Imagination is more important than knowledge. For knowledge is limited, whereas imagination embraces the entire world, stimulating progress, giving birth to evolution.

– Albert Einstein, 1931.

University of Alberta

A COMPUTATIONAL MODEL OF LEARNING FROM REPLAYED EXPERIENCE IN SPATIAL NAVIGATION

by

MahdiehSadat Mirian HosseinAbadi

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

Spring 2012
Edmonton, Alberta

To my beloved parents

Abstract

In this thesis we propose a computational model of animal behavior in spatial navigation, based on reinforcement learning ideas. In the field of computer science and specifically artificial intelligence, replay refers to retrieving and reprocessing the experiences that are stored in an abstract representation of the environment. Our model uses the replay idea that existed separately in both computer science and neuroscience. In neuroscience, it refers to the reactivation of neurons in the hippocampus that were previously active during a learning task, in such a way that can be interpreted as replaying previous experiences. Therefore, it is natural to use RL algorithms to model the biological replay phenomena. We illustrated, through computational experiments, that our replay model can explain many previously hard-to-explain behavioral navigational experiments such as latent learning or insight experiments. There have been many computational models proposed to model rats behavior in mazes or open field environments. We showed that our model has two major advantages over prior ones: (i) The learning algorithm used in our model is simpler than that of previous computational models, yet capable of explaining complicated behavioral phenomena in spatial navigation. (ii) our model generates different replay sequences that are consistent with replay patterns observed in the neural experiments on the rat brain.

Acknowledgements

I have been very fortunate to have spent the most prosperous two years of my life here at the University of Alberta, and for this I am most grateful to Dr. Elliot Ludvig. He guided me through each and every step of my academic research, with enough patience to teach me about all the background material that I needed and did not have. This work could not be done without his valuable comments and ideas.

I would also like to thank my supervisors Prof. Renee Elio and Prof. Rich Sutton who provided me the opportunity to work on a research topic that I am truly passionate about, and helped me with their useful comments and advice. Many thanks to all the people in RLAI lab, and specially Gabor Bartok, for patiently helping me throughout the process of writing this thesis.

My special thanks to Babak Bostan, for his tremendous support in the past year. His knowledge and patience in answering my mathematical questions and listening to small details of my work seemed endless. Thank you for believing in me and always being there to encourage me when I most needed it.

Finally, I like to thank my sister and also best friend, Maryam S. Mirian, who has always been my greatest support in life.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis presents a computational model of certain aspects of how rats navigate through mazes and open field environments. Many animals can quickly learn to navigate complex environments, even finding shortcuts with limited experience in the environment. We use Reinforcement Learning (RL) algorithms and ideas to develop a computational model of spatial learning. RL has been studied in many other areas, such as behavioral psychology, statistics, economics, and game theory. However, in our model we only refer to its computational aspect. So, throughout this thesis every reference to the word RL implicitly refers to computational RL. The main idea in our model is the "experience replay" idea, hence we call our model "The Replay Model of Spatial Navigation".

In RL, *experience* is defined as a sequence of observations and actions that the agent receives in the environment. Based on this definition, *experience replay* refers to the process of retrieving past experiences that are stored in a world model and re-processing them. Based on this idea, our replay model works by maintaining a world model that stores real experiences, and then uses it to retrieve imaginary experiences and relearn from them.

It is also important to distinguish the two uses of the word "model" in this thesis. First, we use the word "model" to represent a structure that is usually more complicated. This enables us to further investigate the system, and predict its outcome in various situations. A computational model of the brain in this sense is a theory suggesting how the brain works, which can also recreate behavior similar to that in animals, and predict how animals should behave in certain situations if the model is a correct representation of their brain functionality. Since it might be impossible to precisely explain how the brain works, a better model is the one that can recreate more behaviors, while having as simple a structure as possible. Second, we use the word "model" as in the phrase "world model", which refers to the abstract representation of the environment that an artificial agent maintains

to remember its past experiences and enables it to rehearse previous memories along with generating novel sequences based on its current estimate of the environment.

Important neural substrates of the spatial navigation ability include the place cells in the hippocampus, which also sometimes show *experience replay*. In neuroscience, experience replay refers to the reactivation of neurons that were firing during a learning task, after the task is over and the subject is at a rest state. These cells are often reactivated in the same order as they were active during the learning task, in such a way that can be interpreted as replaying previous experiences. We show that our replay model is strengthened by these new findings recorded from rats' hippocampal neurons during a spatial navigation task.

The three main criteria that we use to evaluate our computational model are:

- Biological plausibility, and showing that our model is constrained by empirical data from rat hippocampus cells.
- Empirical adequacy, which is established by showing that our model can behave similarly to real animals in different behavioral tasks.
- Simplicity, which is shown by comparing our model to some of the main computational models proposed for spatial navigation and illustrating that our model is still capable of explaining similar experiments.

I begin this thesis with an overview of the background material necessary to understand the details of our model. The learning algorithm we used in our computational model is adapted from the temporal-difference learning method in RL and also uses the ideas presented in the Dyna algorithm for sampling from past memories. We systematically investigated several memory sampling schemes (inspired from existing RL algorithms and from empirical data), comparing model predictions with behavioral and neural data. This requires some background knowledge of RL which we present in Chapter 2. Our model shows that adding the idea of learning from imagined experiences enables sample-based RL algorithms to explain a wider range of behavioral experiments that animals can accomplish, and could inspire other RL algorithms that try to model animal behavior.

In Chapter 3 I overview both the behavioral tasks that rats can accomplish, and the evidence from their hippocampus neurons showing experience replay. I will later use these studies to conclude that our model is capable of explaining necessary experiments, and also is biologically plausible.

In Chapter 4 I look at some of previous computational models that are either proposed

to explain spatial navigation tasks, or have used the experience replay idea to explain other phenomena (the replay model of associated learning).

Our replay model for spatial navigation is described in Chapter 5. It works by having a world model that stores previous memories the agent has encountered, and then present them to the agent again during the learning phase. As a result, the agent will be learning both from the new experience it encounters each time, and also re-learning from samples of its past experiences. We describe the different components of our model along with the main learning algorithm, and also present a comparison between our replay model and the previous models with respect to the basic approach and also the simplicity of each model.

I evaluate our computational model in Chapter 6 based on its ability to show similar behavior as rats in two different behavioral experiments:

- Latent learning which happens where the rats do not show any signs of learning before the reward is presented to them, but after they receive the reward they learn more quickly than the control group which had received the reward from the beginning.

- The Tolman detour task, one of the first experiments showing evidences of "insight" in rats, where a part of a familiar path in the maze is blocked, and the rats show that they can remember other blocked paths and avoid them as well.

We illustrate that our computational model can successfully accomplish these tasks and also present a brief discussion on why other models can or cannot explain these tasks.

Finally, predictions of our replay model along with a brief conclusion are presented in Chapter 7.

Chapter 2

Reinforcement Learning: Background and Inspiration

Our computational model of spatial navigation in rats is based on sample-based Reinforcement Learning (RL) algorithms. In this chapter, I overview the background material necessary to understand the details of our learning algorithm and also describe the origin of the experience replay idea in RL. Basically, experience replay refers to retrieving old experiences and reprocessing them without needing to take actions. It was originally proposed to improve learning efficiency.

RL has been studied in many other areas, such as behavioral psychology, statistics, economics, and game theory. However, in our model we only use the terms, ideas, and algorithms from computational RL. So, throughout this thesis every reference to the word RL means the computational RL even if it is not explicitly mentioned.

I begin this chapter by providing a brief overview of the RL problem and the temporal-difference (TD) algorithm, which we use in our simulations. I also summarize different RL algorithms for choosing which experiences to retrieve from the world model. I will later explain how they are used in our computational model and show their similarity to replay patterns observed in empirical data from rat brains during a navigational task.

2.1 Introduction to Reinforcement Learning

Reinforcement Learning (RL) is a branch of artificial intelligence that provides a computational framework for solving real-time decision-making problems. RL formalizes the interaction between an *agent* and its *environment* in order to maximize the overall reward that the agent receives. The agent acts as both a learner and a decision maker. When modeling animal behavior, the agent can represent the whole animal or only its brain (which is

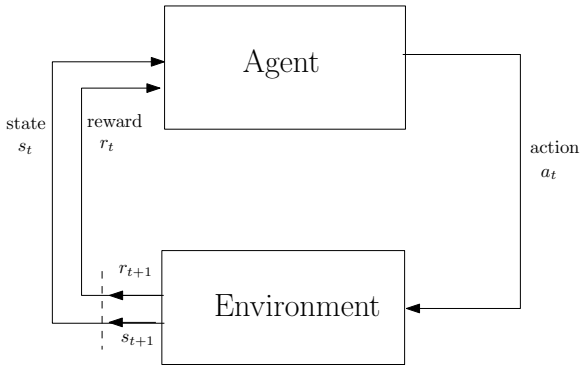


Figure 2.1: The interaction between RL agent and the environment (Sutton and Barto, 1998)

equivalent to the main control system). The environment includes everything the agent interacts with and cannot control. It provides the outcome of the agent's actions and rewards based on agent's previous states and action. By this definition of environment, the boundary between the agent and the environment is not necessarily the physical boundary we might imagine. For example, if we are modeling the brain of a rat trying to escape from a maze as the RL agent. The decision-making process happens in the agent, but the outcome which is physically moving the muscles (which might not happen because of physical disabilities), is considered part of the environment. Each RL problem can be viewed as a series of interactions between the agent and environment, to maximize the total reward. This sequence starts with the agent being in the initial state, and ends when it reaches certain terminal states where its location is reset to the initial state. At each time step in between (t) the agent is at state s_t and takes an action a_t. The environment then provides a reward signal r_{t+1} and the next state s_{t+1}. The interactions between agent and environment are shown in Figure 2.1.

