learning (RL) in non-Markovian environments, by keeping track of previous events and allowing them to take part of the learning context. Although this framework is compelling from a theoretical perspective, it is not clear whether people would indeed rely on WM or on another memory system such as episodic memory when given a non-Markovian task (Zilli and Hasselmo, 2008).

WM has been shown to be involved in different types of learning, including rule-based category learning (Ashby and Spiering, 2004; DeCaro et al., 2008), model-based RL (Otto et al., 2013b,a) and model-free RL (Collins and Frank, 2012; Collins et al., 2014). However, the contributions of WM in these studies mainly consist of either retaining task instructions, storing freshly learned action-outcome associations, maintaining hypotheses to test in subsequent trials, or supporting trial-to-trial adaptation after negative outcomes.

Among these studies, the most relevant one to the present work is the paper by Collins and Frank (2012), which also looked at the role of WM in model-free RL. Collins and Frank (2012) used a relatively simple Markovian task, with deterministic feedback and state-response contingencies that did not involve past events. They showed that a hybrid RL model that has a WM-based learning component better accounted for participants' observed choices than a pure RL model. Their hybrid model was also able to capture previously observed effects in learning experiments such as a bigger involvement of WM in early learning stages in comparison with late stages (Doll et al., 2011). Further support for the separation between RL and WM in their model came from a link found between two genetic markers of the basal ganglia function (which is associated with model-free RL) and the learning rate in the RL part of their model, and an association between a genetic marker of the prefrontal cortex function (which is usually associated with WM) and the WM capacity estimate of their model.

The aim of the present work is to investigate the potential additional role of WM in keeping track of useful cues from the past to serve action selection in non-Markovian settings. For that, we designed an experiment where we measured participants' available WM capacity while doing a simple non-Markovian task based on the 12-AX (Frank et al., 2001), and compared it with their estimated WM capacity in a similar Markovian task. A smaller available WM capacity in the non-Markovian condition would indicate that people use those extra WM resources to solve non-Markovian tasks, as hypothesised by WM-based RL models. We also examined how participants used their WM resources over the course of the learning tasks to see if their allocated WM capacity varied over time, as predicted by the resource allocation model introduced in Chapter 4.

6.2 Material and methods

6.2.1 Participants

Twenty-six volunteers (14 female, mean age 21 ± 3.4 years) took part in the experiment, most of them undergraduate students from the University of Bristol. Before the start of the experiment, participants were informed about the general purpose of the study and gave a written informed consent. All participants had normal or corrected-to-normal vision and did not suffer from any learning disabilities. Participants were paid up to £13 depending on their performance on two separate tasks, on top of a fixed amount of £5. The study conformed to the Helsinki Declaration and was approved by the local faculty ethics committee.

6.2.2 Apparatus

All experiment scripts were written in Matlab (MathWorks, Inc.), using the Psychtoolbox extensions (Brainard, 1997; Pelli, 1997; Kleiner et al., 2007). The experiment was run on an Intel Core i7 (2.7 GHz) computer running Windows XP, and equipped with an Acer 1716 17-inch CRT monitor running at 60 Hz with a screen resolution of 1280×1024.

6.2.3 Design

A repeated measures design was used to test whether people would use more WM resources in a non-Markovian task than in a Markovian version of it. In the test condition, participants' WM capacities were measured using a WM span task, while they were performing the non-Markovian task. In the control condition, their WM capacities were measured while they were performing the Markovian task. The order of presentation of the tasks was counterbalanced, so that half of the participants had to perform the non-Markovian task first; the other half were assigned the tasks in the reversed order.

6.2.4 Procedure

Non-Markovian task: episodic 12-XY

A simplified version of the 12-XY (see Section 3.3.2) was used as a non-Markovian learning task, which we call here the "episodic 12-XY". We chose this task on the basis of some initial model simulations, showing that the RLWM model (see Chapter 3), most of the time, could learn the rules of the task within the number of trials used in the experiment. Also, our pre-experiment pilots showed that increasing the difficulty of the non-Markovian task—for example, by using an episodic and simplified version of the 12-AB-XY task—results in participants no longer being able to learn the main learning task in our dual task setting. Hence, although the episodic 12-XY task seems too simple, from the perspective of our participants who also had to perform a WM span task (more details are given below), this task was challenging enough to probe the learning function and WM.

The task was divided into 240 episodes (8 blocks), in each of which, one, two or three digits 1 or 2 were presented, followed by a target X or Y (Figure 6.1). After each target, participants had to choose a response between the "left arrow" (L) and "right arrow" (R) key. The participants' task was to learn to correctly respond to X and Y using the reward feedback that were given to them after each of their decisions. Reward feedback were provided according to the same rules of the 12-XY task: if the last digit was 1 (resp. 2) and the current letter is X (resp. Y), then choose R response. In all other cases, choose L. Reward was either 10 points for correct responses or 0 points for incorrect responses. No reward was given when a digit is presented, although participants were required to respond to the digit cues by pressing the keyboard key that corresponds to the digit presented on the screen (for example, if 2 was presented, they had to press the key "2" on the keyboard). This was to make sure that they attended to them.

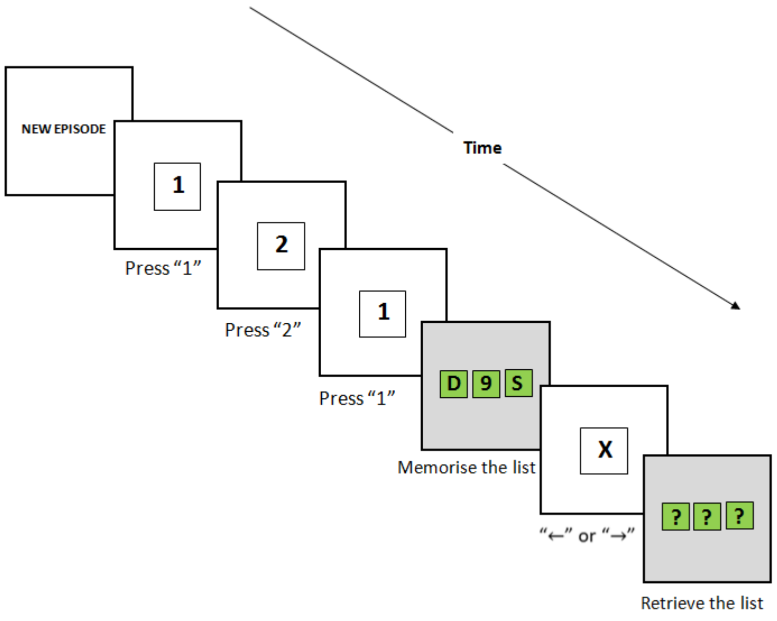


Figure 6.1: Example of a sequence in one episode. Each time participants saw a digit, they had to press the keyboard key that corresponds to that digit. After the last digit cue, they had to retain a span list. Participants then responded to the target letters by choosing the "left arrow" or "right arrow" key. Finally, they were asked to retrieve the span list that they previously memorised.

In order to measure their available WM capacity, participants were also asked to perform a span task, where they had to retain lists of characters—digits and letters—of variable length. Participants were displayed with the span list just before the target, and were required to memorise it until the end of the episode. We chose to display the span list just after the last cue because we wanted the span task to measure participants' remaining WM capacity after they had stored the cues (for example, putting the span list at the onset of the episode would have possibly led the participants to prioritise the span task over the main learning task). After giving their response to the target, they were prompted to remember and enter the characters of the span list in the same order that was shown to them.

Participants were instructed that their primary task is to try to learn to respond correctly to X and Y, but they also should try to perform as well as possible in the memory task because it allows us to measure their memory capacity (we did not give participants monetary reward for performing the WM span task because we found in our pre-experiment pilots that most participants ended up prioritising the span task over the main learning task). Participants were specifically told to give enough memory resources to the learning task, and only use the remaining resources to the memory task. In addition, participants were informed that the memory task was completely independent from the learning task. Finally, participants were given the two following guidelines about the learning task:

- The correct responses to X and Y observations depend on the current and **exactly one** of the preceding characters within the current episode.
- Rewards are deterministic and reliable—that is, a correct response will always trigger a positive reward feedback.

Markovian task

This task was composed of 120 episodes (4 blocks), and had the same structure as the non-Markovian task. The only difference was that the correct responses to the targets X and Y no longer depended on past observations. The rules to be learned in this task were simply: select L when X is encountered and R when Y is observed (randomly counterbalanced between participants by switching the two respective responses to X and Y).

Working memory span task

Being able to estimate the span very quickly and reliably were two important requirements in our paradigm, since we were interested in measuring participants' WM span throughout the experiment. Hence, we used an adaptive method based on two randomly interleaved staircases to estimate WM span (Cornsweet, 1962). Woods et al. (2011) demonstrated the benefit of using adaptive over traditional methods for span estimation (for example where the estimation test is terminated after two successive errors). Among the advantages that they listed were the reduction of the time required to measure WM span and the enhancement of the reliability and precision of the estimates.

More specifically, we used a transformed 1-up/2-down rule (Levitt, 1971; Kaernbach, 1991), where list lengths in the span task were decreased by one after a wrong response, and increased by one after two successive correct responses (the name of the rule might seems the wrong way around given its description, but the convention in psychophysics is to use stimuli, such as the sound of a click, which are made easier by augmenting their intensity and made harder by reducing their intensity; in our case, it is the inverse: the list of items to memorise is made easier by decreasing its length and made harder by augmenting its length). This method targets a 70.71% threshold (Levitt, 1971)—that is, we can obtain the probability of correct recall at 70.71% by simply averaging the list lengths generated on a given interval of interest, after the staircase has converged (in our case, we considered that the two staircases has converged when they have crossed each other).

The main reason for using a double-staircase instead of a single-staircase was to minimise the possibility that subjects detect the rules that govern the generation of list lengths, which may bias their responses. The two staircases were initialised at list lengths of two and six.

6.2.5 Analyses

The decision of whether a participant managed to learn the task rules or not was based on two criteria: First, the type of errors a participant made was examined. This was important given the nature of our task. In fact, some sequences were easier to learn to respond to than others. Hence, high performance does not imply that a participant has fully learned the rules. For example, a participant might correctly respond to sequences that have only one digit cue, but not to sequences having multiple cues. Indeed, the former type of sequence requires a participant to simply remember which button to choose for each combination target$\times$digit cue, without worrying about which digit cue to retain (remember that participants were told that the correct response in each target depended on exactly one of the preceding characters from the same episode). Similarly, sequences that have multiple similar digits, such as 1-1-1-Y, should be easier to learn too.

Second, a criterion of success rate in a sequence of several responses was set up to detect whether a participant managed to learn the task, and if it is the case, in which trial it happened (i.e., time-to-criterion; see Section 3.2.4). The chosen criterion was 95% accuracy on 30 successive responses. This was chosen so that only participants who learned to correctly respond to all types of sequences (i.e., with one cue, multiple similar cues or multiple different cues) would be detected as real learners. This criterion also accurately classified participants in terms of who managed to report the correct rules in the debriefing from those who did not.

6.3 Results

6.3.1 How many participants manage to learn the non-Markovian task?

16 out of 26 participants (61.5%) managed to fully learn the rules of the non-Markovian task, as judged by their time-to-criterion (see Table 6.1). These participants also successfully reported the correct rules of the task in the debriefing session. Moreover, all participants who fully learned the task, except two (the 19th and 21st participant), did it by the end of the 3rd block. A detailed analysis of participants' error rate per sequence type also confirmed the previous finding (see Figure 6.2), but also showed that 3 other participants (2nd, 10th and 11th) partially learned the rules, which allowed them to perform substantially above chance. They learned to correctly respond to the targets when the sequences contained the same digit (e.g., 1-X, 1-Y, 1-1-1-Y), but not when they contained multiple different digits such as 1-2-1-X.

This leaves us with what appear to be three categories of learners: those who fully learned the task (full-learners), those who partially learned it (half-learners) and those whose performance remained at around chance level (non-learners). This three-way classification by eye was confirmed using the K-means method based on the Euclidean distance (MacQueen, 1967). The number of classes (three) was chosen based on the average silhouette value (Rousseeuw, 1987), after trying different numbers of classes (the silhouette value was maximal for two or three clusters with a value of 0.86). Figure 6.3 shows the output of the K-means algorithm. The first cluster was characterised by an error rate of around 50% for each of the three sequence types; this corresponds to the group of non-learners. The second cluster contained participants whose error rates were almost null for all sequence types (full-

Participant	Markovian task	Non-Markovian task
1	9	86
2	1	Half-learner
3	1	33
4	1	Non-learner
5	1	Non-learner
6	1	62
7	49	Non-learner
8	1	12
9	18	24
10	20	Half-learner
11	1	Half-learner
12	2	52
13	25	6
14	36	Non-learner
15	1	9
16	1	40
17	3	60
18	2	Non-learner
19	1	198
20	5	44
21	1	106
22	1	77
23	1	Non-learner
24	1	17
25	58	Non-learner
26	1	43

Table 6.1: Time-to-criterion in the Markovian and non-Markovian task for each subject. For the subjects who could not reach the 95% criterion within the learning time, we display their category of learner instead of time-to-criterion.

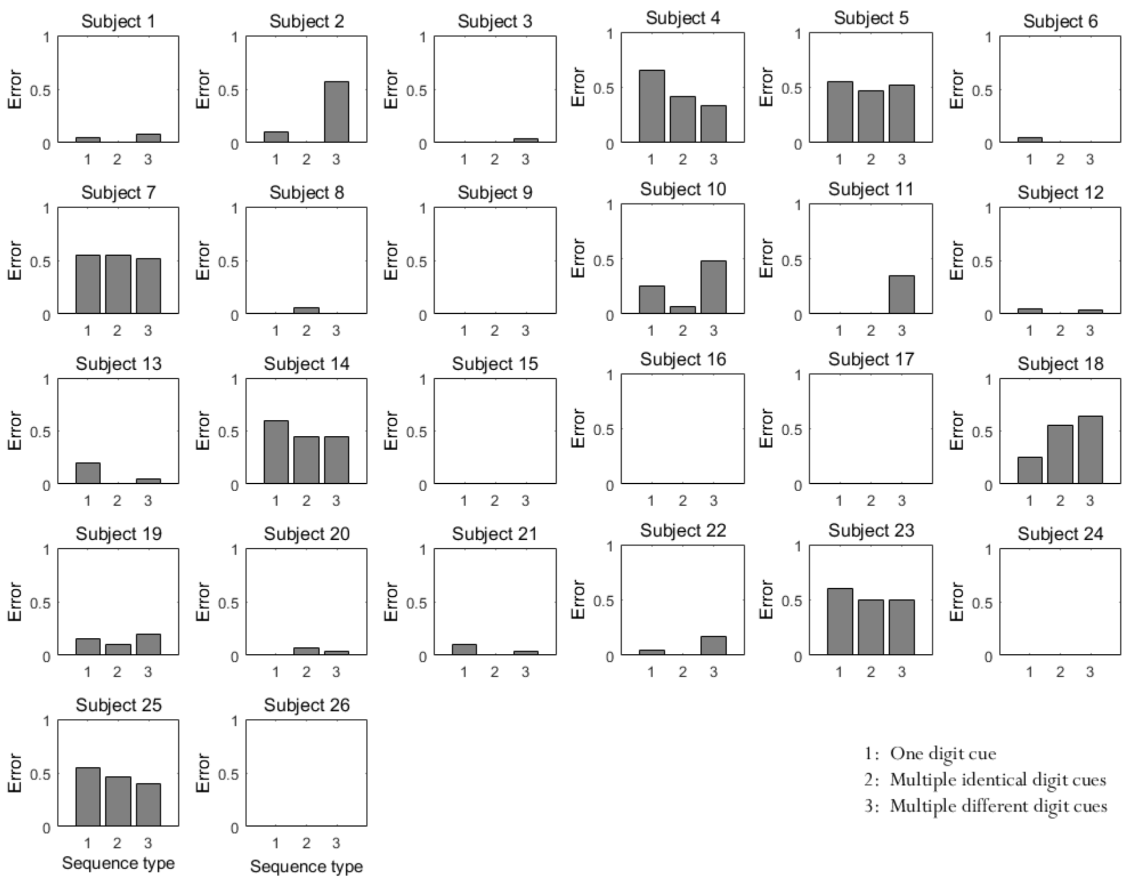


Figure 6.2: Error frequency in the last two blocks (i.e., last 60 episodes) per subject and sequence type.

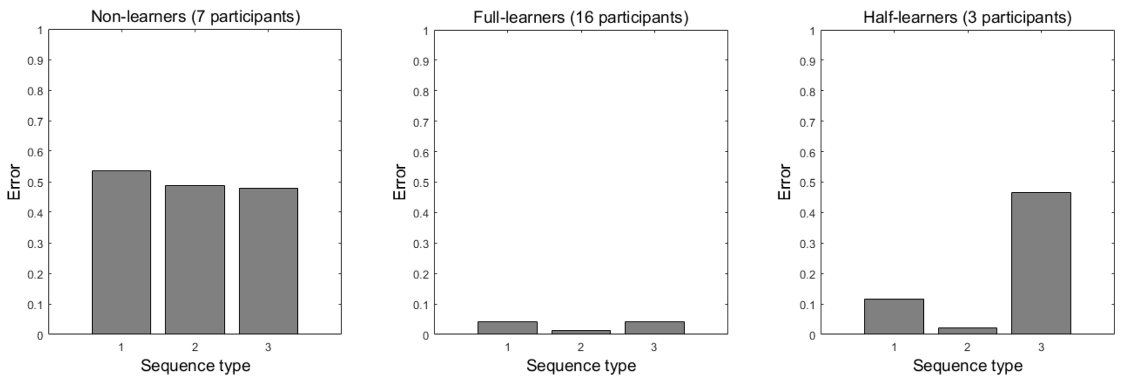


Figure 6.3: K-means classification output.

learners). The third and last category described those participants who made errors mainly in sequences with multiple different digit cues (half-learners).

Concerning the Markovian task, all subjects learned the correct rules and reached the 95% criterion by the end of the second block (see Table 6.1).

6.3.2 Was working memory used to support reinforcement learning?

Overall WM span scores were generally lower in the episodic 12-XY task in comparison with the Markovian task for all types of learners (see Figure 6.4). A mixed-effects analysis with average WM span as the outcome, with task type (Markovian versus non-Markovian) and learner category (non-learner, half-learner and full-learner) entered in this order as fixed effects, and with subjects entered as a random effect, revealed a main effect of task type ($\chi^2(1) = 19.28, p < .001$), indicating that WM span measured in the non-Markovian task ($M = 3.56, SD = 1.22$) was lower than that measured in the Markovian task ($M = 4.75, SD = 0.59$); there was no main effect of learner category ($\chi^2(2) = 4.38, p = .112$). There was an interaction of task type with learner category ($\chi^2(2) = 11.09, p = .004$), indicating that the effect of task type was different depending on the learner type.

Pairwise t-tests revealed that in the Markovian task, there was no effect between any pair of learner categories (all p-values $> .05$). However, in the non-Markovian task, WM scores differed significantly between full-learners and half-learners ($t(17) = -2.95, p = .005, d = -1.37$), half-learners and non-learners ($t(8) = 3.63, p < .001, d = -2.91$), but not between full-learners and non-learners ($p = 0.158$). The pairwise tests also revealed that the difference in WM span between the non-Markovian and Markovian task was significant for full-learners ($t(15) = 4.06, p = .001, d = 1.11$), half-learners ($t(2) = 4.34, p = .049, d = 4.33$), and non-learners

$(t(6) = 2.98, p = .025, d = 1.05)$.

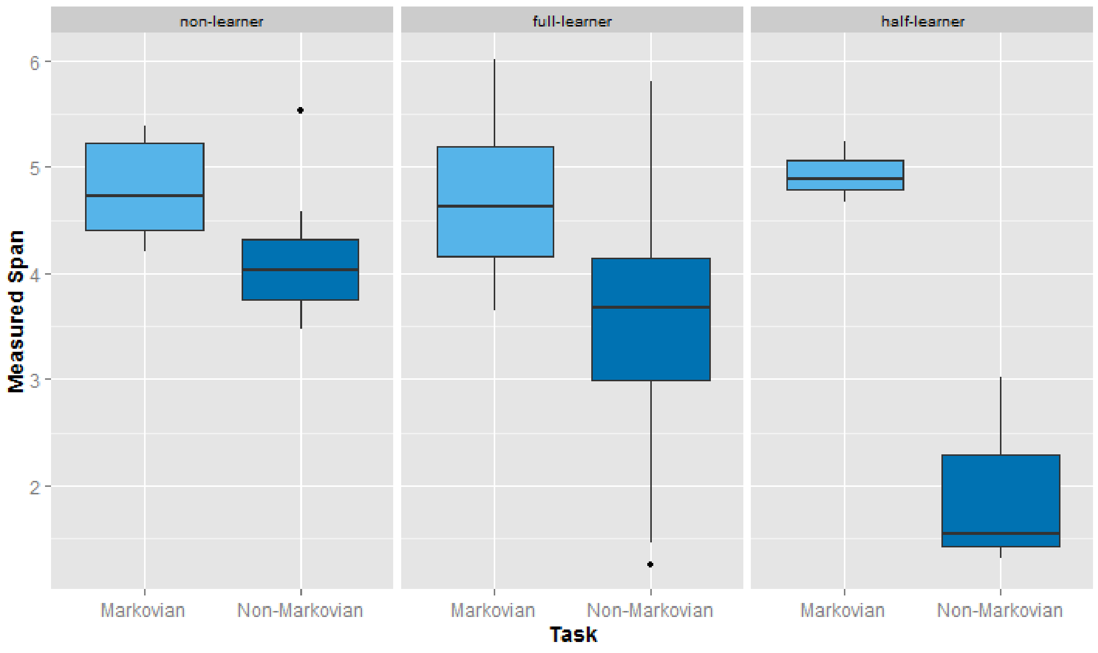


Figure 6.4: Box plots showing the distribution of measured span in each task condition for full-learners, half-learners and non-learners. The black points indicate outliers.

The significant results from the analyses that included the half-learner group should be considered with caution since the group contained only three individuals. The effect sizes might have been overestimated, and so those significant effects need to be verified using a larger group size. Thus, we repeated the previous analyses but now taking into account only full-learners and non-learners (half-learners will also be excluded from all remaining analyses). The new results showed again a main effect of task type $(\chi^2(1) = 16.23, p < .001)$; but not of learner category $(\chi^2(2) = 1.03, p = .309)$. Also, there was no interaction between task type and learner category $(\chi^2(2) = 1.24, p = .266)$, indicating that the effect of task type was similar for full-learners and non-learners.

In our experiment, the order of presentation of the non-Markovian task was counterbalanced, so that half of our participants had to perform the non-Markovian task after the Markovian task. In order to check that the effect of task type on measured WM span does not depend on whether the non-Markovian task was presented first or last, we tested the effect of presentation order on the difference between WM span measured in the Markovian task and that measured in the non-Markovian task (i.e., $Span_{Mark} - Span_{NonMark}$). A two-way independent ANOVA (see Figure 6.5) revealed that learner category did not have a significant effect ($F(1, 19) = 0.83, p = .374$), which is in line with the non-significant interaction between task type and learner category that we obtained previously. Also, there was neither a significant main effect of presentation order ($F(1, 19) = 1.45, p = .244$), nor was there an interaction between the two variables ($F(1, 19) = 0.14, p = 0.712$).

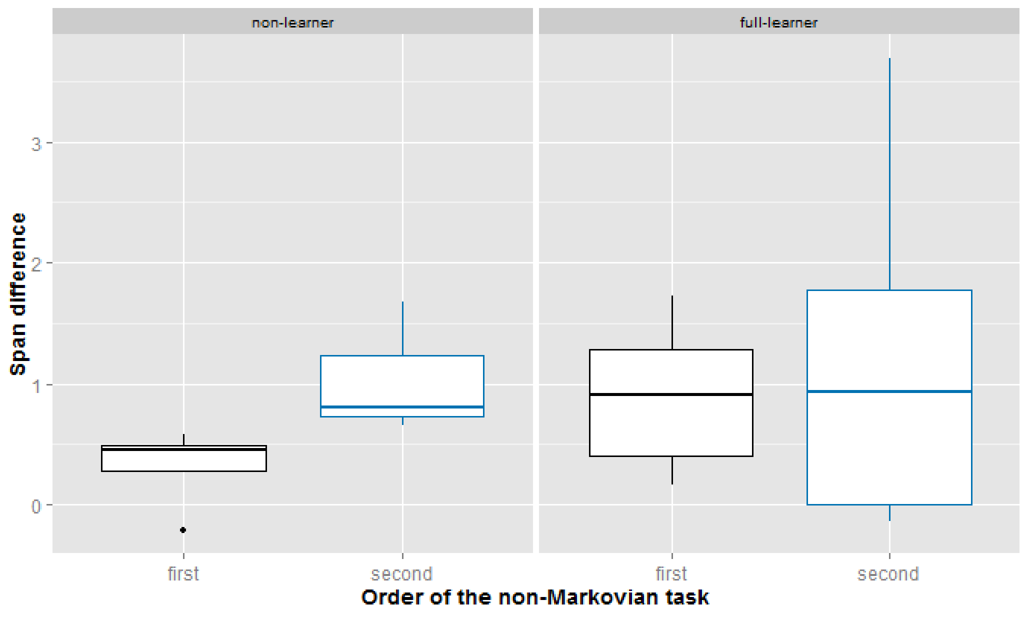


Figure 6.5: Box plots showing the effects of learner category and presentation order on the difference between the span measured in the Markovian task and the span measured in the non-Markovian task. The black points indicate outliers.

These results suggest that indeed WM was used to support RL in our non-Markovian setting, as assumed in the WM-based RL framework. They also show that the overall amount of WM resources allocated to the episodic 12-XY in comparison with the Markovian task was, in general, similar for full-learners and non-learners, suggesting that, maybe, it is not the overall WM capacity used in the 12-XY that determines whether a person can fully learn the episodic 12-XY task or not (ignoring the possibility of partially learning the task). A possible answer, as suggested by our modelling work in Chapter 4, might be that the key factor is how the person allocates her WM resources throughout the task. This is what we are going to look at in the next section.

6.3.3 How did participants use their working memory while learning?

In order to have an idea about how WM span varied over time, we repeated the previous analysis, but now including a factor block, which breaks up each WM span score into the span measured in the first and last block of each task. Hence, we ran a mixed-effects linear model with subjects entered as a random effect along with these fixed-effect factors: task type (Markovian versus non-Markovian), learner category (non-learner or full-learner) and block (first or last). Results (Figure 6.6) show a main effect of block ($\chi^2(1) = 37.74, p < .001$), indicating that WM span increased in the last block ($M = 4.63, SD = 1.03$) as compared with the initial block ($M = 3.70, SD = 0.92$). However, there was no interaction between block and task type ($\chi^2(1) = 1.28, p = .258$) or between block and learner category ($\chi^2(2) = 0.23, p = .628$), and no three-way interaction ($\chi^2(2) = 0.04, p = .844$). All other effects and interactions involving task type and learner category were in the same direction as in the previous analysis.