The ultimate goal is to maximize the reward the agent receives. It is important that the agent chooses the actions that maximize the overall reward and not the immediate reward. To achieve this goal, the agent estimates how good choosing an action in each state is. This evaluation of state-action pairs is based on a function of rewards that the agent will receive afterwards. This function is usually referred to as the expected *return* and is formally defined as a function of future rewards:

$$r_{t+1}, r_{t+2}, ..., r_T$$

where T is the final time step. The return at time step t is denoted by R_t and is defined as

the discounted sum of subsequent rewards according to the following equation:

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

where $0 \leq \gamma \leq 1$ is called the discount factor and causes the rewards that are expected in the future to have less importance. The closer γ is to 1, the greater the effect of states far from the final state.

The agent follows a *policy* (π), which determines how it should choose its next actions. Formally, a policy is a mapping from each state to the probability of selecting each action in that state. If a policy always chooses the action which is expected to have the highest return, it is exploiting its current information. However, it might choose another action that is not the best, in order to explore it and makes its estimate of the environment more accurate. This action is called an exploratory action, and it is important that the agent (and any policy) maintains a balance between exploration and exploitation. If the agent does not explore enough and always chooses the best action based on its current estimate (greedy policy), it might find a suboptimal solution and stick to it as a final solution without ever finding the optimal solution. On the other hand, if it explores too much, it will always have to pay for the non-optimal steps it takes which decreases its final return. Therefore, the exploration vs. exploitation problem is a key factor for each agent.

A simple but very common policy which we also used in our simulations is called the ϵ-greedy policy where ϵ is usually a small parameter defining the probability of choosing a random action (exploratory step). Otherwise, the policy is greedy, meaning that it chooses the action which is the best according to the current estimation.

The spatial navigation problem is formally categorized as a Markov Decision Process because it has the Markov property. This property states that the next state and reward of an action in each state should only depend on the state and action just taken, and not the entire past trajectory. Formally, if we denote the action and state taken at time $t, t-1, ...$ by $a_t, a_{t-1}, ...$ and $s_t, s_{t-1}, ...$ respectively, then in the general case we would have:

$$Pr\{s_{t+1} = s', r_{t+1} = r | s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1}, ..., r_1, s_0, a_0\}$$

This equation means that the outcome, which is both the reward and the next state, depends on all states, actions and rewards experienced so far. However, this is a complicated case which usually does not occur. For example, in a maze, if we know the state of the rat (which is its location) and the action it takes, then the next state and reward might be the

same independent of the path it has taken to reach that location. Therefore we would write:

$$Pr\{s_{t+1} = s', r_{t+1} = r | s_t, a_t\}$$

This equation suggests that the next state and reward can be determined only by the current state and action. This property is called the Markov property. Each reinforcement learning problem with this property is called a Markov Decision Process (MDP). If the state and action sets are finite, the problem would be called a finite MDP. For instance, the maze navigation problem is also a finite MDP because it has the Markov property with finite state and action spaces.

To keep an estimate of how good each state is, the agent maintains a value function ($V(s)$) which assigns a value to each state (s). We might also assign a value to each state-action pair, and update this estimate each time that state-action pair has been chosen. In the latter case, the value is usually shown by $Q(s, a)$, where s and a belong respectively to the state space and action space of each state denoted by$S$ and $A(s)$.

2.2 TD Learning

Temporal Difference (TD) learning is one of the key ideas in RL. The idea is to update the current estimate of value functions based on the difference between the actual reward observed and the previous estimate.

More specifically, by choosing action a_t at state s_t, and going to state s_{t+1} and receiving r_{t+1}, the general update rule in the simplest version of this algorithm is as follows:

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$

The main idea behind TD learning which makes it widely used in the RL domain, is that instead of waiting for the trajectory to be complete before observing the outcome and updating the values, it updates the current estimate, based in part on the next estimate ($V(s_{t+1})$). In our problem, we focus on choosing the optimal state-action pairs, so we mainly deal with action-values ($Q(s, a)$) instead of state value functions($V(s)$). The simplest TD learning update rule in this case is as follows:

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

where s_t and a_t are current state and actions, s_{t+1}, a_{t+1}, and r_{t+1} are respectively the next state, action (chosen by following the current policy), and reward. This variation of TD learning is called SARSA, which is the abbreviation of the following sequence:

$s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}$ that is used in the update rule. The complete Sarsa algorithm is shown below.

Algorithm 1 SARSA algorithm: A version of TD-Learning

Initialize Q(s,a) for all $s \in S$ and $a \in A(s)$
Repeat for each episode:
    $s \leftarrow$ current state
    $a \leftarrow \epsilon$-greedy(s, Q)
    Repeat for each step of episode :
        Take action a, observe r, s'
        $a' \leftarrow \epsilon$-greedy(s', Q)
        $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$
        $s \leftarrow s'$, $a \leftarrow a'$
    until s is terminal

The idea in this algorithm is that in each step, the agent should look one step ahead and estimate the next action value, then update this estimation based on the reward it just obtained.

One of the powerful mechanisms in RL is called the *eligibility traces* and can be combined with TD methods for a more efficient algorithm. The idea behind this is that the error between the current estimation of action values and the actual reward obtained should affect the action value estimation of all state-action pairs visited prior to this state. Therefore, we maintain a *trace* of all state-action pairs that were previously visited and are thus *eligible* for an update. If we denote the value of this trace by $e(s, a)$, then the SARSA update rule can be generalized as follows:

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha \delta_t e_t(s, a) : \forall s \in S, a \in A(s)$$

where δ_t is the error between current estimate and the actual reward and is defined as:

$$\delta_t = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$$

As shown, all state-action pairs are being updated according to this rule. However, it is reasonable to assume that they should not all be affected by the same amount, and those which have been visited recently should undergo more change. To achieve this, we keep an array of eligibility traces called $e(s, a)$ initialized as 0. Every time a state-action pair (s, a) is visited, we increase its corresponding element by 1, and then multiply all elements in the array by a constant variable $0 \leq \lambda \leq 1$, which defines how quickly previous state-action visits should be forgotten. If $\lambda = 1$, this means that all state-action pairs should be impacted independently of how recently they have been visited. Eligibility traces are

updated according to the following rule:

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

Combining this idea with the previous algorithm leads to a $Sarsa(\lambda)$ algorithm where the parameter λ shows that we are taking into account the full visited trajectories with their weights exponentially decreasing by a factor of λ. The complete Sarsa(λ) algorithm is shown below:

Algorithm 2 Sarsa(λ) Algorithm

Initialize Q(s,a) for all $s \in S$ and $a \in A(s)$
Repeat for each episode:
$s \leftarrow$ current state
$a \leftarrow \epsilon - greedy(s, Q)$
Repeat for each step of episode :
Take action a, observe r, s'
$a' \leftarrow \epsilon - greedy(s', Q)$
$\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$
$e(s, a) \leftarrow e(s, a) + 1$
For all s,a :
$Q(s, a) \leftarrow Q(s, a) + \alpha\delta e(s, a)$
$e(s, a) \leftarrow \gamma\lambda e(s, a)$
$s \leftarrow s'$, $a \leftarrow a'$
until s is terminal

2.3 RL architectures with replay

2.3.1 Model-based replay

The idea of Dyna was presented for the first time as an AI architecture to integrate the process of learning and planning together incrementally as the agent is exploring the environment in real-time (Sutton, 1990). The term "Dyna" comes from dynamic programming which is a method for learning the optimal value function.

In the Dyna algorithm, a world model is maintained which includes any information the agent uses to predict how the environment produces next states and rewards. It can be deterministic, meaning that for each state and action the next state and reward is stored, and changed if a new experience is observed. On the other hand, a stochastic model includes the transition probabilities (the probability that an action in each state will lead to a certain state) and after each new experience only these probabilities are updated. There is one key difference between these two that reveals when the environment is changing. If the environment is deterministic, the model will always contain the latest event experienced

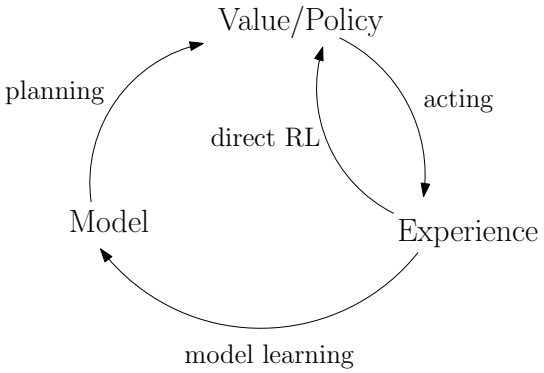


Figure 2.2: Relationship between learning, planning and action (Sutton and Barto, 1998)

by the agent no matter how many times previous events had occurred. However, if the world model is stochastic and maintains transition probabilities, encountering a change in the environment for the first time will not greatly affect the model. In this case, deciding how the agent should store past experiences mainly depend on the nature of the task.

When the agent uses this world model to predict how the environment behaves, it is *planning* its next actions. Dyna (Sutton and Barto, 1998) combined the idea of learning, planning and actions together. Figure 2.2 shows how these steps interact with each other. The agent uses its experiences to learn value functions using an RL algorithm such as the previously described TD learning (Direct RL). Experience is also used to update a model (model learning). This world model is then used for planning, which also uses the same learning algorithm. The policy uses the learned value function to determine next actions, so produces new experience again, and completes the cycle.

The following algorithm shows the simplest form of Dyna. As can be seen, after each time step a reward and next state are observed, the model is updated and then N planning steps are taken. In the simplest form, each replay step includes choosing a random state and taking a random action in that state, predicting the outcome based on the world model, and finally updating the value functions using the same update rule.