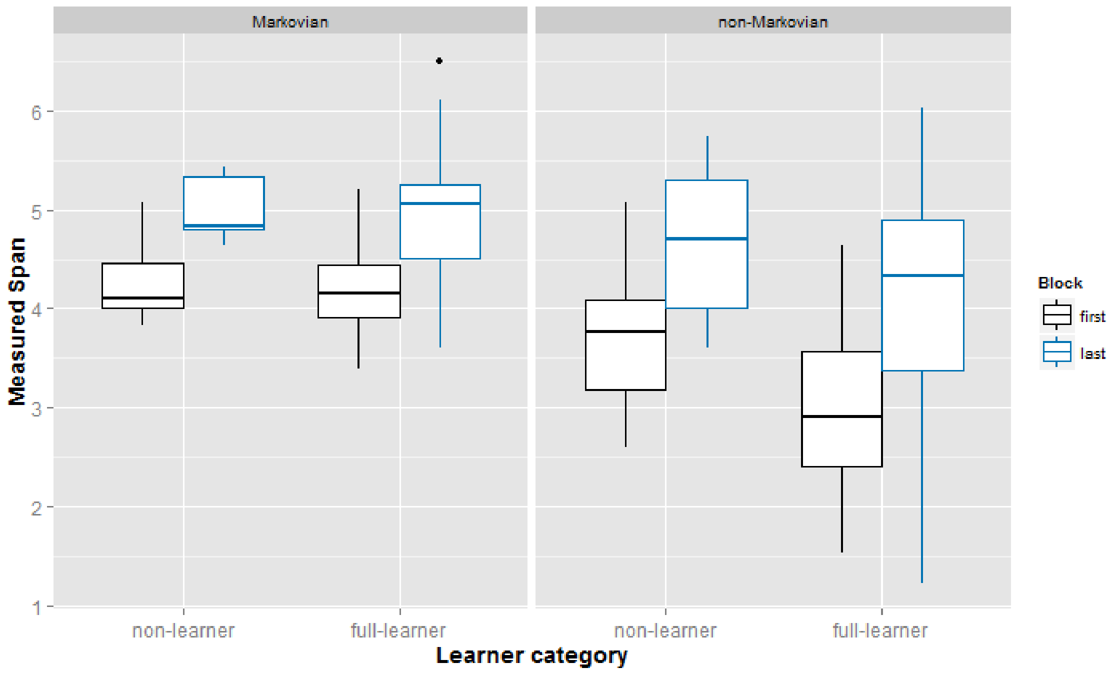


Figure 6.6: Comparison of WM span between the first and last block in both the Markovian and non-Markovian task.

The moving average of WM span also seemed to increase, in general, over time in both tasks and for both categories of learners, confirming our last result (see Figure 6.7). At first glance, this result seems to go against what is usually assumed in WM-based RL models, which is that the learner uses a fixed WM capacity throughout the learning task. It also seems to contradict the predictions of our flexible resource allocation model, which considers that WM resources are allocated gradually to the non-Markovian task, as opposed to decrease over time as we found here (however, this is valid only if we assume that WM resources that are not used in the span task are systematically allocated to solve the episodic 12-XY task). But if this increase in measured WM span was due to a strategic choice from participants, why did it happen also in the Markovian task? Could it be simply due to a training effect?

In order to test whether the increase in WM span scores in the last block in

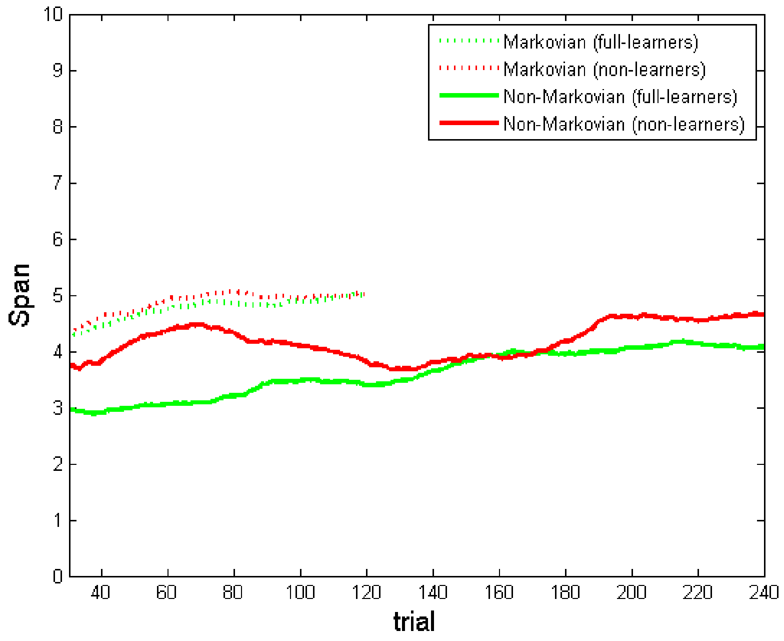


Figure 6.7: Running average span score of the "average full-learner" and "average non-learner" with a window size of 30, in both the Markovian and non-Markovian condition.

comparison with the initial block was due to training or not, we analysed the effect of presentation order (i.e., whether the task was presented first or second) on WM span scores in the first block of the Markovian task (see Figure 6.8). If there were a training effect, we would expect the measured WM span in the first block of the Markovian task to be bigger after completing the non-Markovian task, than if the Markovian task was presented first. This is because the participant, in the former case, would be more familiar with the task structure and instructions, and particularly with the WM span task. Results using a two-way independent ANOVA showed that presentation order did not have a significant effect ($F(1, 19) = 0.19, p = .669$), nor did it interact with learner category ($F(1, 19) = 0.01, p = .929$), suggesting that no transfer occurred between the two tasks for both full-learners and non-learners.

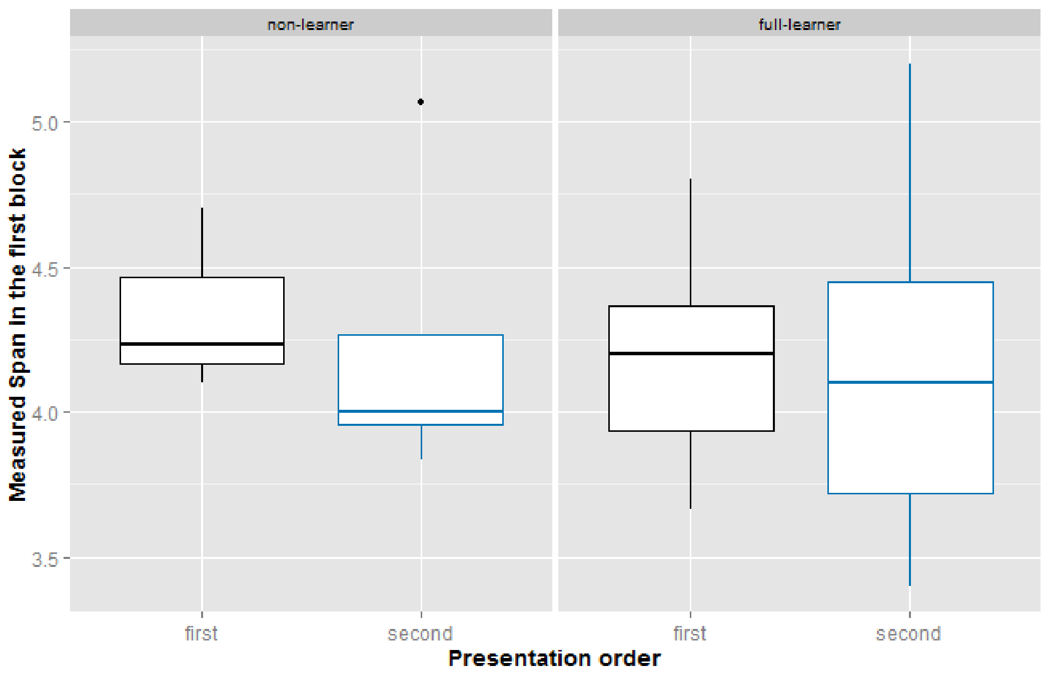


Figure 6.8: Box plots showing the effects of presentation order on measured span in the first block of the Markovian task, for both full-learners and non-learners. The black points indicate outliers.

Regardless of the cause of the increase in available WM capacity over time in both tasks, we can fairly assume that the extra factors that might concern us were controlled for in the Markovian condition (the only difference between the two tasks was that the non-Markovian task had mapping rules that depended on the past). In order to test whether the increase in WM capacity between the first and last block of the non-Markovian task was systematic (i.e., due to the use of non-Markovian rules) or not, we analysed the effect of task type (Markovian versus non-Markovian) on WM span difference between the first and last block (i.e., $Span_{last\ block} - Span_{first\ block}$). Thus, we ran a mixed-effects linear model with task type (Markovian versus non-Markovian) and learner category (non-learner or full-learner) as factors. The analysis (Figure 6.9) showed that there was no main effect of task type ($\chi^2(1) = 1.43, p =$

.232) or learner category ($\chi^2(1) = 0.21, p = .648$). Also, the interaction between the two variables was non-significant ($\chi^2(1) = 0.04, p = .834$). These results suggest that the increase in WM capacity in the last block in comparison with the first block was not due to the use of non-Markovian task rules, and that this was true for both full-learners and non-learners. This supports the commonly used assumption in WM-based RL models, which is that WM resources that are used to store past observations remain fixed throughout the learning task.

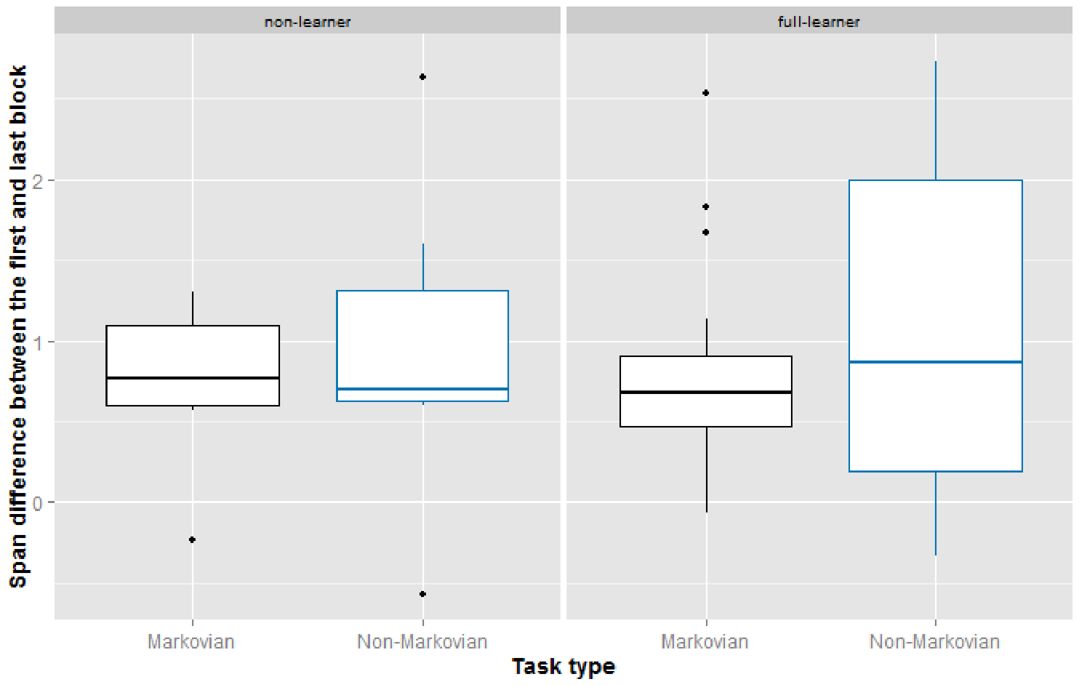


Figure 6.9: Box plots showing the effect of task type on the difference of measured span in the first block and last block, for full-learners and non-learners. The black points indicate outliers.

6.4 Discussion

WM-based RL models predict that people would use more WM resources in the non-Markovian task as compared with the Markovian task. Our results support that hypothesis, suggesting that WM was used to supplement RL in the non-Markovian task. However, surprisingly only 61.5% of our participants managed to learn the rules of the episodic 12-XY, which is a very simple version of the 12-AX task. This is a lower learning rate in comparison with the 83.3% found in Krueger (2011) using the original 12-AX, but in our case, the overall tasks were made more complicated by the addition of the WM span task.

We also showed that the measured WM capacity increased in the last block as compared with the initial block in both the Markovian and non-Markovian task (i.e., WM resources that are allocated to each task decreased over time). However, the pattern of increase did not differ between the two tasks, suggesting that it does not reflect a strategic use of WM resources to remember past observations in the non-Markovian condition. The fact that we told participants that only one past cue is important in their decision at the target was probably determinant here. Thus, further experiments should be carried out to investigate more this result, but in any future experiment, participants should be given the freedom to choose the number of cues to keep track.

The decrease in WM load over time in the case of the Markovian task (reflected by the increase in measured WM span), although somehow unexpected since the mapping rules to be learned were simple, has also been reported in other studies. For example, Collins and Frank (2012) showed that WM load effects (such as the sensitivity to the number of stimuli or the delay until the same stimulus is observed) disappeared as learning progresses in their Markovian learning task. Further support for this result comes from studies combining computational modelling and neurogenetics. In fact, Doll et al. (2011) found that the link between model-free learning rates and genetic markers of the basal ganglia function (which is associated with model-free RL) becomes apparent only after initial acquisition. Instead, during initial learning, it is a genetic marker of the prefrontal cortex function (which is usually linked with WM) that is strongly associated with learning rates of hypothesis testing (see also Collins and Frank, 2012, for a more detailed discussion).

Collins and Frank (2012) accounted for those WM load effects using a hybrid RL model that includes a WM-based learning component. A simpler alternative explanation is that WM capacity might have been freed up at the end of our Markovian task simply because participants have learned the mapping rules and transferred them into their long-term memory (i.e., participants were using WM to keep track of the task rules). Another possibility is that the observed increase in WM span scores was simply due to a training effect in our experiment; however, our analyses ruled out this possibility (see the end of Section 6.3.3).

The increase pattern might also simply reflect the convergence of WM span estimate-by-average towards the true WM span. In fact, since we initialised the double-staircase at 2 and 6 (i.e., the initial estimate of WM span was equal to 4), the estimated WM span of a participant that has a large WM capacity (greater than 4) would take some time before it converges to her true WM span. Thus, it would have been more precise to compute WM span estimates only after convergence of the double-staircase (i.e., after the intersection of the two staircases). Here, we were restricted by the fact that, for some subjects, the staircases did not cross each other until very late, making it impossible to compute their WM span in the first block.

Another limitation of our analysis of the evolution of measured WM span over time is that we only took into account the first and last block in each task. A more comprehensive analysis would need to include all blocks in the ANOVA analysis (although it will not be straightforward since the Markovian task contained 4 blocks whereas there were 8 blocks in the non-Markovian task).

Future analyses should also examine how WM span varied in different learning phases. We would expect WM span to be lower before time-to-criterion than after reaching it. Change-point analysis could possibly be used to shed light on this question.