2.3.2 Trajectory-based replay

The TD learning algorithms only use new interactions with the environment to learn action values, and do not include any planning steps. Some experiences might be rare or highly costly. Therefore, there should be a mechanism for the agent to learn from these experiences efficiently without needing to revisit them. This motivated the idea of experience replay which was originally due to Lin (1992). In this paper he proposed the experience replay idea as an extension to previous RL frameworks and showed how it speeds up the

Algorithm 3 Dyna algorithm using Sarsa updates (Sutton and Barto, 1998)

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in Sand a \in A(s)$
loop
  $s \leftarrow$ current state
  $a \leftarrow \epsilon - greedy(s,Q)$
  Execute action a; observe resultant state s' and reward r
  $a' \leftarrow \epsilon - greedy(Q,s')$
  $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(a,a)]$
  $Model(s,a) \leftarrow s', r$
  for N replay steps **do**
    $s \leftarrow$ random previously observed state
    $a \leftarrow$ random action previously taken in s
    $s', r \leftarrow Model(s,a)$
    $a' \leftarrow \epsilon - greedy(Q,s')$
    $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(a,a)]$
  end for
end loop

learning process in TD-based algorithms with less experience needed. In the original paper, experience replay referred to the process of continuously presenting past experiences to the learning algorithm. This idea sped up the process of learning optimal values in a stationary environment. It should be noted that if the environment changes quickly, previous experiences are irrelevant and replaying them only decreases the learning rate.

In addition to experience replay, the idea of "backward replay" in the RL context was presented in Lin (1992) and referred to replaying previous experiences in a temporally backward order. Lin (1992) showed that using this strategy makes experience replay method more effective.

There is one main difference between experience replay and Dyna: In Dyna a hypothetical state is chosen randomly at each replay step and then the agent uses the same policy to choose the next action. This means that some sequences can be generated in planning steps which might have never been experienced. This does not happen in the experience replay model because the hypothetical steps are sequences that have been previously observed and stored in the world model.

2.3.3 Prioritized Sweeping

The idea of prioritized sweeping was presented in Moore and Atkeson (1993) and Peng and Williams (1993). In this idea the replay steps are directed toward the states that need more learning that are the ones that have led to a greater prediction error. For this

prioritization scheme, the most recent prediction error obtained for each state is used to build a priority queue. Every time a state-action pair is visited and the prediction error of this visit is more than a certain (usually small) threshold ϵ, this state along with all other states leading to it are placed into the queue. Therefore, the first element of the queue will always contain the transition that has led to the greatest change. Afterwards, instead of picking a state-action randomly from a uniform distribution over all experienced states, the top element of the queue is always retrieved for replay.

Another variation of the prioritized sweeping algorithm was due to McMahan and Gordon (2005) which only added those states whose value function had recently changed and not its proceedings.

One difference between prioritized sweeping and the original Dyna idea is that in the Dyna algorithm, a fixed number of replay steps is repeated in each step of the algorithm. However in the prioritized sweeping algorithm, the process continues for a fixed large number, or until the queue has no more elements, which happens when the learning has been completed and all prediction errors are below a certain threshold.

Although in these papers the authors only proposed prioritized sweeping as a heuristic and did not provide any mathematical proof for the convergence of the algorithm, they showed how this enabled the TD learning model to solve real-time problems with even large state spaces more efficiently than previous RL schemes.

2.3.4 Dyna2 Architecture

In the original Dyna algorithm the world model was represented by a tabular matrix. This limited its use to only the problems with relatively small state spaces. The Dyna algorithm was later extended by presenting the Dyna2 (Silver et al., 2008) algorithm to account for large state spaces using linear function approximation. Silver et al. (2008) showed that this algorithm converges to a unique solution independent of how samples are generated in the planning steps. They also presented the algorithm for prioritized sweeping with linear function approximation (linear prioritized sweeping) and proved that it converges to the optimal point which was previously only shown empirically.

Chapter 3

The Psychological Phenomena

In this chapter I present the behavioral and neurobiological phenomena, which I will use in Chapter 6 to evaluate our computational model of rats navigation. First, the studies that have reported how rats behave in different mazes or open field environments. These studies provide a guideline to verify if the computational models can correctly mimic the main features of animals' behavior. Second, the studies that have revealed some facts about how navigation takes place in the brain by looking at the neuronal activities during and after performing a spatial navigation task. I briefly explain which neurons are believed to be involved in navigation, and elaborate on the origin of the term "experience replay" in neuroscience. These studies are later used to show that our model is biologically plausible.

3.1 Selected Behavioral Phenomena in Spatial Navigation

As previously mentioned, in addition to the biological plausibility of a computational model, we also expect it to be able to behave similarly to animals in navigational tasks. In order to evaluate different models based on this criteria, we need to establish standard experimental results that studies have agreed upon, and that computational models try to explain. In this section I explain these experiments that I later use as a basis for evaluating the behavior of our navigational model against animals and other computational models.

3.1.1 Acquisition

The first and most important behavior is acquisition, in which the animals learn to go to a reward location in the environment. The environment does not provide any clue about the position of the reward or the direction in which the animal should go. The reward is something the animal is highly motivated to achieve (food when it is hungry, or escaping from a water maze for rats, etc.). The animals are motivated to find the shortest path toward

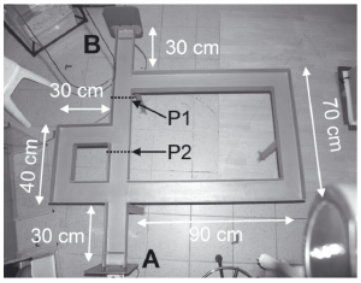


Figure 3.1: Maze configuration in the detour task regenerated in (Alvernhe et al., 2011). A is the initial state and B is the reward location. The rat first explores all three paths, then P2 is blocked and it learns to choose the left path in one trial. Similarly, after adding a blockade at P1 the rat chooses the right path in the next trial.

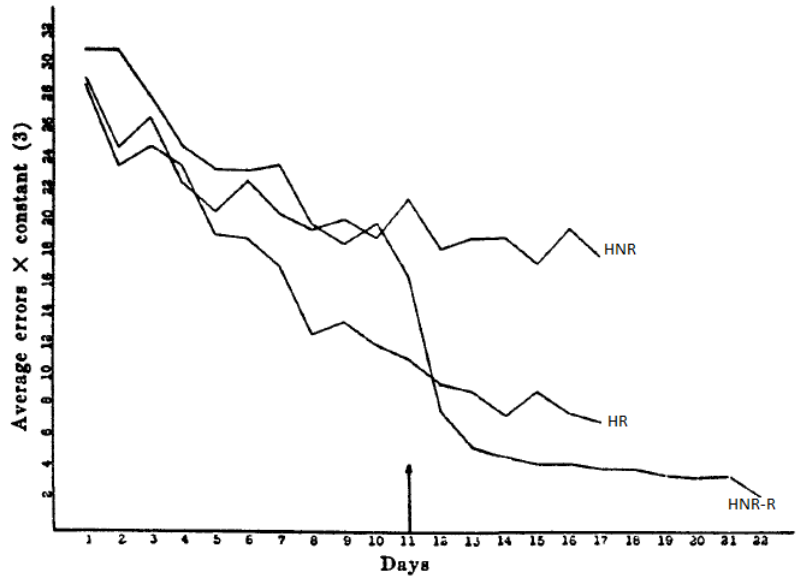
the goal, and the more experience they get, the fewer errors they make in finding their paths. The reward is usually placed far from the initial location (for example at the end of a multiple-T maze)

There are different variations for this experiment which make the task harder. For example, placing the animal in different start locations will prevent it from memorizing the series of actions which have previously led to goal. This forces the animal to learn the structure of the environment. Another variant is to place the maze itself in different locations at each day and include several trials. This is called Delayed Matching to Place (DMP) and ensures that rats only learn their paths within the local environment, without any external clues. The third alternative is to change the reward location each day, whether it is the escape gate from a water maze or a food reward. Rats have shown a gradual decrease in the number of errors they make to find the goal each day, and after a few days they can find the goal after only one trial (Foster et al., 2000). This suggests that the rats are learning a representation of the environment which is independent of the goal.

It should also be mentioned that after each trial, subjects are picked up and placed into the start locations again. This will make them lose track of their origin if they learn relative distances to the goal, and causes inconsistency which is referred to as the *global consistency problem* (Foster et al., 2000).

3.1.2 Tolman Detour Task

This classical experiment was proposed by Tolman (1948), and showed the first evidence that rats do not merely learn stimulus-response associations, they build a cognitive map of their environment which allows them to quickly plan novel paths in the environment. The original maze configuration for this experiment is shown in Figure 3.1. First, the rats are



Error curves for HR, HNR, and HNR–R

FIG. 6

(From E. C. Tolman and C. H. Honzik, Introduction and removal of reward, and maze performance in rats. *Univ. Calif. Publ. Psychol.*, 1930, 4, No. 19, p. 267.)

Figure 3.2: Original result of the latent learning experiment by Tolman (Tolman, 1948). The HNR group does not receive any rewards. The HR group receives the reward from the first trial, and the HNR-R group is rewarded only after the 11th trial (shown by the arrow). The fast drop in the average number of errors show that the pre-exposure to the environment has affected the learning although it was not shown before presenting the reward.

given enough trials to explore all three different paths. If the middle path is blocked at point P2, and the rats immediately choose the second shorter path on the left. Alternatively, if the shortest path is blocked at point P1, then the rats immediately avoid choosing both blocked paths and go to the right, even though they have only experienced the block while being in the middle section.

3.1.3 Shortcutting

In an open field experiment and using environmental cues, animals are trained to navigate to the reward through a long path, while the direct path is hidden or blocked. Once the blockade is removed and the shorter path becomes available, the animals choose the second path although they have not previously taken it.

3.1.4 Latent Learning

The idea behind this experiment was originally proposed by Blodgett (1929). The main idea is that the animal first runs through several trials where no reward is present, and then it is

exposed to the reward. It does not show any sign of learning before being exposed to the reward, so this type of behavior is called *latent* learning. Tolman and Honzik ran this experiment on three different groups of rats in a maze. The first group (HNR) never had a food reward in any of the trials. The second group (HR) had the reward in a specified location from the very first trial. The third group (HNR-R), which was the main experimental set, had the reward in the maze only after the 11th trial. The result of the original experiment is shown in Figure 3.2. There is a major drop in the number of errors after the 11th trial, which is faster than the gradual decrease observed in the HR group. This drop suggests that learning had been occurring even during the trials without any rewards present. Notice that the first group, which did not receive any reward, also showed a slight decrease in the number of trials because escaping from the maze is itself a small reward for the animals, although not explicitly specified.