Finally, most of the existing studies about the WM-based RL framework has solely used modelling. To the best of our knowledge, the only work that included human experiments, and which inspired the present study, was done by Krueger (2011). We hope that the methodology used in the present study would inspire new experimental works looking at the interaction between WM and RL. Several open questions still need to be answered, among them: Could we predict learning performance just by looking at how WM resources were allocated during the learning task? Would existing models fit people's performance well? If we do not give people the exact number of cues to keep track of, would people allocate their WM resources flexibly?

Chapter 7

General discussion

This thesis was initially motivated by the lack of experimental studies investigating human reinforcement learning (RL) in partially observable tasks in comparison with studies that have looked at the problem using a purely modelling approach. We considered two types of partial observability: (1) that which is due to state uncertainty and (2) that which is due to a dependency of some states on past events. The presence of partial observability makes learning problems more difficult to solve for both humans and artificial machines. Modelling studies often report extremely slow learning results, raising doubts about the practicality of experimentally testing partially observable tasks on humans, especially those with memory dependency.

The present work made important contributions to improve our understanding of why these models are often slow to learn in memory-dependent partially observable domains, and how these can be improved. The thesis made also contributions in the experimental side by proposing two studies that investigated human RL in each of the two cases of partial observability considered here. This chapter summarises these contributions, identifies the implications and limitations of our work, and presents ideas for further research.

7.1 Summary of modelling work

One of our main contributions in this thesis is the introduction of a number of modifications to the Actor-Critic gating model of Todd et al. (2009)—an influential model for solving memory-dependent partially observable tasks—which made the model learn substantially faster and enhanced its plausibility as a model of human learning behaviour. Most of the suggested modifications can also be applied to other working memory-based RL models that share some features with the Actor-Critic gating model (e.g., the PBWM model of O'Reilly and Frank, 2006). These modifications were mostly based on our work in Chapter 3 in which we identified there major problems with the model of Todd et al. (2009).

The first problem arose as we were attempting to replicate Todd et al.'s (2009) finding that their Actor-Critic gating model can successfully learn the 12-AX task. We found that the model could not completely learn the rules of the task although we used the same parameters that were reported in Todd et al.'s (2009) paper. Our analyses revealed that this was due to the multiple gating agents acting independently from each other, and thus ending up acting redundantly (i.e., storing the same information in different working memory cells). To solve this problem, we proposed that a single gating agent controls all working memory (WM) cells, instead of having multiple gating agents, each controlling one cell. This modification not only solved the redundancy problem, but also allowed the model to learn faster since it reduced the number of possible gating actions (from $2C$ to $C+1$ actions with a WM capacity of C; see Chapter 3).

It might be argued that the redundancy problem happened simply because of the nature of the 12-AX task, and might not be a cause of concern in other WM learning tasks. In fact, a high asymptotic accuracy could be obtained in the 12-AX using just one WM cell (about 95%). This might explain why the original Actor-

Critic gating model tried to mimic the one-cell version by using its two WM cells as if it only had one WM cell.

Regardless of whether or not the redundancy problem could arise in other WM learning tasks, it seems more plausible to consider a single gating agent controlling all WM cells than to assume that there are multiple gating agents that can modify WM simultaneously and independently from each other. There is a growing body of evidence pointing towards the existence of a system—namely the focus of attention—that controls WM memory in a serial fashion (Garavan, 1998; Oberauer, 2002). This system is limited in the sense that it can attend to only one item in WM at a given time. For example, Garavan (1998) showed that people could not keep track of two running counts in WM with equal speed. More precisely, he found that there was a time cost when participants switched from a running count to the other one, in comparison with when they had to update the same running count twice in a row. Similarly, Kessler and Meiran (2008) gave participants a WM updating task where they manipulated the number of presented items that must be replaced in WM with a different item, and found that the time required to carry out WM updating operations increased in general with the number of items to replace. These findings clearly go against a WM system that can simultaneously and independently update all stored items.

The second issue that we identified is due to a rigid assumption that is used in most existing WM-based RL models, which is that WM capacity remains fixed over the course of the task at hand, and that the learner must choose the WM capacity to allocate to the task before embarking on it. This is not only an implausible assumption, but also does not deal well with the trade-off between learning speed and accuracy that arises in some complex WM learning tasks (Chapter 3). In fact, while a large WM capacity might maximise long-term performance, it often slows down learning by increasing the dimensionality of the problem to solve. In contrast, allocating a small WM capacity might speed up learning, but can also limit the final accuracy that the learner can achieve.

Instead of learning with a constant WM capacity throughout a learning task, we proposed an extension to the Actor-Critic gating model in which the learner is assumed to gradually increase its allocated WM capacity in response to its current accuracy level (Chapter 4). The resulting model—the flexible resource allocation—showed significant improvements in learning speed on the 12-AX in comparison with the one-gating Actor-Critic gating model (i.e., our RLWM model), confirming the benefit of the starting small approach when learning complex tasks (Elman, 1993; Krueger and Dayan, 2009). Another advantage of this extension is that it allows the model to be smoothly applied to Markovian tasks as well, thus adding more generality to it. In fact, if the task at hand is solvable using a memoryless policy (i.e., does not need memory to be solved), the flexible resource allocation model would potentially learn it in the same manner as a Markovian RL model during the initial stage of resource allocation.

Our work on the gradual allocation approach inspired us to introduce the Bayesian approach to the study of WM-based RL (Chapter 5). The main idea behind introducing the Bayesian approach was to enhance the adaptive resource allocation model (though we did not get the chance to construct a Bayesian resource allocation model in this thesis) by taking advantage of the ability of Bayesian methods to automatically tune the exploration and learning parameters, and to propose a new criterion for deciding when to increase WM resources based on the precision of the Q-value estimates rather than accuracy. The Bayesian approach has other advantages that go beyond just improving the flexible resource allocation model. One of the advantages is to reduce the number of free parameters to set. In fact, the Bayesian model that we introduced has only two free parameters (the discount parameter γ and the variance of the white noise σ^2); the learning rate, the exploration rate and eligibility decay parameter being all replaced by native learning mechanisms for value updating and action-selection that do not necessitate those parameters.

More importantly, as our simulation results show, the Bayesian WM-based RL model that we introduced learns at comparable or at faster levels than the non-Bayesian version. For example, in the simple 12-XY task, the model reached an accuracy level of 95% within less than 800 trials, and thus can be potentially tested against human performance in a potential future experiment based on the 12-XY task. This new experiment, as opposed to the experiment that we reported in Chapter 6, would use the full version of the 12-XY (which is more challenging to learn), and would not include the extra WM span task to measure WM capacity.

The third major issue that we identified concerns the non-coordination between the gating and motor agent. As we demonstrated in Section 3.3.3, resolving this issue could potentially improve the learning speed of the Actor-Critic gating model by up to seven times in the 12-AX task. Although we did not treat in depth this area of possible improvement, we proposed several ways that could help overcome this challenge. For example, one could implement a similar solution to the one we used to resolve the redundancy problem—that is, use a single agent that simultaneously takes both gating and motor actions. This is somewhat similar to the assumption used by Zilli and Hasselmo (2008) in their gating model; however, they only allowed the agent to take either a gating or a motor action at a time. Another possible solution, which was proposed by Lanzi (2000) as a way to improve performance in WM-based RL models (although it was not intended to directly address the coordination problem), is to reduce the exploration associated with the gating agent in comparison with that of the motor agent. This should provide a more stable WM context for the motor agent to carry out the learning of the motor policy. However, this might also create an under-exploration problem, especially in high dimensional problems.

While working on the aforementioned areas of improvement in the Actor-Critic gating model, we found other interesting ways to enhance the model. For example, the use of Dabney and Barto's (2012) adaptive learning rate instead of a fixed learning rate—which we initially introduced in the context of the resource allocation model—substantially speeded up learning with the Actor-Critic gating model (Chapter 4). This was surprising as we did not expect this simple manipulation to approximately halve the learning time of the model in the 12-AX. A similar manipulation that we introduced was the use of a decreasing exploration rate, which improved the learning performance of the flexible resource allocation model (although we did not implement it under the Actor-Critic gating). This shows that careful selection of parameters plays an important role in WM-based RL models, and thus methods that automatise the parameter selection process could be beneficial. The Bayesian approach is very useful in that sense, since it has the advantage, among others, of removing the need to use parametrised action selection and eligibility trace functions, as well as the learning rate parameter.

Finally, an important finding that it is worth mentioning here concerns the 12-AX task. This task has been used as a benchmark in many studies evaluating WM-based RL models (Frank et al., 2001; O'Reilly and Frank, 2006; Todd et al., 2009; Krueger and Dayan, 2009; Raizer and Gudwin, 2015), and is often presented as a task that requires the learner to independently keep track of two set of cues (the digits 1 and 2, and the letters A, B and C). It is true that this is a requirement to perfectly learn the 12-AX, but as we demonstrated, a very high performance level can be achieved by using only one WM cell (see Chapter 3). An implication of this finding, is that one must be cautious when using the 12-AX to evaluate learning models. In particular, time-to-criterion measures that are based on accuracy rates in a succession of trials can be misleading. To illustrate this, we tested if the RLWM model (i.e., our modified version of the Actor-Critic gating model; see Chapter 3) with one WM cell can reach the criterion used in Todd et al. (2009) and another seminal paper by O'Reilly and Frank (2006) (i.e., 300 successive correct responses) on the 12-AX. We found that in 30 simulation runs, the model could reach this criterion within 10^6 trials in 4 simulation runs. Thus, there is a risk of reporting a model as having completely learned the 12-AX, when in fact it has only converged to a suboptimal policy.

In summary, the present thesis introduced a number of modifications to the Actor-Critic gating model of Todd et al. (2009), which made it more plausible as a model of human learning, and substantially improved its learning speed. However, these modifications are still far from making the model perform at comparable levels to humans in complex learning tasks such as the 12-AX. Certainly, we can further reduce the model learning time by pursuing other avenues of possible improvements, such as structuring the learning between the motor and gating agent; however, it seems very unlikely that we could reach a level where the 12-AX, for example, can be fully learned within 1000 trials under the RL framework, as can be done by human participants (Krueger, 2011). Therefore, there is a need to think about alternative approaches to account for human learning in complex WM learning tasks. One possible direction would be to construct heuristic models, as we shall discuss in Section 7.3.

7.2 Summary of experimental work

We conducted two behavioural experiments to study people's learning behaviour in partially observable environments with either state uncertainty or memory dependency: the first, a simple psychophysical experiment, which tested how people perform RL when the environment provides ambiguous inputs, and to evaluate the impact of an unsignalled reversal of stimulus-response contingencies on their learning performance. The second experiment aimed at evaluating the involvement of WM in human RL in memory-dependent partially observable tasks (i.e., non-Markovian tasks).

In the first experiment (Chapter 2), participants did not have difficulty to learn under state uncertainty conditions. Most of them also managed to detect the unsignalled switch, supporting findings from previous studies that looked at expected and unexpected uncertainty when the source of stochasticity is the reward function, rather than the state as in our study (see for example, Behrens et al., 2007; Nassar et al., 2010).

Model fitting results suggest that most participants were well described by a simple temporal difference-based RL model, in which the learner is assumed to ignore state uncertainty and simply update the Q-value of the state that she identifies as the true state (Larsen et al., 2010). This included participants from the control condition, where there was only expected state uncertainty, and participants who adapted to the switch in the test condition. Participants who did not adapt or adapted slowly to the switch were better accounted by a Bayesian RL that uses reward feedback to inform beliefs about the true state, which are then used to update the state-action values (Larsen et al., 2010).

These results suggest that people could successfully learn partially observable tasks with state uncertainty, and that their behaviour could be well described in terms of two RL mechanisms. However, the task that we used in this first experiment was very simple involving only two states and two response options. Hence, it remains to be seen if people can still solve this type of tasks when these are more challenging, and whether their learning can still be accounted for using RL models.

The second experiment (Chapter 6) tested a core assumption behind WM-based RL models, which is that WM supports RL in non Markovian domains by keeping track of previous events and allowing them to take part of the learning context. For that, we compared participants' available WM span in two conditions: while they were learning a non-Markovian task (the episodic 12-XY) and while they were learning a Markovian version of the same task. We found that participants' measured WM span were lower in the non-Markovian condition in comparison with the Markovian condition, suggesting that participants used extra WM resources to support RL in the non-Markovian condition.

Surprisingly, only about 60% of the participants completely learned the rules of the non-Markovian task, although the task was very basic in comparison with the tasks that have been used in the WM-based RL literature. This is far from the percentage of participants who managed to learn the 12-AX in Krueger's (2011) experiment, which was greater than 80%. However, participants in our case had to also perform the WM span task, which certainly made their overall task more challenging.

To sum up our experimental work, we demonstrated that our participants were, overall, capable of successfully learning under partial observability conditions either because of state uncertainty (even when we introduced unexpected uncertainty) or due to some states dependent on the past, so some sort of memory is needed (participants had to also perform a WM span task at the same time). Participants' choices in the first scenario were well accounted by two RL models: a simple model that explained most participants data and a Bayesian model that accounted mainly for those participants who did not adapt or adapted slowly to the unexpected uncertainty. The main purpose of the second experiment was not to measure or explain participants behaviour, but to simply test whether or not WM is used to keep track of past events in non-Markovian tasks. One could use the same learning task (without the WM span task) to measure human performance and compare it against the performance of existent WM-based RL models.

7.3 Future directions

The present thesis identified three major shortcomings of the Actor-Critic gating model of Todd et al. (2009). We successfully addressed two of them—one related to the use of multiple independent gating agents and the second due the use of a fixed WM capacity—and suggested possible ways to solve the third problem, which concerns the question of how to best coordinate the behaviour of the gating and motor agent. More specifically, the third issue arises as a result of simultaneously updating both gating and motor state-action values, which might lead the model to wrongly modify a correct motor strategy after receiving a negative reward feedback, when the problem is due to an incorrect gating response and vice versa. One of our suggestions to overcome this issue was to introduce a criterion for detecting which agent—gating, motor or both—is responsible for the observed reward, then allocate the reward to each agent accordingly. However, it is not obvious how one could define this criterion.

One possibility to explore would be to construct the criterion using a Bayesian approach that is similar to the one used in the PWRL model (i.e., the Bayesian RL model of Larsen et al., 2010, which can deal with state uncertainty). More specifically, we would compute the posterior probability—after receiving the reward—that the chosen gating action is correct and the posterior probability that the chosen motor action is correct. Once this is done (we leave the details of how to compute the posterior probabilities for future work), the Q-values of each agent would be updated according to the computed posterior probabilities as in the PWRL model.

Another possibility is to use a heuristic approach based on hypothesis testing. Basically, one could structure learning by updating only the motor policy each time a reward is observed, while the gating policy is kept constant and assumed to be correct. The gating policy is updated only when an inconsistency happens while learning the motor policy, suggesting that there is a problem with the gating policy instead of the motor policy. This approach would work nicely in tasks that have deterministic rewards, such as the 12-AX, in which case rewards give information about the correctness of the selected response. The inconsistency that triggers the updating of the gating policy could arise when a previously positively rewarded motor response is not rewarded again given the same observation from the environment and the same WM contents. We have started implementing this idea and have obtained very promising results on the 12-AX (learning could be completed in less than five thousand trials), but we still need to work out how we can generalise the method so that it can also work with stochastic reward functions.

One limitation of our modelling work is that we applied our models exclusively to the 12-AX task or some variant of it. It would be interesting to see if the proposed models can solve other non-Markovian tasks, which are different in nature from the 12-AX—preferably tasks that have already been tested on humans (see for example, Hampton et al., 2006; Clarke et al., 2015; for a review, see Walsh and Anderson, 2014). This would provide us with a baseline learning performance to compare it to the performance of our WM-based RL models. It would also allow us to see if the proposed models can explain some features and limitations in human learning in partially observable domains. Appendix E provides a list of potential tasks that we selected from previous learning studies that tested human or animal subjects (although all but one task have been used in a different context than studying WM-based RL).

Nevertheless, our modelling work has generated a number of predictions that can be tested in the lab. For instance, the resource allocation model predicts that people would gradually allocate their working memory resources if given a non-Markovian RL task. We started to test this prediction in our second experiment (Chapter 6), but we cannot draw a final conclusion from this experiment, since we prompted the subjects to keep track of one cue at a time by informing them that the correct responses to the targets X and Y depended on exactly one cue. To effectively test the gradual allocation hypothesis, the same experimental design could be used with the exception that participants should not be given any instruction about the number of items they need to keep track of at any given time.

One would expect three possible outcomes from such experiment: (1) the allocated WM would remain, in general, constant throughout the non-Markovian task relative to the Markovian task as we found in our experiment. This would give support to the assumption that learning is performed with fixed WM capacity as assumed by most WM-based RL models; (2) participants' allocated WM resources would increase in general as assumed by the resource allocation model; or (3) the opposite pattern would arise, i.e. the allocated WM would decrease over time. This might suggest that people use a different mechanism than RL, such as hypothesis testing. In fact, a process like hypothesis testing would require participants to use more WM resources in early stages of learning to generate hypotheses and hold them in mind while testing them, then as learning progresses, WM would be freed as less rules to learn would remain.

Even if such experiment does not provide evidence for gradual allocation of WM resources in human learning, we can still test if inducing people to use the gradual resource allocation strategy can improve their learning speed as with the gating model. This can be done, for example, by loading participants' WM using a WM span task while they are performing a WM learning task (the function of the WM span task in this case will not be to measure their WM capacity but to consume a part of their WM resources), such that the amount of WM load is high at the initial stages of learning to force them to use low WM resources, then reducing the WM load in later learning stages to allow them to allocate more resources.

Concerning the two experiments that we presented in this thesis, both provided rich data about how people learn in partially observable tasks. We identified different types of learners in each case, suggesting that people might be using different approaches to learn under partial observability, instead of one particular learning mechanism. In particular, the different patterns of change-point detection in the first experiment were interesting (some participants adapted quickly to the reversal and some took more time, while other participants did not adapt to the reversal at all), but we could not make strong statistical inferences from them given the small number of participants in each learner category.

Similarly, in the other experiment about WM-based RL, we identified three categories of learners: those who fully learned the non-Markovian task (full-learners), those who performed at chance level throughout the task (non-learners) and those who partially learned the task (half-learners); however, again we could not make the most of this variability, especially we could not use the group of half-learners in our analyses since it contained only three participants. Including this group could potentially help us answer some interesting questions such as: what is preventing half-learners from completely learning the task? Are they slow to learn the rules for easy sequences or do they do it quickly but then struggle with the hard sequence? Do they learn similarly to participants who fully but slowly learn the task during the initial stages of learning?

7.4 Conclusions

This thesis examined how people learn through reward in partially observable environments where some states are either uncertain or dependent on the past. We showed that people are generally capable of learning under both these types of partial observability when the tasks involved are simple. In addition, the behaviour of participants, in the case of state uncertainty, was well explained by two temporal difference-based RL models.

In order to facilitate a potential use of WM-based RL models to explain people's learning behaviour in memory-dependent partially observable tasks, we worked toward a modified version of the Actor-Critic gating model of Todd et al. (2009) that can solve these tasks as fast as humans. Although we did not completely reach this target, our studies provide important insights into why the Actor-Critic gating model—and WM-based RL in general–are often slow to learn. We highlighted, for the first time, problems with some of the core assumptions in these models, and proposed a number of modifications to address those issues, which resulted in a substantial increase in learning speed and made the Actor-Critic gating model more plausible. We also introduced a Bayesian WM-based RL model and showed that it outperformed our improved version of the Actor-Critic gating model on a simple WM learning task, while not requiring a lot of free parameters to set. This is the first time that a Bayesian approach has been used in the study of WM-Based RL.

Partial observability presents a great challenge to the applicability of RL methods as potential mechanisms for explaining human learning behaviour and as tools for machine learning in applied settings. There is still a long way to go before we can claim that people use an RL process or another learning mechanism when dealing with partial observable environments. In this thesis, we took the initial step of trying to improve existing RL models to make them faster and more plausible. It would be interesting in the future to explore other learning approaches to see if they learn faster than RL models and whether or not they can better explain human behaviour.

Bibliography

Andreae, J. H. (1969). Learning machines-a unified view. In Meetham, A. R. and Hudson, R. A., editors, *Encyclopedia of Information, Linguistics, and Control*, pages 261–270. Pergamon, Oxford.

Ashby, F. G. and Spiering, B. J. (2004). The neurobiology of category learning. *Behavioral and cognitive neuroscience reviews*, 3(2):101–113.

Aström, K. (1965). Optimal control of markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10(1):174 – 205.

Atkinson, R. C. and Shiffrin, R. M. (1971). The control of short-term memory. *Scientific American*, 225(2):82–90.

Baddeley, A. D. (2012). Working memory: Theories, models, and controversies. *Annual Review of Psychology*, 63:1–29.

Baddeley, A. D., Thomson, N., and Buchanan, M. (1975). Word length and the structure of short-term memory. *Journal of verbal learning and verbal behavior*, 14(6):575–589.

Badre, D. (2012). Opening the gate to working memory. *Proceedings of the National Academy of Sciences*, 109(49):19878–19879.

Balleine, B. W. and O'Doherty, J. P. (2010). Human and rodent homologies in action control: corticostriatal determinants of goal-directed and habitual action. *Neuropsychopharmacology*, 35(1):48–69.

Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846.

Bayer, H. M., Lau, B., and Glimcher, P. W. (2007). Statistics of midbrain dopamine neuron spike trains in the awake primate. *Journal of Neurophysiology*, 98(3):1428–1439.

Bays, P. M., Gorgoraptis, N., Wee, N., Marshall, L., and Husain, M. (2011). Temporal dynamics of encoding, storage, and reallocation of visual working memory. *Journal of vision*, 11(10):6–6.

Behrens, T. E., Woolrich, M. W., Walton, M. E., and Rushworth, M. F. (2007). Learning the value of information in an uncertain world. *Nature neuroscience*, 10(9):1214–1221.

Bellman, R. (1957). *Dynamic programming.* Princeton University press, Princeton.

Bland, A. R. and Schaefer, A. (2012). Different varieties of uncertainty in human decision-making. *Frontiers in neuroscience*, 6(85).

Bogacz, R., Brown, E., Moehlis, J., Holmes, P., and Cohen, J. D. (2006). The physics of optimal decision making: a formal analysis of models of performance in two-alternative forced-choice tasks. *Psychological review*, 113(4):700.

Boyd, R., Richerson, P. J., and Henrich, J. (2011). Colloquium paper: The cultural niche: Why social learning is essential for human adaptation. In *Proceedings of the National Academy of Sciences*, pages 10918–10925. 108 (Supplement 2).

Brainard, D. H. (1997). The psychophysics toolbox. *Spatial Vision*, 10(4):433–436.

Braver, T. S. and Cohen, J. D. (2000). On the control of control: The role of dopamine in regulating prefrontal function and working memory. In Monsell, S. and Driver, J. S., editors, *Control of Cognitive Processes: Attention and Performance XVIII*, pages 713–737. MIT Press, Cambridge, MA.

Braver, T. S. and Cohen, J. D. (2001). Working memory, cognitive control, and the prefrontal cortex: Computational and empirical studies. *Cognitive Processing*, 2(1):2555.

Bush, R. R. and Mosteller, F. (1951). A mathematical model for simple learning. *Psychological Review*, 58(5):313–323.

Busoniu, L., Babuska, R., and De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(2):156–172.

Case, R., Kurland, D. M., and Goldberg, J. (1982). Operational efficiency and the growth of short-term memory span. *Journal of Experimental Child Psychology*, 33(3):386–404.

Chatham, C. H. and Badre, D. (2013). Working memory management and predicted utility. *Frontiers in behavioral neuroscience*, 7(83):1–12.

Chatham, C. H. and Badre, D. (2015). Multiple gates on working memory. *Current opinion in behavioral sciences*, 1:23–31.

Chi, M., VanLehn, K., and Litman, D. (2010). Do micro-level tutorial decisions matter: Applying reinforcement learning to induce pedagogical tutorial tactics. In *Intelligent Tutoring Systems*, pages 224–234. Springer.

Clark, H. H. and Clark, E. V. (1977). *Psychology and language: An introduction to psycholinguistics*. Harcourt Brace Jovanovich, New York.

Clarke, A. M., Friedrich, J., Tartaglia, E. M., Marchesotti, S., Senn, W., and Herzog, M. H. (2015). Human and Machine Learning in Non-Markovian Decision Making. *Plos One*, 10(4):e0123105.

Cohen, J., Barch, D., Carter, C., and Servan-Schreiber, D. (1999a). Schizophrenic deficits in the processing of context: converging evidence from three theoretically motivated cognitive tasks. *Journal of Abnormal Psychology*, 108(1):120–133.

Cohen, J. D., Barch, D. M., Carter, C., and Servan-Schreiber, D. (1999b). Context-processing deficits in schizophrenia: converging evidence from three theoretically motivated cognitive tasks. *Journal of abnormal psychology*, 108(1):120.

Collins, A. G., Brown, J. K., Gold, J. M., Waltz, J. A., and Frank, M. J. (2014). Working memory contributions to reinforcement learning impairments in schizophrenia. *The Journal of Neuroscience*, 34(41):13747–13756.

Collins, A. G. and Frank, M. J. (2012). How much of reinforcement learning is working memory, not reinforcement learning? a behavioral, computational, and neurogenetic analysis. *European Journal of Neuroscience*, 35(7):1024–1035.

Conway, A. R. A., Kane, M. J., Bunting, M. F., Hambrick, D. Z., Wilhelm, O., and Engle, R. W. (2005). Working memory span tasks: A methodological review and user's guide. *Psychonomic Bulletin and Review*, 12(5):769–786.

Cornsweet, T. N. (1962). The Staircase-Method in Psychophysics. *The American Journal of Sports Medicine*, 75(3):485–491.

Cowan, N. (2001). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, 24(1):87–185.

Cowan, N. (2005). *Working memory capacity.* Psychology Press, Hove, East Sussex, UK.

Dabney, W. and Barto, A. (2012). Adaptive step-size for online temporal difference learning. In *Twenty-sixth AAAI conference on artificial intelligence.*

Dahlin, E., Neely, A. S., Larsson, A., Bäckman, L., and Nyberg, L. (2008). Transfer of learning after updating training mediated by the striatum. *Science*, 320(5882):1510–1512.

Daneman, M. and Carpenter, P. A. (1980). Individual differences in working memory and reading. *Journal of Verbal Learning and Verbal Behavior*, 19(4):450–466.

Daw, N. D. (2011). Trial-by-trial data analysis using computational models. *Decision making, affect, and learning: Attention and performance XXIII*, 23:3–38.

Daw, N. D., Niv, Y., and Dayan, P. (2005). Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature neuroscience*, 8(12):1704–1711.

DeCaro, M. S., Thomas, R. D., and Beilock, S. L. (2008). Individual differences in category learning: Sometimes less working memory capacity is better than more. *Cognition*, 107(1):284–294.

D'Errico, J. (2005). fminsearchbnd an extension to fminsearch. http://www.mathworks.com/matlabcentral/fileexchange/8277-fminsearchbnd–fminsearchcon. Latest update 2012.

Dirlam, D. K. (1972). Most efficient chunk sizes. *Cognitive Psychology*, 3:355–359.

Doll, B. B., Hutchison, K. E., and Frank, M. J. (2011). Dopaminergic genes predict individual differences in susceptibility to confirmation bias. *The Journal of neuroscience*, 31(16):6188–6198.

Doshi-Velez, F. and Ghahramani, Z. (2011). A comparison of human and agent reinforcement learning in partially observable domains. In *Proceeding of the 33rd Annual Meeting of Cognitive Science Society*.

Elman, J. L. (1993). Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99.

Engel, Y., Mannor, S., and Meir, R. (2003). Bayes meets bellman: The gaussian process approach to temporal learning. In *Machine Learning, Proceedings of the Twentieth International Conference = (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 154–161.

Engel, Y., Mannor, S., and Meir, R. (2005). Reinforcement learning with Gaussian processes. *Proceedings of the 22nd international conference on Machine learning - ICML '05*, pages 201–208.

Engle, R. W. (2002). Working memory capacity as executive attention. *Current Directions in Psychological Science*, 11(1):19–23.

Ez-zizi, A., Farrell, S., and Leslie, D. (2015). Bayesian reinforcement learning in markovian and non-markovian tasks. In *IEEE Symposium Series on Computational Intelligence*, pages 579–586.