3.1.5 Spatial Alternation

In this set of experiments, the reward location changes at each trial based on the animal's previous actions. For example, in a T-maze configuration, the reward is first presented at the right arm, but then is presented to the animal only if it chooses the opposite direction of its previous trial. This is usually challenging for computational models because it removes the Markov properties of the task by making the problem dependent on the animal's previous actions.

3.2 Spatial Navigation in the Brain

3.2.1 Place Cells

There have been many experiments showing that neurons in the dorsal hippocampus are active when an animal is at a certain location in an environment but not in others. These units are called *place units* or *place cells* since they encode the location of the animal in the environment. The term *place field* refers to the place in the environment to which these units are mapped (O'Keefe and Conway, 1978). This mapping is then used by the brain to build a cognitive map of the environment. It has been shown that the firing of these cells indicates that the animal has entered the corresponding place field, regardless of its speed or head direction (O'Keefe and Burgess, 2005). These place cells are not bound to a specific environment, but within each environment, their firing is mapped to only a certain portion of the environment. As shown by O'Keefe and Burgess (2005), some of the place cells

respond only to the spatial location, whereas others respond to the specific objects or cues in that location. For example, a small percentage of place fields are considered as "edge fields," and fire when the animal is at the edge of the environment (Burgess et al., 1994). So, each place cell participates in the representation of more than one environment (Burgess et al., 1994).

There is no detailed explanation for how the mapping from places to place cells is done, but there are several possibilities. One is that this mapping is only dependent on certain cues in the environment, meaning that by knowing all the environmental cues (geometrical, sensory, etc.) we can find the corresponding place unit. However, experiments have revealed that these place cells do not fire only due to external stimuli in the environment (such as color or smell), although a correlation exists between them. For example, if all the external environmental cues are removed (light, color, smell, or a food reward) then many of the place fields are lost, but some continue to fire, and sometimes their firing rate is increased (O'Keefe and Conway, 1978). Another option is that they might also depend on the animal's activity in those areas. For example running or pausing will affect the place cell mappings. However, by ruling out many of these factors (for instance, forcing the animal to maintain the same speed, or turn in one direction) the same results still hold. This suggests that some of the place cells are encoding the information about the animal's location, regardless of external cues in the environment (Alvernhe et al., 2008).

It should be mentioned that "grid cells" are also involved in spatial navigation. Grid cells are neurons in the entorhinal cortex that fire when the animal is at any of the three vertices of a grid of triangles (Hafting et al., 2005) that covers the entire two-dimensional environment. The spacing, orientation and spatial phase of these triangle-shaped grids is believed to help the animal build a spatial map. In contrast to place cells that fire over a continuous portion of the environment, these grid cells fire periodically with different frequencies. There are several theories suggesting that the firing of grid cells at a given location is mapped to the place cells centering at that location (O'Keefe and Burgess, 2005). Since then, many computational models have been proposed to find the underlying mechanism of this mapping (Rolls et al., 2006). However, in this work I do not focus on how place cells form a cognitive map, and rather focus on how this representation can be used for navigation.

3.2.2 Experience Replay in the Hippocampus

As mentioned, the firing sequence of place cells was mapped to the sequence of locations in the environment that the animal had visited. Later, it was observed that these place cells continued firing in the same sequence even after the task was over and the subjects were at a rest state (Pavlides and Winson, 1989). More specifically, for a pair of non-overlapping place fields, their place cells fire in the same order that the rat had visited them (Skaggs and McNaughton, 1996); and no such firing sequence was recorded before the rat had experienced those place fields (Pavlides and Winson, 1989; Wilson and McNaughton, 1994; Skaggs and McNaughton, 1996).

This pattern of reactivation was called *experience replay* and was believed to play a key role in memory consolidation. It is used in many theories about the role of sleep in learning (Maquet, 2001) and retrieval of recent memories. It has also been observed that the firing rate of these patterns was 6-7 times faster than what was observed during the actual experience on rats (Euston et al., 2007), suggesting that when there is no physical constraint of actually moving, the brain can replay the experiences 6-7 times faster.

Another study of the rats' hippocampus activities where they were running in a T-shaped maze showed that the same experience replay behavior occurred at the decision point where the animal paused (Johnson and Redish, 2007). More importantly, when the rat was at the junction point, the replays mostly started from its current location, and swept both stems of the maze forward. This provided evidence for suggesting that replays also have a role in decision making. If they were only passive replays of past experiences, they should have contained the path the animal had taken before decision point. However, replaying both possible directions forward can suggest that the brain is actively planning.

Evidence suggests that place cells do more than representing current spatial location; they may show the structure of the environment by storing the connections between states. Thus, they provide information about available routes in the environment (Alvernhe et al., 2008) which makes the hippocampus essential for navigation. This role of hippocampus enables the animal to predict the future states based on current location. If this is the case, then a change in the structure of the environment (and not current state of the animal) should be reflected in place fields firing as well. Alvernhe et al. (2008, 2011) tested this hypothesis and investigated the effect of changing the environment on place cells mapping.

Alvernhe et al. (2008) looked at place cells of rats while they were running through a familiar maze, and also after adding a new shortcut to the maze. The results indicated

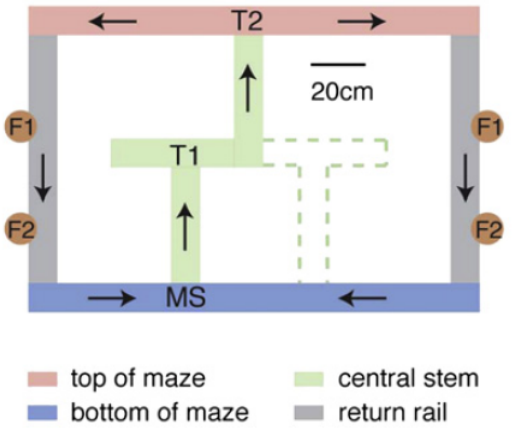


Figure 3.3: Maze configuration used in the experiment to show the relationship between relay content and experience (Gupta et al., 2010). Only one of the T-shaped paths is open at each trial and the rat is rewarded only if it chooses the opposite direction of the last trial at location T2. Arrows indicate a one-way pathway.

that once the new shorter path became available (which all the subjects found and used it immediately), the firing rate of place cells (especially in the CA3 region) mapped to those locations was highly increased. As similar experiment was performed to test the other case: a barrier was added to an optimal and familiar path which forced the rats to take the longer path in the maze (Alvernhe et al., 2011). Results were consistent with previous findings, and indicated that the firing rate of place cells around the location of the blockade was increased. They adapted quickly to the new obstacle, and chose the detour.

All these experiments strengthened the idea that place cells have a role both in both embedding information about the connection between states in the environment and building a cognitive map (since the changes in states connectivity in both cases increased place cells' firing rate), and rats' navigation (since the rats rapidly adapted their behavior to the change).

One theory explaining the relationship between experience and replays, suggested a correlation between the number of times a sequence of place cells is activated during the task, and the number of times it is replayed during awake rest states or sleep periods after trials (Jackson et al., 2006). It was assumed that reactivation should increase with experience. However, Gupta et al. (2010) showed new behavior in replay patterns which could not be explained by frequency-based theories. This research suggested that the relation between replay patterns and experience might not be as straightforward as previously assumed.

They tested a group of rats on a maze configuration which is shown in Figure 3.3. There are two paths available (the second one is shown in the dashed line), two decision points: T1 and T2, and two reward locations (with a food reward): F1 and F2. During the training sessions, a reward was presented based on the choice point at T2: it was either in the right path, left path, or alternating between right and left, and this contingency was changed daily. So within each session, the rats sampled both loops, but how long it has been since a loop was traversed was based on the reward contingency. They recorded the neural activity of place cells during the period when the animal paused at reward locations and rest sessions after each trial session and found three main characteristics in replay content:

- **Forward and backward replay**

  The replayed trajectories contained forward paths (starting from the animal's current location and moving forward towards the goal) and backward paths (starting from reward locations, or from its current location sweeping the path it had just taken. This backward replay has been observed in previous experiments (Foster and Wilson, 2006), but in this case it was specially interesting because the subjects were not allowed to move backward in the experiment by Gupta et al. (2010). As a result, these backward trajectories were considered new imaginary trajectories.

- **Opposite side replays**

  When the reward location was changed, although the rat quickly adapted to the change and moved toward the new location, it still continued replaying the trajectories of the opposite site (old reward location). Specially, Gupta et al. (2010) also showed a forward replay of trajectories leading to the reward sites, and backward trajectories starting from reward locations. This strengthens the theory that although the rat is taking the right paths, they keep thinking about the parts that were surprising.

- **Shortcut sequences**

  Another interesting pattern observed was what happened with the shortcut paths, which are straight paths from F1 to F2 or vice versa. Considering that rats had not experienced backward trajectories, this observation provides evidence that the hippocampus also generates new trajectories, and does not only exhibit passive replays of past experiences.

3.3 Summary

In this chapter two categories of studies on rats were presented. The first section reported how rats behave in different spatial tasks, and established an empirical testbed for the computational models. We looked at some of the most important ones including acquisition, finding shortcuts and detours, latent learning, and spatial alternation which we will later use to show how our computational model behaves.

The second section established the hippocampal place cells are important in encoding spatial information about the current location of animals in their environment. I also explained how they are believed to maintain an abstract representation similar to a cognitive map that contains information about available paths in the environment. I then explained some experimental results that have shown reactivation of place cells firing after a task or when the animal is resting or sleeping. These studies suggested that replays are not merely passive reactivation of firing of previous active place cells, and some novel patterns recorded from rats' brains strengthened the theory that replays may play an active role in learning.