Farley, B. G. and Clark, W. A. (1954). Simulation of self-organizing systems by digital computer. *IRE Transactions on Information Theory*, 4(4):76–84.

Frank, M. J., Loughry, B., and O'Reilly, R. C. (2001). Interactions between the frontal cortex and basal ganglia in working memory: A computational model. *Cognitive, Affective, & Behavioral Neuroscience*, 1(2):137–160.

Garavan, H. (1998). Serial attention within working memory. *Memory & cognition*, 26(2):263–276.

Geist, M. and Pietquin, O. (2010). Kalman temporal differences. *Journal of Artificial Intelligence Research*, 39:483–532.

Geist, M. and Pietquin, O. (2011). Kalman filtering & colored noises: the (autoregressive) moving-average case. *Workshop Proceedings of ICMLA 2011*, (9).

Geramifard, A., Bowling, M., and Sutton, R. S. (2006). Incremental least-squares temporal difference learning. In *Proceedings of the National Conference on Artificial Intelligence*, volume 1, pages 356–361.

Gers, F. A., Schmidhuber, J., and Cummins, F. (2000). Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471.

Gobet, F. and Clarkson, G. (2004). Chunks in expert memory: Evidence for the magical number four ... or is it two? *Memory*, 12(6):732–747.

Gorgoraptis, N., Catalao, R. F., Bays, P. M., and Husain, M. (2011). Dynamic updating of working memory resources for visual objects. *The Journal of Neuroscience*, 31(23):8502–8511.

Gorski, N. A. (2012). *Learning to Use Memory.* PhD thesis, The University of Michigan.

Hampton, A. N., Bossaerts, P., and O'Doherty, J. P. (2006). The Role of the Ventromedial Prefrontal Cortex in Abstract State-Based Inference during Decision Making in Humans. *Journal of Neuroscience*, 26(32):8360–8367.

Hasinoff, S. W. (2002). Reinforcement learning for problems with hidden state. *University of Toronto, Technical Report.*

Hertwig, R. and Todd, P. M. (2003). *More Is Not Always Better: The Benefits of Cognitive Limits*, chapter 11, pages 213–232. John Wiley & Sons, Ltd.

Hochman, G. and Erev, I. (2013). The partial-reinforcement extinction effect and the contingent-sampling hypothesis. *Psychonomic bulletin & review*, 20(6):1336–1342.

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

Hollerman, J. R. and Schultz, W. (1998). Dopamine neurons report an error in the temporal prediction of reward during learning. *Nature neuroscience*, 1(4):304–309.

Howard, R. A. (1960). *Dynamic Programming and Markov Processes.* MIT press, Cambridge, MA.

Humphreys, L. G. (1939). The effect of random alternation of reinforcement on the acquisition and extinction of conditioned eyelid reactions. *Journal of Experimental Psychology*, 25(2):141.

Joel, D., Niv, Y., and Ruppin, E. (2002). Actor–critic models of the basal ganglia: New anatomical and computational perspectives. *Neural networks*, 15(4):535–547.

Jonides, J., Schumacher, E. H., Smith, E. E., Lauber, E. J., Awh, E., Minoshima, S., and Koeppe, R. A. (1997). Verbal working memory load affects regional brain activation as measured by PET. *Journal of Cognitive Neuroscience*, 9(4):462–475.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Artificial intelligence planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

Kaernbach, C. (1991). Simple adaptive testing with the weighted up-down method. *Attention, Perception, & Psychophysics*, 49(3):227–229.

Kamin, L. J. (1969). Predictability, surprise, attention, and conditioning. *Punishment and aversive behavior*, pages 279–296.

Kareev, Y. (2000). Seven (indeed, plus or minus two) and the detection of correlations. *Psychological review*, 107(2):397.

Kareev, Y., Lieberman, I., and Lev, M. (1997). Through a narrow window: Sample size and the perception of correlation. *Journal of Experimental Psychology: General*, 126(3):278.

Kessler, Y. and Meiran, N. (2008). Two dissociable updating processes in working memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 34(6):1339.

Kim, H. J., Jordan, M. I., Sastry, S., and Ng, A. Y. (2004). Autonomous helicopter flight via reinforcement learning. In *Advances in Neural Information Processing Systems 16*, pages 799–806. MIT Press.

Kimberg, D. Y., D'Esposito, M., and Farah, M. J. (1997). Effects of bromocriptine on human subjects depend on working memory capacity. *Neuroreport*, 8(16):3581–3585.

Kirchner, W. K. (1958). Age differences in short-term retention of rapidly changing information. *Journal of Experimental Psychology*, 55(4):352–358.

Kleiner, M., Brainard, D., and Pelli, D. (2007). What's new in Psychtoolbox-3? *Perception*, 36S.

Kober, J. and Peters, J. (2013). Reinforcement learning in robotics: A survey. In *Wiering, M., & Otterlo, M. (Eds.)*, volume 12, pages 579–610. Springer.

Krueger, K. A. (2011). *Sequential learning in the form of shaping as a source of cognitive flexibility*. PhD thesis, University College London.

Krueger, K. A. and Dayan, P. (2009). Flexible shaping: How learning in small steps helps. *Cognition*, 110(3):380–394.

Kuleshov, V. and Precup, D. (2000). Algorithms for the multi-armed bandit problem. *Journal of Machine Learning*, 1:1–48.

Lanzi, P. L. (1998). Adding memory to XCS. In *IEEE International Conference on Computational Intelligence*, pages 609–614.

Lanzi, P. L. (2000). Adaptive agents with reinforcement learning and internal memory. In *The Sixth International Conference on the Simulation of Adaptive Behavior (SAB2000)*.

Larsen, T., Leslie, D., Collins, E., and Bogacz, R. (2010). Posterior weighted reinforcement learning with state uncertainty. *Neural Computation*, 22(5):1149–1179.

Le Pelley, M. E. (2004). The role of associative history in models of associative learning: A selective review and a hybrid model. *Quarterly Journal of Experimental Psychology Section B*, 57(3):193–243.

Levitt, H. (1971). Transformed up-down methods in psychoacoustics. *The Journal of the Acoustical society of America*, 49(2B):467–477.

Lipton, Z. C. (2015). A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*.

Littman, M. L. (1994). Memoryless policies : theoretical limitations and practical results. In *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pages 238–245. MIT press.

Littman, M. L. (2009). A tutorial on partially observable Markov decision processes. *Journal of Mathematical Psychology*, 53(3):119–125.

Lloyd, K., Becker, N., Jones, M. W., and Bogacz, R. (2012). Learning to use working memory: a reinforcement learning gating model of rule acquisition in rats. *Frontiers in Computational Neuroscience*, 6(87).

Loch, J. and Singh, S. (1998). Using eligibility traces to find the best memoryless policy in partially observable Markov decision processes. In *the proceedings of the fifteenth Internationl Conference on Machine Learning*, pages 323–331, San Francisco, CA. Morgan Kaufmann.

Ma, W. J., Husain, M., and Bays, P. M. (2014). Changing concepts of working memory. *Nature neuroscience*, 17(3):347–356.

MacGregor, J. N. (1987). Short-term memory capacity: Limitation or optimization? *Psychological Review*, 94:107–108.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. *5th Berkeley Symposium on Mathematical Statistics and Probability 1967*, 1(233):281–297.

May, B. C., Korda, N., Lee, A., and Leslie, D. S. (2012). Optimistic bayesian sampling in contextual-bandit problems. *The Journal of Machine Learning Research*, 13:2069–2106.

McCallum, A. K. (1993). Overcoming incomplete perception with utile distinction memory. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 190–196. Morgan Kaufmann.

Michie, D. (1961). Trial and error. *Science Survey, Part*, 2:129–145.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63:81–97.

Minsky, M. L. (1954). *Theory of Neural-Analog Reinforcement Systems and its Application to the Brain-Model Problem.* PhD thesis, Princeton University.

Moriarty, D. E., Schultz, A. C., and Grefenstette, J. J. (1999). Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research*, 11:241–276.

Morris, G., Nevet, A., Arkadir, D., Vaadia, E., and Bergman, H. (2006). Midbrain dopamine neurons encode decisions for future action. *Nature neuroscience*, 9(8):1057–1063.

Nassar, M. R., Wilson, R. C., Heasly, B., and Gold, J. I. (2010). An approximately bayesian delta-rule model explains the dynamics of belief updating in a changing environment. *The Journal of Neuroscience*, 30(37):12366–12378.

Nevin, J. A. and Grace, R. C. (2000). Behavioral momentum: Empirical, theoretical, and metaphorical issues. *Behavioral and Brain Sciences*, 23:117–125.

Niv, Y. (2009). Reinforcement learning in the brain. *Journal of Mathematical Psychology*, 53(3):139–154.

Oberauer, K. (2002). Access to information in working memory: Exploring the focus of attention. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 28(3):411–421.

Olton, D. S. and Samuelson, R. J. (1976). Remembrance of places passed: spatial memory in rats. *Journal of Experimental Psychology: Animal Behavior Processes*, 2(2):97–116.

O'Reilly, R. C. and Frank, M. J. (2006). Making working memory work: A computational model of learning in the prefrontal cortex and basal ganglia. *Neural Computation*, 18(2):283–328.

Otto, A. R., Gershman, S. J., Markman, A. B., and Daw, N. D. (2013a). The curse of planning dissecting multiple reinforcement-learning systems by taxing the central executive. *Psychological science*, 24(5):751–761.

Otto, A. R., Raio, C. M., Chiang, A., Phelps, E. A., and Daw, N. D. (2013b). Working-memory capacity protects model-based learning from stress. *Proceedings of the National Academy of Sciences*, 110(52):20941–20946.

Packard, M. G. and Knowlton, B. J. (2002). Learning and memory functions of the basal ganglia. *Annual review of neuroscience*, 25(1):563–593.

Pavlov, I. P. (1927). *Conditioned reflexes: An Investigation of the physiological activity of the cerebral cortex.* Oxford University Press, London.

Pelli, D. G. (1997). The videotoolbox software for visual psychophysics: transforming numbers into movies. *Spatial Vision*, 10(4):437–442.

Peshkin, L., Meuleau, N., and Kaelbling, L. (1999). Learning policies with external memory. *Machine Learning: Proceedings of the Sixteenth International Conference*, pages 307–314.

Pollack, I., Johnson, L. B., and Knaff, P. R. (1959). Running memory span. *Journal of Experimental Psychology*, 57:137–146.

Puterman, M. L. and Shin, M. C. (1978). Modified policy iteration algorithms for discounted Markov decision problems. *Management Science*, 24(11):1127–1137.

Raizer, K. and Gudwin, R. R. (2015). A neuroscience inspired gated learning action selection mechanism. *Biologically Inspired Cognitive Architectures*, 11:65–74.

Rangel, A., Camerer, C., and Montague, P. R. (2008). A framework for studying the neurobiology of value-based decision making. *Nature Reviews Neuroscience*, 9(7):545–556.

Reitman, J. S. (1971). Mechanisms of forgetting in short-term memory. *Cognitive Psychology*, 2(2):185–195.

Rescorla, R. A. (1971). Variation in the effectiveness of reinforcement and non-reinforcement following prior inhibitory conditioning. *Learning and motivation*, 2(2):113–123.

Rescorla, R. A. and Wagner, A. R. (1972). A theory of pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In *Classical Conditioning II: Current Research and Theory (eds. A.H. Black & W.F. Prokasy)*, pages 64–99. Appleton-Century-Crofts.

Roesch, M. R., Calu, D. J., and Schoenbaum, G. (2007). Dopamine neurons encode the better option in rats deciding between differently delayed or sized rewards. *Nature neuroscience*, 10(12):1615–1624.

Rohrer, D., Pashler, H., and Etchegaray, J. (1998). When two memories can and cannot be retrieved concurrently. *Memory & cognition*, 26(4):731–739.

Rosvold, H. E., Mirsky, A. F., Sarason, I., Bransome Jr, E. D., and Beck, L. H. (1956). A continuous performance test of brain damage. *Journal of consulting psychology*, 20(5):343–350.

Rougier, A., Noelle, D., Braver, T., Cohen, J., and O'Reilly, R. (2005). Prefrontal cortex and flexible cognitive control: Rules without symbols. *Proceedings of the National Academy of Sciences*, 102(20):7338–7343.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

Rummery, G. A. and Niranjan, M. (1994). On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR166, Department of Engineering, University of Cambridge.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229.

Sawaguchi, T., Matsumura, M., and Kubota, K. (1990). Effects of dopamine antagonists on neuronal activity related to a delayed response task in monkey prefrontal cortex. *Journal of neurophysiology*, 63(6):1401–1412.

Schultz, W., Apicella, P., and Ljungberg, T. (1993). Responses of monkey dopamine neurons to reward and conditioned stimuli during successive steps of learning a delayed response task. *The Journal of Neuroscience*, 13(3):900–913.

Schultz, W., Dayan, P., and Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599.

Schwarz, G. et al. (1978). Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464.

Shadlen, M. N. and Newsome, W. T. (1996). Motion perception: seeing and deciding. *Proceedings of the National Academy of Sciences*, 93(2):628–633.

Shadlen, M. N. and Newsome, W. T. (2001). Neural basis of a perceptual decision in the parietal cortex (area lip) of the rhesus monkey. *Journal of neurophysiology*, 86(4):1916–1936.

Shiffrin, R. M. (1973). Information persistence in short-term memory. *Journal of Experimental Psychology*, 100(1):39–49.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Singh, S., Jaakkola, T., Littman, M. L., and Szepesvári, C. (2000). Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38(3):287–308.

Singh, S. P., Jaakkola, T., and Jordan, M. I. (1994). Learning without state-estimation in partially observable markovian decision processes. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 284–292. Morgan Kaufmann.

Singh, S. P. and Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine learning*, 22(1-3):123–158.

Skinner, B. F. (1935). Two types of conditioned reflex and a pseudo type. *The Journal of General Psychology*, 12(1):66–77.

Stankiewicz, B. J., McCabe, M., and Legge, G. E. (2004). Studying human spatial navigation processes using pomdps. In *AAAI workshop on learning and planning in Markov processes: Advances and challenges*, pages 97–102.

Sutton, R. and Barto, A. (1998). *Reinforcement learning: An introduction.* MIT Press, Cambridge, MA.

Sutton, R. S. (1984). *Temporal Credit Assignment in Reinforcement Learning.* PhD thesis, University of Massachusetts Amherst.

Sutton, R. S. (1990). Integrated architectures for learning, planning and reacting. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 216–224.

Sutton, R. S. and Barto, A. G. (1981). Toward a modern theory of adaptive networks: expectation and prediction. *Psychological review*, 88(2):135–170.

Swanson, H. L. and O'Connor, R. (2009). The role of working memory and fluency practice on reading comprehension of students who are dysfluent readers. *Journal of Learning Disabilities*, 42(6):548–575.

Tanaka, S. C., Shishida, K., Schweighofer, N., Okamoto, Y., Yamawaki, S., and Doya, K. (2009). Serotonin affects association of aversive outcomes to past actions. *Journal of neuroscience*, 29(50):15669–74.

Tesauro, G. (1992). Practical issues in temporal difference learning. *Machine Learning*, 8:257–277.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):pp. 285–294.

Thorndike, E. L. (1911). *Animal Intelligence.* Hafner, Darien, Conn.

Todd, M. T., Niv, Y., and Cohen, J. D. (2009). Learning to use working memory in partially observable environments through dopaminergic reinforcement. In *Advances in neural information processing systems*, pages 1689–1696.

Tomasello, M., Carpenter, M., Call, J., Behne, T., and Moll, H. (2005). Understanding and sharing intentions: The origins of cultural cognition. *Behavioral and brain sciences*, 28(5):675–690.

Tsetlin, M. L. (1973). *Automaton theory and modeling of biological systems.* [translated from the Russian] Academic Press, New York.

Turner, M. L. and Engle, R. W. (1989). Is working memory capacity task dependent? *Journal of Memory and Language*, 28(2):127–154.

Van Vaerenbergh, K., De Hauwere, Y. M., Van Moffaert, K., and Nowe, A. (2015). A policy gradient with parameter-based exploration approach for zone-heating. In *IEEE Symposium Series on Computational Intelligence*, pages 556–563.

Van Zandt, T. and Townsend, J. T. (1993). Self-terminating vs exhaustive processes in rapid visual, and memory search: An evaluative review. *Perception and Psychophysics*, 53:563–580.

Walsh, M. M. and Anderson, J. R. (2014). Navigating complex decision spaces: Problems and paradigms in sequential choice. *Psychological bulletin*, 140(2):466.

Watanabe, M., Kodama, T., and Hikosaka, K. (1997). Increase of extracellular dopamine in primate prefrontal cortex during a working memory task. *Journal of Neurophysiology*, 78(5):2795–2798.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards.* PhD thesis, University of Cambridge England.

Welch, G. and Bishop, G. (2000). An introduction to the Kalman filter. Technical report, Department of Computer Science, University of North Carolina, Chapel Hill.

White, C. C. (1991). A survey of solution techniques for the partially observed Markov decision process. *Annals of Operations Research*, 32(1):215–230.

Whitehead, S. D. and Ballard, D. H. (1991). Learning to perceive and act by trial and error. *Machine Learning*, 7(1):45–83.

Widrow, B., Gupta, N. K., and Maitra, S. (1973). Punish/reward: Learning with a critic in adaptive threshold systems. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-3(5):455–465.

Wiering, M. and Schmidhuber, J. (1997). Hq-learning. *Adaptive Behavior*, 6(2):219–246.

Wilson, R. C. and Niv, Y. (2011). Inferring relevance in a changing world. *Frontiers in human neuroscience*, 5:189.

Wilson, S. W. (1986). Knowledge growth in an artificial animal. In *Adaptive and Learning Systems*, pages 255–264. Springer.

Woods, D., Kishiyama, M., Yund, E., Herron, T., Edwards, B., Poliva, O., Hink, R., and B, R. (2011). Improving digit span assessment of short-term verbal memory. *Journal of clinical and experimental Neuropsychology*, 33(1):101–111.

Yntema, D. B. and Mueser, G. E. (1960). Remembering the present states of a number of variables. *Journal of Experimental Psychology*, 60(1):18–22.

Yu, A. and Dayan, P. (2003). Expected and unexpected uncertainty: Ach and ne in the neocortex. In *Advances in neural information processing systems 15 (eds. S.T.S. Becker, & K. Obermayer)*, pages 157–164. MIT press, Cambridge, MA.

Yu, A. and Dayan, P. (2005). Uncertainty, neuromodulation, and attention. *Neuron*, 46(4):681–692.

Zilli, E. A. and Hasselmo, M. E. (2008). Modeling the role of working memory and episodic memory in behavioral tasks. *Hippocampus*, 18(2):193–209.

Appendix A

Supplementary materials for Chapter 2

A.1 Fitting results with the confidence level ρ fixed at 0.65

Table A.1: The mean ± S.D. best-fitting parameter values by model and condition when ρ is fixed at 0.65, along with the total BIC scores.

Model	Parameter	Constraint	Fit value (SD)		Total BIC	
			Control	Test	Control	Test
PWRL	α	$0 \leq \alpha \leq 1$	0.31 (0.35)	0.09 (0.08)	3141	3202
	T	$0 \leq T < \infty$	0.59 (1.01)	0.69 (0.87)		
WTA-RL	α	$0 \leq \alpha \leq 1$	0.20 (0.36)	0.19 (0.31)	3064	3218
	T	$0 \leq T < \infty$	0.09 (0.07)	0.43 (0.24)		

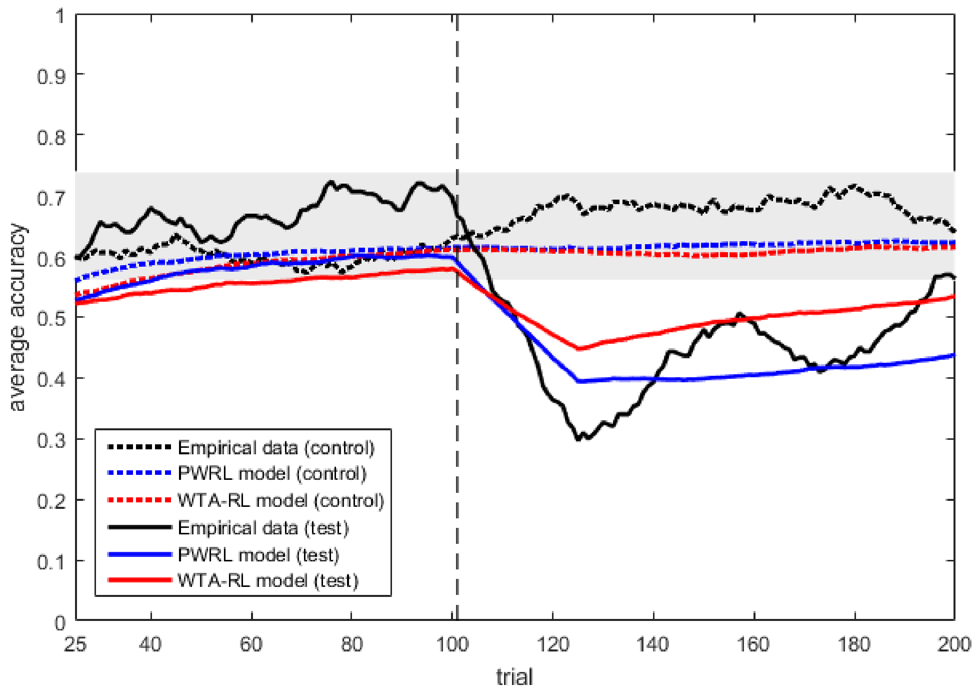


Figure A.1: Comparison of learning curves of participants and fitted models when ρ is fixed at 0.65. To draw model learning curves, we first constructed a learning curve for each subject by averaging over 100 simulations with her fitted parameters, then averaged across participants to get the overall learning curve. The light grey region represents a 95% confidence interval of the moving average with a probability of accuracy of 65%. The vertical dashed line indicates the trial where the switch occurs in the test condition.

A.2 Individual learning curves

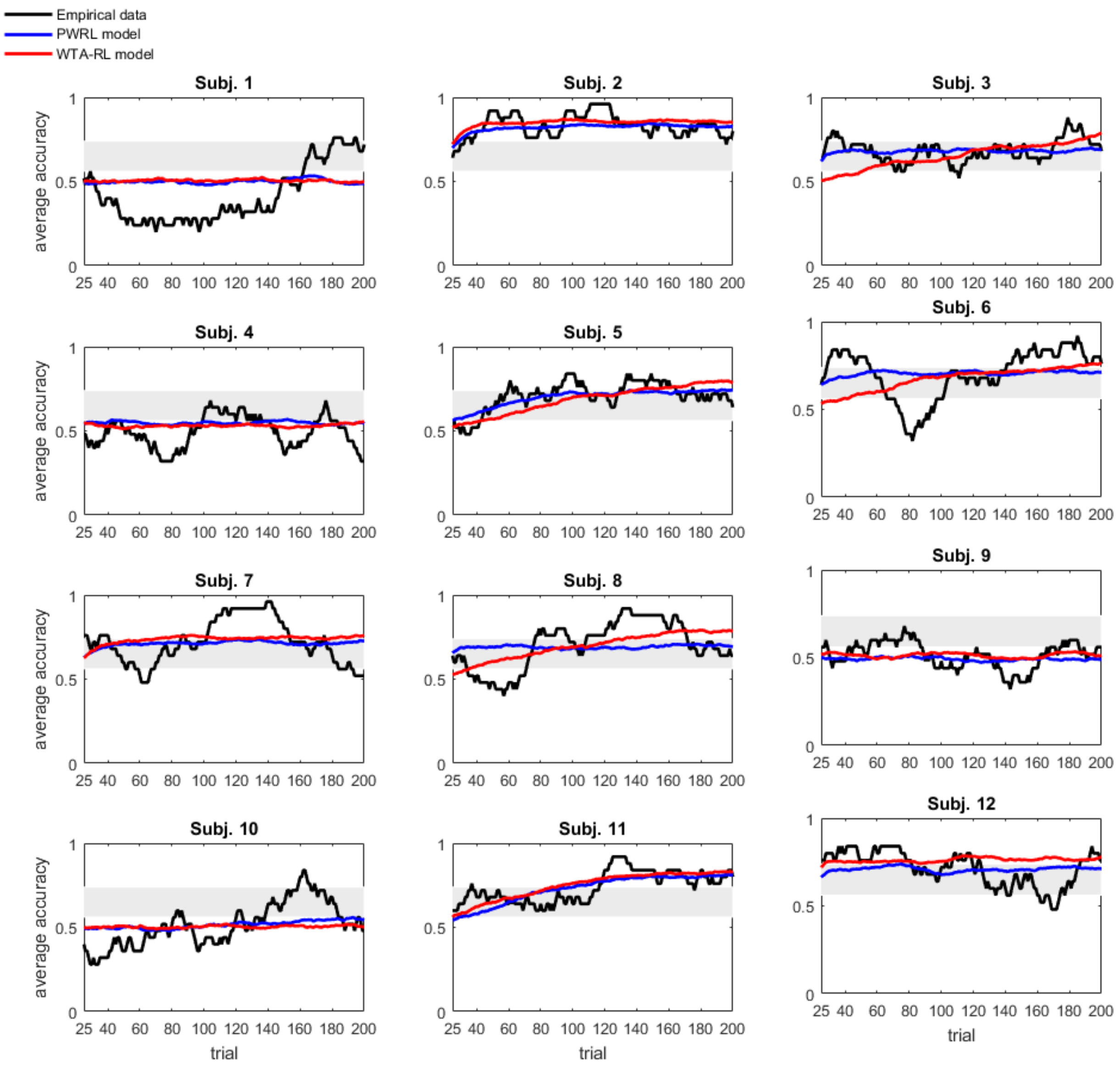


Figure A.2: Comparison of individual learning curves of participants and fitted models under the control condition. We constructed each participant's fitted learning curve by averaging over 100 simulations with her fitted parameters. The light grey region represents a 95% confidence interval of the moving average with a probability of accuracy of 65%.

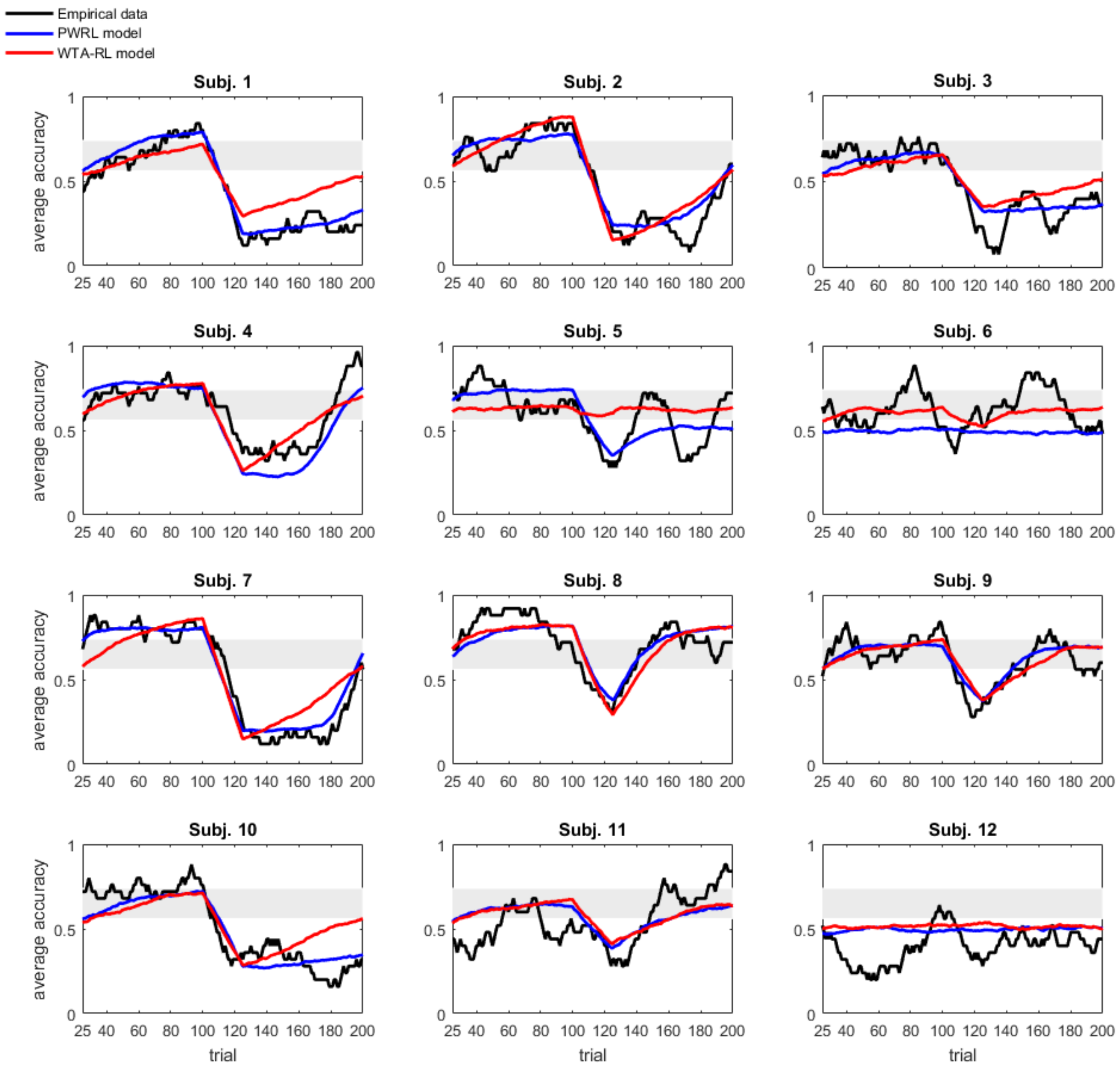


Figure A.3: Comparison of individual learning curves of participants and fitted models under the test condition. We constructed each participant's fitted learning curve by averaging over 100 simulations with her fitted parameters. The light grey region represents a 95% confidence interval of the moving average with a probability of accuracy of 65%.