Chapter 4

Related Computational Models

> When walking alone in a jungle of true darkness, there are three things that can show you the way: instinct to survive, the knowledge of navigation, creative imagination. Without them, you are lost.
>
> Toba Beta, My Ancestor was an Ancient Astronaut

In this chapter, I review previous computational models that have been proposed to explain spatial navigation in rats. I use these models later in Chapter 6 and show why they can or cannot explain various behavioral tasks and demonstrate their differences to our replay model.

In the second section of this chapter, I describe how the replay idea is used in other areas. As an example, I illustrate how integrating the replay idea into conventional models of associative learning enables them to explain many hard-to-explain phenomena.

4.1 Computational Models of Spatial Navigation

Most proposed models use the idea of TD learning described in section 2.2. Though the implementation might be different, the key idea is to learn based on the error between a prediction and an observed outcome. Thus, TD-based models are referred to as reward-based models since they are dependent on a reward signal. These models are capable of efficiently learning correct optimal values and do not depend on the geographical or topological structure of the environment as long as the reward location is consistent. However, once the reward location is changed, the behavior of these models diverges from that of animals. This happens for two reasons: first, the previous value functions become misleading and guide the agent to wrong directions, which reduces the performance. Second, the agent

needs to learn new values for each state and cannot use its past knowledge, so it requires more experience. This is also the opposite of the one-trial learning behavior that the animals demonstrate in the DMP task.

TD learning with Metric Coordinates

Foster et al. (2000) simulated the firing of 493 place cells which covered the environment (the same number that they found through experiment) by a Gaussian distribution where the centers were set to the recorded centers of each place field and the breadth of each field was set to 0.16 (also the same as the measured value). These values represented the state of the agent in the environment, and the actions were eight possible directions the agent could take. The reward signal was set to 1 if the agent was in the specified reward location at the right time, and 0 otherwise. Using the basic TD-model, the agent tried to learn the appropriate value function of each state (which was defined by simulated place cell firings) and was able to show the one-trial learning behavior in a simple acquisition problem, and also when the start location was changed at each trial.

Foster et al. (2000) added a metric component to the model to learn the coordinate system of the environment. This was inspired by dead-reckoning abilities in rats. This ability provides animals with some information about head direction enabling them to estimate their current self-motion. This information is provided by cells in the postsubiculum that their firing represents the head direction of the animals at each moment (Brown and Sharp, 1995). By simulating this information, Brown and Sharp (1995) simulated the direction in which the agent was moving and the speed. Combining this with the information about current location which was provided by place cells, the agent could accurately update its coordinates. Although it did not receive any direct information about the origin of this coordinate system, it was shown that the learned coordinates were stable over different trials. The metric and TD learning components worked tightly together in such a way that the metric system provided the direction toward the location that was determined by the TD component as a goal. However, when there is no reward location available (for instance, during a DMP task where the agent finds that the reward is no longer at the location it had previously predicted) the coordinate system generates random exploratory actions. This system would be goal-independent and enable the agent to use its previous knowledge to quickly adapt to the new reward locations.

Modeling the role of working and episodic memory

Animal behavior and spatial navigation are complicated processes and cannot be easily explained. One of the reasons is that not only does their behavior depend on their current state, it is also affected by the animal's past memories. This is troublesome for the computational models (and specifically RL-based models) since it removes the Markov property condition of the problem (which requires that the problem depends only on the current state and action). In RL, this problem can be solved by including the necessary information about animal's past decisions in its current state representation. Zilli and Hasselmo (2008) used this idea and included this information about past decisions in a memory. There were two types of memory considered: (i)Working Memory (WM) which was defined to be active only for a short period of time and (ii) Episodic Memory (EM) which stores past memories. Zilli and Hasselmo (2008) extended the RL models to include these two types of memory and make the RL model more similar to what animals' brains do in navigation tasks. In order to keep the Markov property, they considered these two memory components as part of the environment and not the agent itself. Therefore at each time step, the agent received its physical location as its state (S) along with current representation of WM and EM. The action values were also estimated for all possible combinations of S, WM, and EM instead of only for possible states. WM contained current state and next action until the action was actually taken. Therefore, the set of possible WMs defined the policy that the agent was following.

EM was implemented as a content addressable memory and contained a list of n most recent states and actions experienced. Moreover, EM was temporarily indexed so that by retrieving a specific state, the whole trajectory (other states that are temporally close to it) could be replayed. The rest of learning algorithm was similar to TD, with an ϵ-greedy exploration policy. Since WM contained short-term states and actions, Zilli and Hasselmo (2008) suggested that if a task can be solved by only considering the recent experiences, modeling WM would be sufficient. Otherwise, for tasks that need memories from previous experiments, EM should be used as well. By incorporating the role of memory systems, they extended the domain of problems that RL could model to include more complicated tasks which had a delay imposed.

Modeling hippocampus replays with TD learning

The paper by Johnson and Redish (2005) was the first to suggest that the idea of experience replay in RL is closely related to the replay of recent memories observed in hippocampal

cells during sleep or rest periods after a task. In this paper, they used the TD learning model with replay steps on a simulated multiple-T choice-tasks which contained four T-shaped choices. A reward was presented only at one arm of the final T. The simulated replay steps started at a random state that was previously experienced, but chose the action that was most similar to the change in that state. The error was defined as entering an incorrect arm of the T mazes.

Johnson and Redish (2005) showed two main effects in their computational model: First, TD learning algorithms without a replay component, learn more slowly than real animals. Also by adding replay steps, the learning rate speeds up and becomes more similar to the learning rate of real animals.

Second, the same path stereotypy behavior which was seen in rats was observed in the agents as the calculated correlation between the paths they chose increased over time (Although the performance was still slower than rats' performace).

Johnson and Redish (2005) predicted that replay should develop with experience. However, this might not always be the case. For example, by using the idea of prioritized sweeping, with more experience the prediction error that the model generates decreases and as a result fewer state-action pairs are added to the priority queue, which will result in fewer replays.

A model of hippocampus function

Burgess et al. (1994) presented a computational model for hippocampus function by approximating the firing rates of place cells using radial basis functions on a 2-D open field environment. The navigation is based on a population vector and the vector direction is then calculated by looking at previous active cells. The direction of navigation is then calculated based on the population vector from goal cells to the agent's current position. The model proposed by Burgess et al. (1994) is a five layer neural network in which the layers are respectively sensory-input hard-wired to the entorhical cells, place cells, subicular cells, and goal cells which represent the location of the reinforcer. It should be noted that they also included the head direction of the agent when it encounters the reinforcer, and this direction is also stored in goal cells layer. The first two layers provide the information from sensory cues, and the main learning takes place by adjusting weights between the two layers of place cells and subicular cells. Whenever the agent encounters a reward location, the specific goal cell (which corresponds to its head direction) receives a reinforcer signal, and the connections between this cell and the ones in previous layers that were recently active

are strengthened.

In Burgess et al. (1994), they showed that the implemented agent could solve the acquisition task using brief exploration in the environment. Also, since the modification in the connections between place and subicular cells was done as the agent was exploring the environment, their agent was able to successfully perform the latent learning task. Later they implemented their model on a mobile robot with visual and odometric sensors which provided input for simulated place cells. The robot was able to successfully navigate in an open field maze (Burgess et al., 1997).

One of the main concerns in the model proposed by Burgess et al. (1994) was the representation of cues, which highly affected the learning performance of their agent. As they mentioned, with four cues present in the environment, the time it took to find the goal dramatically increased. Another constraint was that the agent's later performance was affected if it took a long time to find the goal, and it always takes a long time for the agent to find the goal in the first trials. This happened because the agent might have reached the goal from a longer path. Therefore its head direction and the simulated place cells that were recently active were not necessarily the optimal ones, but were used for learning. Also this model required that the agent looked at all possible directions when encountering the goal, in order to update all goal cells for each direction. If this did not happen, the model predicted that it should affect the navigation performance.

Cognitive mapping with a neural network

Voicu and Schmajuk (2002) proposed a computational model for spatial learning which was composed of four different components:

- A motivation system, which defines goals for the agent. This motivation system is very similar to the idea of the reward system in reinforcement learning. However the slight difference here is that the unexamined locations of the environment were defined as less-valuable goals. In this way, if the agent does not predict any goal locations anywhere in the environment, it is motivated to explore states it has not traversed yet. This modified reward system also keeps the balance between exploration and exploitation.

- An action system, specifying how the agent should choose the next action. This system works by using the goals defined by the motivation system. The agent should move either to a goal state or a state that is predicted to reach a goal state. If none of

them exist, it starts exploring the environment because the motivation system defines unexplored regions as second order goals.

- A cognitive map, storing connections between locations in the environment which is implemented by a neural network. A key assumption here is that initially all states are assumed to be connected to each other, and as the agent explores the environment, the map becomes more accurate.

- A short-term memory component which acts as a buffer and stores the future path that the action system has specified to be taken next.

At the beginning, the agent assumes all states are connected and there is no block. Also the reward location is still unknown, so the motivation system specifies exploratory actions as goals and it starts exploring the environment, while updating its cognitive map to reflect blocked links. As soon as the agent reaches a goal state, the reward gets backpropagated and the connection weights between all states on the path are increased. The change in connection weights is based on the length of the adjacent nodes to the reward.

Voicu and Schmajuk (2002) showed that their simulated rat could successfully model the Tolman detour experiment, shortcut task, and latent learning using this model.

4.2 The Replay Idea in Simple Associative Learning

The replay idea is not limited to spatial navigation. It can be used in a variety of other areas as well. As an example, in this section I briefly explain how we integrated the idea of replay into the existing model of associative learning. We worked on this project with Dr. Elliot Ludvig [1], Prof. Jim Kehoe [2], and Prof. Rich Sutton [3]. The result was a replay model that extends the Rescorla-Wagner (RW) model of associative learning to account for more learning phenomena, which has been hard to explain previously. Our proposed replay model is a computational model for the rehearsal and consolidation of memory. It suggests that new information should be integrated with previous memories and both new and previous experiences should participate in the process of learning.

[1] Princeton Neuroscience Institute, Princeton University, eludvig@princeton.edu
[2] School of Psychology, University of New South Wales, j.kehoe@unsw.edu.au
[3] Department of Computing Science, University of Alberta, rsutton@ualberta.ca

4.2.1 Specification of the Replay Model

In the original RW model (Rescorla and Wagner, 1972), there is an associative strength (V) assigned to each stimulus present in the environment, and the predicted outcome is the total sum of these values defined as $V_{\sum}$ (this is the same as the value function in TD learning which is an estimation for the total reward). This estimation is calculated based on the difference between the received reward (r) and $V_{\sum}$ according to the following rule:

$$\Delta V = \alpha(r - V_{\sum})$$

This value is calculated at the end of each trial, and all associated strengths are updated by this amount.

As an example, we look at the standard blocking procedure. The subjects first receive reinforced conditioned stimulus (CS) paired with an unconditioned stimulus (US), which we show as A+ trials. Then this CS is paired with another new CS, while the reinforcer is still present (AB+). The result is that the subjects completely ignore the presence of this new CS, and do not respond to it, instead they continue responding to the first CS, as before. This can be explained by the RW model, by considering that at the beginning of the A+ phase, the only CS present is A and the term ΔV is positive. Therefore, the associative strength of A (V_A) increases to match the value of r. However, in the next phase, A is a good predictor of the US, therefore there is no ΔV and the associative strength of B does not change, this phenomenon is called blocking.

To extend this model with the replay component, we suggested that the real experience should be used in two ways. First, the agent uses the CS-US pair as in the original RW-model to update the corresponding associative strengths using a TD learning update rule. Second, in addition to this direct update, the CS-US pair is stored in the memory, which represents a world model. After the update rule has been applied to the associative strengths, a certain number of previous CS-US pairs are retrieved from memory and the same learning rule is applied to them again. This essentially means that at each step, the agent is actively learning from both current and past trials.

4.2.2 Behavioral Phenomena Explained by the Replay Model

The simple extension of replay enables the RW model to explain a wide range of experiments. In this section, I explain three main examples: latent inhibition, spontaneous recovery, and retrospective revaluation.

Latent inhibition occurs when animals are first exposed to a CS without a reinforcer being present. Afterwards, when the CS is paired with a reward, the subjects show a decrease in their learning speed compared to the control group which has not been pre-exposed to the CS (Lubow, 1973; Lubow and Moore, 1959). Since there is no reward present during the first phase (inhibition), the ΔV value in the original RW model is 0, and nothing is learned. However with our proposed extension, a world model is also added which stores the A- trials, and in the next phase (A+ trials) presents them again to the agent. Therefore, during the acquisition phase the agent is learning from both reinforced (A+) and unreinforced (A-) trials which slows down the learning.

In the spontaneous recovery procedure, the subject is first presented with a CS paired with a US (acquisition) and then receives CS alone trials (extinction). It has been shown (Napier et al., 1992; Haberlandt et al., 1978) that the animal responds to the CS at the end of the acquisition phase and also stops responding at the end of extinction phase. However, as time passes, the animal gradually starts to respond to the CS again. The degree of this responding is proportional to the duration of time elapsed since the end of the extinction phase. The more time that passes, the more the animal responds to the CS. This behavior is again one of the limitations of the RW model, due to the fact that after the extinction phase, nothing happens to cause a change in the associative strength of A. However by adding the replay component, both the A+ and A- trials get replayed after the extinction phase, which causes the gradual re-acquisition of the associative strength of A.

Retrospective revaluation refers to the behavior whereby a subject changes its response to a CS as a result of being trained with a different CS. The replay model can achieve this behavior because it is constantly retrieving previous trials from memory and updating their associative strengths. Examples of this phenomena are backward blocking, backward condition inhibition, and recovery-after-blocking, which can be explained by our replay model. For instance, in a recovery-after-blocking experiment, subjects are first trained with a standard blocking procedure (A+ followed by AB+ trials), then the blocking stimulus (A) becomes extinct (by receiving A- trials). After this, the subjects gradually start to respond to the blocked stimulus (B) again (Blaisdell et al., 1999). With the original RW model, the second stimulus (B) gets blocked, thus its associative strength remains at 0 the whole time. However, by replaying previous trials, the AB+ trials also get replayed A becomes extinct, which causes the gradual decrease in the associative strength of B.

4.3 Summary

In this chapter, I first described the replay model as an extension to the original RW model of associative learning and briefly explained how it can explain latent inhibition, spontaneous recovery, and recovery after blocking experiments. Next, I reviewed some of the main computational models proposed so far for modeling spatial navigation in animals, and also described some of the behavioral tasks that the models are expected to achieve.

Chapter 5

The Replay Model of Spatial Navigation

> We are constantly altering our memories so the past won't conflict with the present
>
> In treatment, Season 2, Episode 15

This chapter presents the specifications of our replay model for spatial navigation, including its general architecture, description of each of its components, and the learning algorithm. These details will help better understand the behavior of our model in the computational experiments in Chapter 6. In this chapter, I also provide a discussion about the similarities and differences between our replay model and the previous computational models presented in Chapter 4. Our model is based on the Dyna algorithm in RL, a class of sample-based learning algorithms that works by retrieving past experiences and reprocessing them. The retrieval process can be implemented in different ways which I explain here. Their effect in a sample experiment will be evaluated in the next chapter.

5.1 General Architecture

The core idea in our replay model is that learning should take place from both real and imaginary experiences. Therefore, we generate imaginary scenarios from the internal model of the agent and run the same learning algorithm on these retrieved transitions along with applying the learning algorithm to real experiences as generated by interactions between the agent and environment. This idea is based on the Dyna architecture (Sutton and Barto, 1998) (described in Section 2.3.1) which combines the processes of learning, planning, and acting.

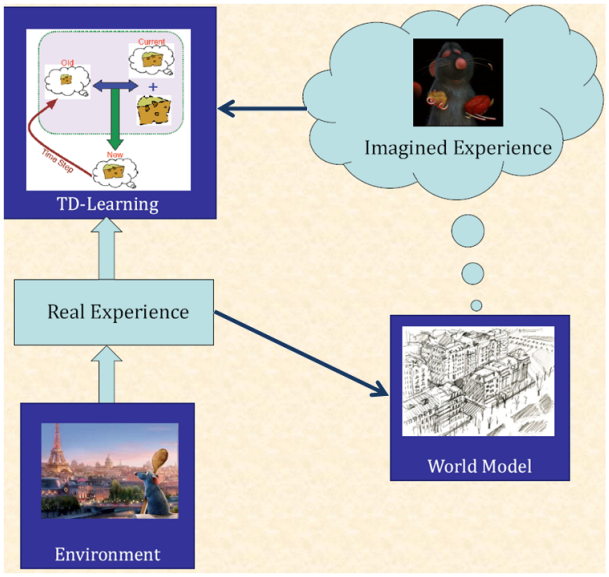


Figure 5.1: Schematic view of the replay model. The agent's interactions with the environment are real experiences, which are used both in the TD learning algorithm and to update a world model. The world model then generates imaginary experiences and they are used in the learning process as well.

A schematic view of the model is shown in Figure 5.1. "Real experience" refers to the agent's interaction with the environment. The TD learning algorithm uses these experiences to learn optimal value functions. On the other hand, the real experience is also used to develop a world model, which generates imaginary trajectories, or as we call them replay steps, for the agent to rehearse and re-learn between each real interaction with the world according to the same TD learning approach. A description of each component of the model is given below.

- **The Environment** provides an outcome every time the agent executes an action. Actions are the only way an agent can interact with its environment. This outcome includes some representation of the next state of the agent in the environment and a single numerical reward signal at each time step (which might be negative as a punishment, positive as a reward, or 0). The states and actions are represented with $s \in S$ and $a \in A(s)$ (where S is the states space and $A(s)$ is the set of all possible actions in state s). In a deterministic environment, each action at a given state will lead to a certain outcome. In a stochastic environment, there is a probability distribution function for each state-action pair which determines the next state.

- **Real Experience** refers to an interaction between the agent and its environment. It consists of four components: current state (s_t), action(a_t), reward signal (r_{t+1}), and

next state (s_{t+1}). This interaction is the only source of information about the environment for the agent.

- The **World Model** is maintained by the agent as it experiences the environment. It keeps some representation of each interaction which has occurred in the past between the agent and environment, and can be used by the agent to predict the environment's properties. The model might be stochastic or deterministic based on the problem. If the environment is changing and the agent forgets the old information, a deterministic model could be useful, in which the first time the agent is faced with the new situation, it replaces the old memories. However, if the environment is noisy, replacing each new event causes the model to change with each inaccurate outcome which will delay learning. Therefore, in this case storing the transition probabilities seems a better option, because the probabilities will reflect the frequency that a specific (s_{t+1}, r) has been observed for each (s, a).

- **Imaginary Experience** has the same four components as the real experience: state, action, reward, and next state. However, the difference is that next state and reward are obtained from the model of the world that the agent maintains instead of the real environment. Learning from imagined experiences is especially useful if there exist some rare but important events, or the ones that are highly costly to obtained. In this case, the world model will remember visiting such experiences for a few times, and these rare or expensive experiences can be regenerated as many times as needed for the agent to learn them. This is similar to a situation in which your body shows an allergic reaction to something you have eaten. Instead of once again eating what you ate that day, you try to remember what might have caused the problem and associate the allergic reaction to that specific ingredient to avoid eating that food in the future.

- A **TD Learning Component** will be used which can be any arbitrary algorithm in general case, chosen based on the specific task. However, we will use TD learning, which is one of the core ideas in RL as described in Section 2.2, and is also strengthened by the experimental evidence that the brain codes similar TD errors. The TD learning algorithm suggests that the current estimate of action values should be updated based on the difference between the observed reward and current estimate.