Appendix B

Supplementary materials for Chapter 3

Table B.1: Summary of the steps followed by the agent under the Actor-Critic gating model of Todd et al. (2009).

Current state s_t, observation o_t	
Choose motor action a_t, and gating action g_t based on motor and gating state-action values Q^M and Q^G	$a_t \leftarrow Softmax(Q^M, s)$ $g_t \leftarrow Softmax(Q^G, s)$
Observe reward r_t and go to the next state s_{t+1}	$r_t \leftarrow a_t, o_t$ $s_{t+1} \leftarrow s_t, o_t, g_t$
Compute TD-error based on state value $V(s_t)$	$\delta_t \leftarrow r_t + \gamma V(s_t + 1) - V(s_t)$
Update motor and gating eligibility traces e^M, e^G	$e^M(s,a) = \begin{cases} \gamma\lambda e^M(s,a) + 1 & \text{if } s = s_t, a = a_t \\ 0 & \text{if } s = s_t, a \neq a_t \\ \gamma\lambda e^M(s,a) & \text{otherwise.} \end{cases}$
Update state eligibility traces e^V	$e^V(s) = \begin{cases} \gamma\lambda e^V(s) + 1 & \text{if } s = s_t, \\ \gamma\lambda e^V(s) & \text{otherwise.} \end{cases}$
Update state, motor and gating values	$V(s) = V(s) + \alpha\delta_t e^V(s)$ $Q^M(s,a) = Q^M(s,a) + \alpha\delta_t e^M(s,a)$ $Q^G(s,g) = Q^G(s,g) + \alpha\delta_t e^G(s,g)$
Go to the next trial	$s_t \leftarrow s_{t+1}$

Table B.2: Optimal parameter values by model and task.

Task	Model	Parameter	Best parameter value
12-AX	Two-gating Actor-Critic	α	0.1
		T	5
		λ	0.9
	Pure RL (zero WM cell)	α	0.2
		T	1
		λ	0
	One-cell RLWM	α	0.15
		T	2
		λ	0.9
	Two-cell RLWM	α	0.2
		T	6
		λ	0.9
12-XY	One-cell RLWM	α	0.2
		T	3.5
		λ	0.9
	Two-cell RLWM	α	0.2
		T	2.5
		λ	0.8
12-AB-XY	One-cell RLWM	α	0.2
		T	2.5
		λ	0.9
	Two-cell RLWM	α	0.2
		T	2.5
		λ	0.8

Appendix C

Supplementary materials for Chapter 4

Table C.1: Chosen parameter values for each model.

Model	Parameter	Parameter value
Pure RL (zero WM cell)	λ	0
	T	2
One-cell RLWM	λ	0.9
	T	3
Two-cell RLWM	λ	0.9
	T	4
Resource allocation (manual)	λ	0.9
	T_0	2
	T_1	3
	T_2	4
	N_1	10^4
	N_2	3×10^5
Resource allocation (automatic)	λ	0.8
	a_0	10^{-2}
	a_1	10^{-3}
	a_2	10^{-5}
	b_0	1.5
	b_1	1.5
	b_2	1.4
	w_0	10^3
	w_1	10^4
	ϵ_0	0.005
	ϵ_1	0.0025

Appendix D

Supplementary materials for Chapter 5

Table D.1: Parameters used for each model and task.

Task	Model	Parameter	Chosen value
12-XY	Bayesian one-cell RLWM	σ^2	0.5
	Standard one-cell RLWM	T	1
		λ	0.7
12-AB-XY	Bayesian two-cell RLWM	σ^2	0.1
	Standard two-cell RLWM	T	1
		λ	0.9
Both tasks	Both models	γ	0.75

Appendix E

Supplementary materials for Chapter 7

Here, we present four partially observable tasks that can be used to test working memory-based reinforcement learning models. These tasks have the particularity of having been tested on humans or animals.

E.1 Switch-state task (Clarke et al., 2015)

This is a grammar-like task, where there are 8 possible states represented by pictures (Figure E.1). The learner starts randomly at one location and has to reach the goal position (indicated by Yeah! in the figure) by passing through the switch state (here represented by the computer image). The episode will then terminate and a positive reward will be given. If the learner does not follow the rule of passing through the switch state before reaching the car position, she will be directed to the telephone state instead of the car state, and will still need to find an appropriate path to the goal. The objective of the task is to complete the episode as quickly as possible (to encourage the model to find the shortest paths, we could for example punish the agent by a small negative reward each time it goes to a new state).

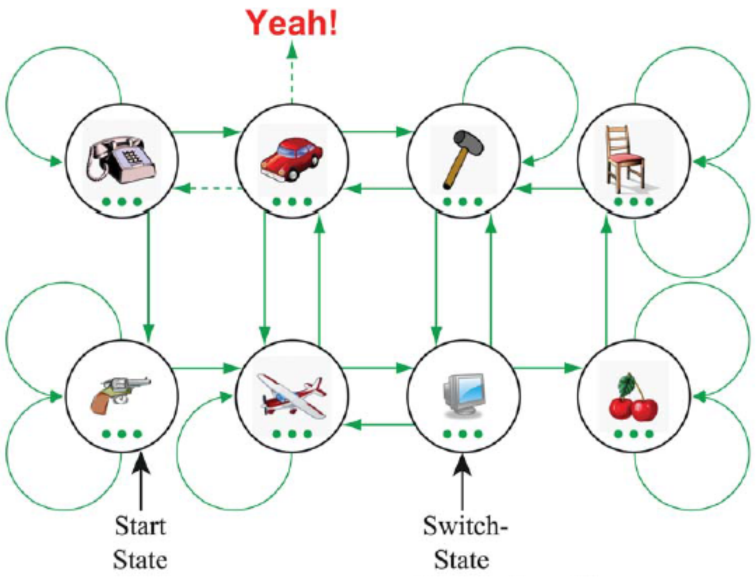


Figure E.1: Switch-statet task. Figure taken from Clarke et al. (2015).

This task can presumably be solved using a working memory of capacity one, mainly to store the information about whether the switch-state has been visited or not. To require the agent to use more than one working memory cell, we could simply add more switch states.

E.2 Reversal two-armed bandit task (Hampton et al., 2006)

This is a task adapted from Hampton et al. (2006)'s experiment, where at each trial, participants are presented with two pictures, and have to learn to select the one that gives the highest reward (equivalent to a two-armed bandit task). One interesting aspect of this task is that the reward functions of the two pictures switch if the best picture is selected three times in a row (see Figure E.2). The good picture (arm), represented by the green coloured circle in the figure, provides rewards according to the following reward function:

$$R_t = \begin{cases} +1 & \text{with probability } 0.7, \\ -1 & \text{with probability } 0.3. \end{cases} \tag{E.1}$$

The bad picture (arm), represented by the red coloured circle in the figure, gives reward according to this function:

$$R_t = \begin{cases} +1 & \text{with probability } 0.4, \\ -1 & \text{with probability } 0.6. \end{cases} \tag{E.2}$$

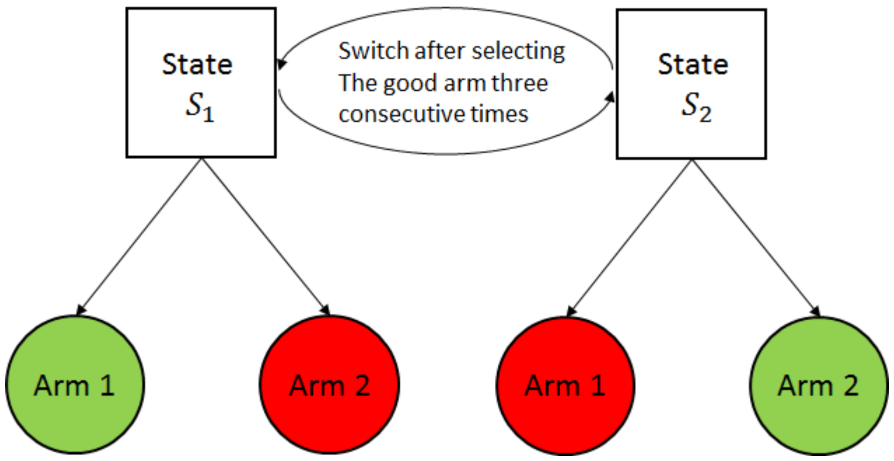


Figure E.2: reversal bandit task.

The RLWM model with three working memory cells should be able to solve this task by storing the last three actions. The advantage of using this task is that it has a stochastic reward function, as all the tasks that we used in the present thesis—and in the working memory-based literature in general—had deterministic rewards.

E.3 Partially observable maze task (Lloyd et al., 2012)

This task has been tested on rats. A rat starts at one of two arms R1 or R2, and has to learn to choose the arm at the other end of the maze which is opposite to the starting arm (see Figure E.3). Thus, the rat has to remember its choice at the guided turn position and use the same action at the choice turn (right turn in the figure). The advantage of this task is that it belongs to the family of navigation tasks, which have been used extensively to test working memory-based reinforcement learning models and partially observable Markov decision processes in general (mainly from a machine learning point of view).

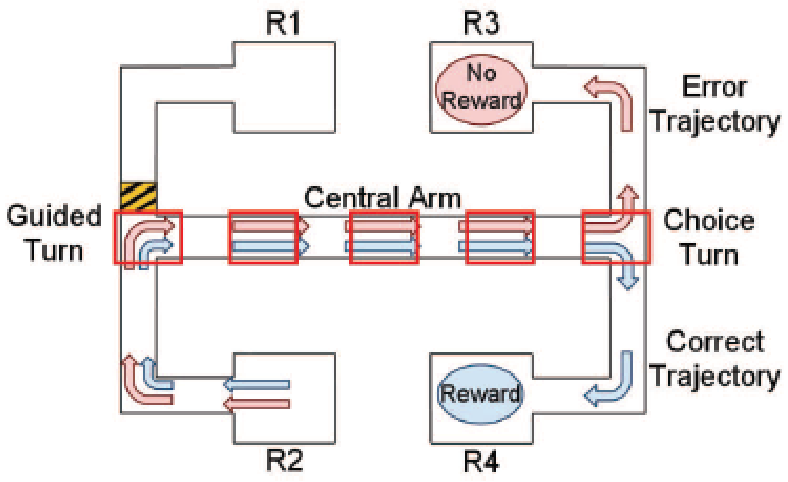


Figure E.3: Partially observable maze task. Figure taken from Lloyd et al. (2012).

E.4 Tanaka's delayed reward task (Tanaka et al., 2009)

This task does not require working memory to be solved, but requires the use of eligibility traces. Nevertheless, the task can be used to test whether or not the Bayesian model that we defined in Chapter 5 proposes a good alternative to eligibility trace.

The task requires participants to select one of two displayed pictures (see Figure E.4). There were 8 possible pictures, each one was associated with a reward that can be either positive or negative, and either immediate or delayed. Participants were instructed to learn to maximise their total reward. We can think of this task as involving multiple two-armed bandits with 8 possible arms. The reward associated with each arm is given in Table E.1.

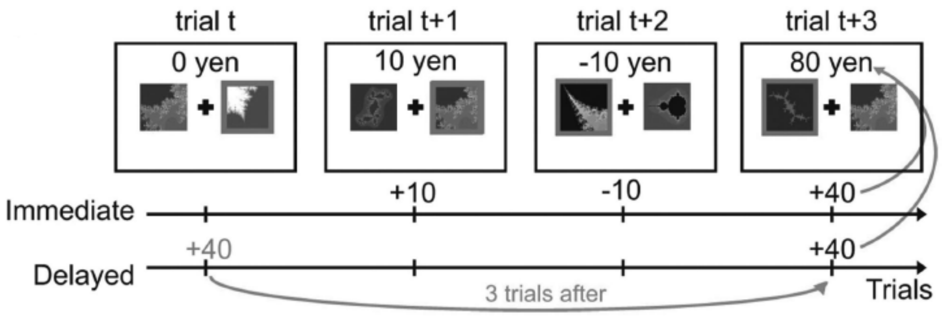


Figure E.4: Tanaka's delayed reward task. Figure taken from Tanaka et al. (2009).

Arm	1	2	3	4	5	6	7	8
Reward	+40(0)	+10(0)	-10(0)	-40(0)	+40(3)	+10(3)	-10(3)	-40(3)

Table E.1: Reward associated with each arm. +40(3) means that reward is 40 and that it is delivered after 3 trials, whereas +40(0) means that it is delivered immediately.