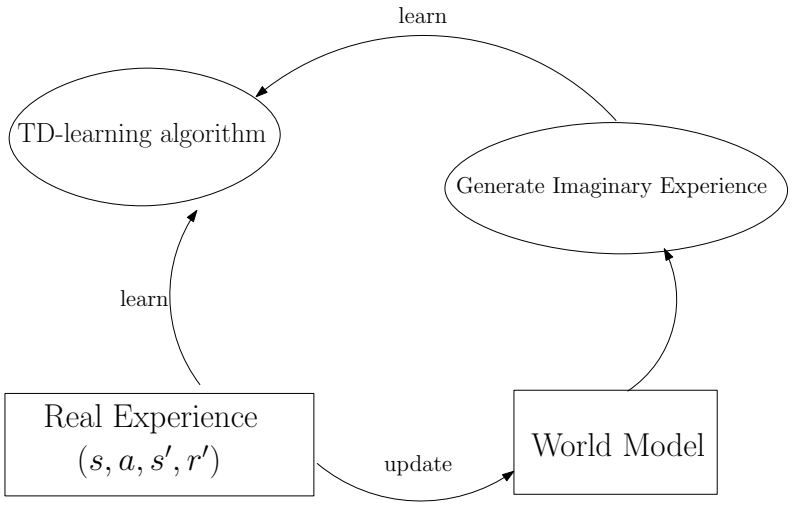


Figure 5.2: General structure of the replay model. Real experience is identified by four elements $(s_t, a_t, r_{t+1}, s_{t+1})$ and is used in two paths: to update the world mode and in the TD learning algorithm. The world model then generates imaginary experiences which are again presented to the learning algorithm.

5.2 Learning Algorithm

Figure 5.2 shows how our model works. One key part of the model is how the imaginary experiences are generated from the world model. The high-level learning algorithm is shown in Algorithm **??** which follows the general Dyna structure (Sutton and Barto, 1998). The algorithm we used in our simulations was SARSA with eligibility traces.

This algorithm mainly consists of two parts for real and imaginary experiences: Lines 4 to 8 show the interactions between environment and the agent. The TD learning algorithm is applied at line 6 to update the current estimate of action-value function(Q). At line 9, s_{replay} is chosen as the initial state for the replay sequence and an imaginary trajectory starts from that state using the current model of the world, which is updated in line 6. In the original version of Dyna-Q algorithm (Sutton and Barto, 1998), a new random state is chosen at each replay step. However, here we choose the initial replay state and then a trajectory is generated starting from it based on the same *policy* the agent is following. This trajectory continues until the replay state reaches the goal state, or N_{replay} replay steps have passed. This is the same for the real trials. The agent is allowed to be in the maze for at most N time steps. If it has not reached the goal state, its location is reset to the initial state.

The model may contain memories of full trajectories of the past. At each replay step, the world model may present one full trajectory to the agent to re-learn it. Alternatively, the agent may store only one-step transitions which gives more freedom in generating replay

Algorithm 4 The learning algorithm in Replay Model

loop
  $s \leftarrow$ initial state
  repeat
    $a \leftarrow policy(s)$
    Take action a, observe r, s'
    $UpdateModel(s, a, r, s')$
    TD-Learning(s,a,r,s')
    $s \leftarrow s'$
    $s_{replay} \leftarrow$ Choose an initial state for replay
    repeat
      $a_{replay} \leftarrow policy_{replay}(s_{replay})$
      $s'_{replay}, r_{replay} \leftarrow Model(s_{replay}, a_{replay})$
      TD-Learning($s_{replay}, a_{replay}, r_{replay}, s'_{replay}$)
      $s_{replay} \leftarrow s'_{replay}$
    until s_{replay} is terminal state $||N_r$ number of replay steps has passed
  until s is terminal state $||N$ number of replay steps has passed
end loop

Algorithm 5 TD-Learning update rule with eligibility traces

$\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$
$e(s, a) \leftarrow e(s, a) + 1$
for all s,a **do**
  $Q(s, a) \leftarrow Q(s, a) + \alpha\delta e(s, a)$
  $e(s, a) \leftarrow \gamma\lambda e(s, a)$
end for

trajectories. In this case, it is possible to generate some imaginary scenarios which have never been previously experienced. For example, the agent might have traversed $s_1 \rightarrow s_2 \rightarrow s_3$ in one episode, and $s_4 \rightarrow s_2 \rightarrow s_5$ in another. But in generating imaginary experiences, the trajectory $s_4 \rightarrow s_2 \rightarrow s_3$ might be generated which has not been directly experienced before. This enables the agent to find potential shortcuts by only replaying past experiences, and is a key feature of the model especially in navigation tasks.

5.3 Relation to previous models

In Chapter 4 I presented a brief overview of previous models that described how the encoded information in place cells can be used to solve different navigational tasks. In this section, I point out the similarities and differences between the key ideas of previous models and our replay model.

Reward-based models lacked the important ability to model latent learning experiments, because learning only occurred after the agent had received the reward. The significance of the model proposed by Burgess et al. (1994) was that the connection weights in the simulated neural network were updated independent of the reward and the agent could learn even without a reward present. Our model has added this feature by maintaining a world model that is updated as the agent explores the environment. This world model is learned even if the agent does not receive a reward and stores information about the transition steps. This enables our agent to successfully perform the latent learning experiment wherein previous exposure to the environment enhances learning.

Foster et al. (2000) also incorporated a TD learning component but added a metric component to the model which enabled the agent to quickly adapt to the change in the reward location after learning the old configuration of the environment. In our replay model, we have a similar TD-component but no special component is added to account for a dynamic reward location task. In such an experiment, our agent will continue replaying the trajectories that led to the old reward location which is no longer reinforced. These replay steps will cause the action-values of states near the old goal state to decrease, which in turn results in more exploratory actions and finding the new reward location. This means that the same TD learning algorithm is again used to adapt to the new environment, without needing to actually execute additional steps which previously caused the models to diverge from the animals' behavior in this task.

The same comparison holds for the model proposed by Balakrishnan et al. as well. They

developed their model based on metric distances to build a cognitive map of the environment and find available shortcuts in it. We implemented the idea of a cognitive map by having the world model, which is maintained as the agent explores the environment. In order to find shortcuts, we again rely on the replay steps which generate imaginary trajectories. Therefore, the agent will use its world model to connect different paths of the environment which have been traversed in separate trials, and find available shortcuts.

Voicu and Schmajuk (2002) had also used the idea of storing a cognitive map in their proposed model. One of the strengths of this model was that the agent could find a shortcut even in an unexperienced open field environment. This was possible because they assumed all states are initially connected unless a blocked path is observed. In our model, there is no initial assumption about states until they are visited for the first time. This enforces the condition that our agent should first visit the whole environment at least once before finding shortcuts.

Another difference is that Voicu and Schmajuk (2002) have distinguished between the two notions of memory and a cognitive map. A cognitive map contains information about the connection between states, while memory is only used as the short-term buffer storing the path that the action system chooses to take. However in our model, we used the term memory as the unit which stores the model of the world which is learned by the agent and represents an abstract world model similar to the role of cognitive map.

Moreover, once the goal is determined, their agent acts greedily and moves to the states with higher value. As a result, the rate of exploration decreases once the goal has been found. Therefore, if there is a change in the environment, the agent might completely ignore it. For example, if a barrier is added to the shortest path to the goal and is removed after a while, the agent learns to avoid the barrier, but does not explore enough to find that it has been removed.

However, since we used the ϵ-greedy action selection method, there is always a small factor of exploration which will eventually lead the agent to discover the new shorter paths. Another difference is that Voicu and Schmajuk (2002) assumed that all parts of the environment are connected (all weights are initialized as one), unless the agent experiences otherwise. Therefore, the agent is always overestimating how good the environment is, while our agent is under-estimating it until it has actually experienced it.

Li et al. (2005) have proposed a slightly different version of the TD learning algorithm by having different reinforcement centers. However, they have maintained a separate reinforcement center for each possible action and at each step chose the "winner" among them.

This was in essence the same as keeping action values and at each state and choosing the best action based on a current estimate of action values. Therefore, our model, which maintains action values and follows the same TD learning algorithm will also behave similarly in the test experiments that were proposed in Li et al. (2005).

The model proposed by Zilli and Hasselmo (2008) used "working memory," to store the next action to be executed, and "episodic memory" to store past trajectories. The idea of storing past memories in a memory model is the same as in our model. However, we did not store the whole trajectories. Instead, only the next state and rewards for each state and action are stored, which allows the agent to generate novel trajectories by connecting different states that were experienced separately. The idea of working memory was also helpful in the spatial alternation task where the reward location changed based on the agent's previous decisions, and caused the problem to lose its Markov property. To be able to model this, we can extend our state representation to include this information, and keep the problem as a MDP. This idea is also similar to the model by Zilli and Hasselmo (2008) in which the animal's current state was determined by its location along with the status of its working and episodic memory.

The most similar model to our replay model is the model of the hippocampus proposed by Johnson and Redish (2005). They were the first to use Dyna model in RL, which already had the idea of learning from imaginary experiences to model reactivation of place cells in the hippocampus. However, they only showed basic behavioral tasks and did not extend the model to account for more complicated tasks such as latent learning or finding shortcuts. Moreover, based on their model they predicted that replay should develop with experience. However, this might not always hold. For example by using the idea of prioritized sweeping in our model, with more experience the prediction error decreases and as a result fewer state-action pairs are added to the priority queue, resulting in fewer replays. As described in the next section, we look more carefully into the replay content and how it is affects learning.

To sum up, the main advantages of the replay model are two-fold: First, using the single idea of experience replay which has a biological basis to account for different behavioral tasks. Second, we use the same TD-based learning rule for both real and imagined scenarios and also in generating these scenarios from an abstract memory model without needing a complex data structure, which is a simpler approach than previous work.

5.4 Retrieval Strategies

A key decision in our model is how trajectories should be chosen for replay, which we refer to as replay content. In this section, I present the different methods that we explored in our simulations.

Frequency-Based Strategy

The first and most intuitive strategy is to replay the states based on the number of times they have been visited. To simulate this, we should maintain the number of times each state has been visited along with the outcome. The replay steps will then mostly include the most frequently visited ones.

For example, in a T-shape maze the agent would be replaying the states in the middle path more often than the other two ends, since it has spent most of its time in this section. This idea is biologically inspired and has been reported in many studies(e.g. (Pavlides and Winson, 1989; Jackson et al., 2006)) where place cells that fired the most during the task were shown to be more likely to continue firing when the task was over.

Recency-Based Strategy

Another idea is to give priority to the states that have been visited more recently. This has also been shown in hippocampal place cells (Wilson and McNaughton, 1994; Skaggs and McNaughton, 1996). Place cells that were active recently, tend to continue firing during the rest states following the task as well. This is specifically useful if the environment is expected to change. In this case the agent will replay the new states more often than the old ones which helps it to adapt more quickly to the change.

To simulate this, we assign a numerical priority to each state. Every time a state is visited, we increase its priority by 1 and multiply the priority of others by a constant factor less than 1. Then we choose the initial replay state from a probability distribution function based on these priorities. These values can be initialized randomly or all be made equal to 0.

Random Strategy

Computationally the simplest way to choose the initial replay state is at random. This is also the same as the original Dyna model (Sutton and Barto, 1998). In this case, there is no priority assigned to states. An imaginary trajectory starts at a state chosen from a random distribution over previously experienced states.

Since there is no specific assumption about the changes in the environment nor which states should be considered more important, this method can be considered as a general approach.

Backward Strategy

This approach suggests that the agent starts the imagined experience from the reward location and chooses one of the states leading to it, then sets its current state to this one and continues replaying in a backward direction. The interesting point in this approach is that sometimes in the original experiment, the agent is only allowed to move forward. Thus, in order to move backward from a sample state S, it has to search the model to find which states have led to S. It then chooses one of these precedent states (S') and executes the action taking it from S' to S. This process is then repeated for S' until it is equal to the maze's initial state, or a certain number of replay steps has passed. This behavior was reported in the experiments done on rats as well (Foster and Wilson, 2006; Gupta et al., 2010). Although rats were not allowed to move in the opposite direction in the maze, this backward reactivation of hippocampal place cells was observed in their place cells.

Most-Rewarding Strategy

Another variation is to replay the states that were more *valuable* than others. More formally, this strategy is equivalent to choosing to replay the state with the highest value function. However, this method will eventually be similar to the replay-from-the-goal-state. Therefore it makes sense to simulate the trajectories backward, starting from the state with highest value function and moving backward to the states that are known to be leading to it, and so on.

Prioritized Sweeping Strategy

Prioritized sweeping is suggested as one of the strategies for choosing replay trajectories in a Dyna algorithm (Moore and Atkeson, 1993; Lin, 1992), and was described in section 2.3.3. The idea was to replay the experiences that were considered to be more *surprising*. More specifically, the agent maintains the latest value of all prediction errors, and in each trial chooses to start replaying from the state with the largest error. The agent continues simulating the imagined experience from that state until it reaches the goal, or a maximum number steps have passed. In this way, if the environment changes, the agent is more often replaying the parts where the experience has been contradicting its model of the

world. Therefore, it adapts to the new environment more quickly.

5.5 Summary

In this section I presented our replay model, which extends previous TD learning models by allowing the agent to generate experiences from its model of the world and re-learn from them again and again. I then compared our model with previous computational models which tried to simulate behavioral tasks performed by animals in a navigational task. As mentioned, the main advantage of our model was its simplicity in using same learning algorithm for both real and imaginary scenarios, along with its accordance with biological evidences for replay patterns recorded from place cells. Finally, I presented different approaches for generating imaginary scenarios that are inspired by the observed patterns of replays recorded from rats' hippocampal cells.

Chapter 6

Experiments and Evaluations

This chapter presents how we evaluated our model, and why we consider it as a powerful computational model of spatial navigation that uses a simple idea, and can explain many behavioral tasks, while being supported by neurobiological findings.

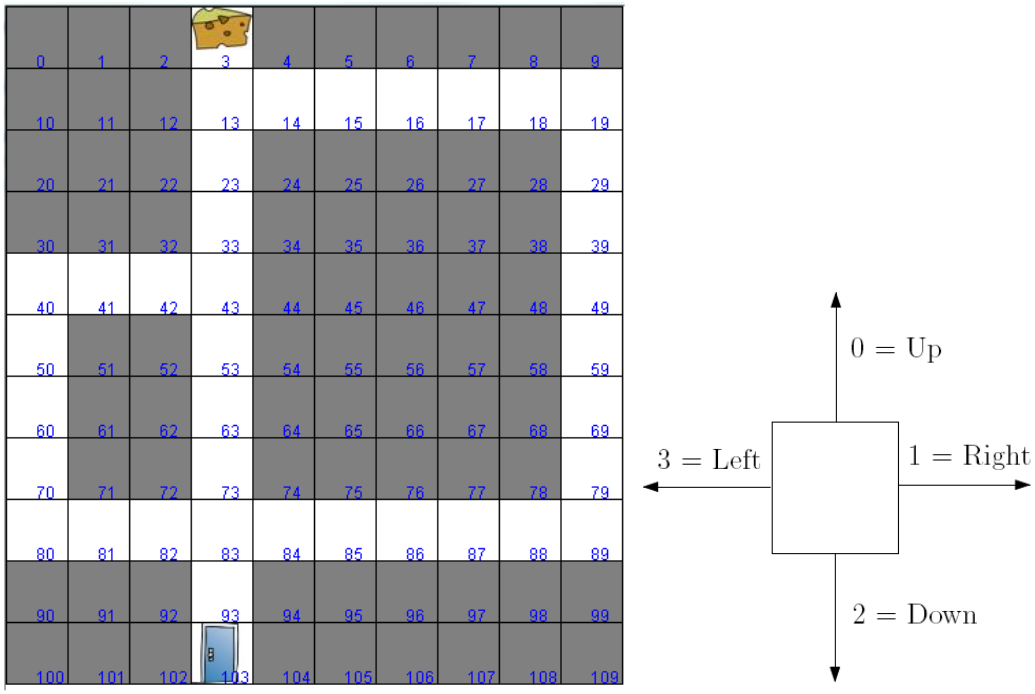
I first present the environment we used for our computational experiments, and explain the we will use throughout the rest of this chapter. Then I demonstrate the behavior of our replay model in two learning tasks including the latent learning and the Tolman detour experiment. These computational experiments are designed to show certain properties of the model and how adding the simple idea of experience replay enables the agent to solve different maze configurations that were considered complex for computational models. For each of these experiments I also provide a discussion on how previous models can or cannot achieve the same results.

I will then compare the performance of our replay model using various strategies for retrieving past experiences and illustrate how recent findings from rats' brains support our model and these replay schemes we suggest.

These three approaches (performance in behavioral tasks, being in accordance with neurobiological data, and having advantages over existing models) form the main evaluation criteria for our model.

6.1 Experiment Settings

The environment that was used for all experiments in this chapter is a maze domain with a start state (shown in the figures with a door) and a final state with a reward value of +5. We formalized the navigation task as a deterministic, finite MDP. The representation of states and actions is shown in Figure 6.1. Each state represents the approximate location of the agent in the maze and is represented by an integer corresponding to its index (Figure 6.1a).



(a) States (b) Actions

Figure 6.1: (a) The Tolman maze configuration with states represented by indexes. The door shows the initial state and the piece of cheese represents a reward at the goal state. Shaded areas are blocked. (b) Actions.

	0	1	2	3
⋮	⋮	⋮	⋮	⋮
82	82	83	82	81
83	73	84	93	82
84	84	85	84	83
85	85	86	85	84
⋮	⋮	⋮	⋮	⋮

Table 6.1: Matrix representation of next states in agent's world model

	0	1	2	3
⋮	⋮	⋮	⋮	⋮
13	5	0	0	0
14	0	0	0	0
15	0	0	0	0
⋮	⋮	⋮	⋮	⋮

Table 6.2: Matrix representation of rewards in agent's world model

Actions can have four possible values, as shown in Figure 6.1b. The initial location is shown by a door, the reward is shown by a piece of cheese, and grey areas represent blocks. The environment is deterministic, meaning that each action will move the agent one step closer to that direction, unless there is a block, in which case nothing happens and the agent's state remains unchanged. The reward is set to +5 for the case where the agent reaches the goal state (state 3 in Figure 6.1a) and 0 otherwise.

The agent maintains a model of the world as it experiences the environment. The model is represented by two matrices, one for remembering next states and one for remembering received rewards. Each row of the matrix shows the state index and the columns correspond to the four possible actions. At each step, the agent chooses an action and receives the next state and reward, and updates the corresponding matrices to the values just received.

Table 6.1 shows a section of the matrix storing next states corresponding to Figure 6.1a. Each action takes the agent one step in that direction, unless the new state is blocked in which case the state will remain the same. A one-way path or a door in the environment, will also be represented as blocks which do not change the agent's state.

The reward table is shown in Table 6.2. In the environment shown in Figure 6.1a there is only one reward with value of 5 at state 3, which the agent only receives by going up at state 13. As shown, all other reward values are set to 0 and there is no punishment for hitting the walls or blocks.

Parameter Name	Description	Value
α	learning rate	0.1
α_{replay}	learning rate in replay steps	0.05
λ	eligibility traces decay rate	0.94
γ	discount factor	0.97
ϵ	exploration rate in the policy	0.1
ϵ_{replay}	exploration rate in policy for replays	0.4
N	Maximum number of time steps	300
N_{replay}	Maximum number of replay steps	10

Table 6.3: Summary of parameters used in our simulations

Our learning algorithm was shown in algorithm **??**. We used the ϵ-greedy action selection method, where $\epsilon = 0.1$ for real experiences (in line 5) and $\epsilon = 0.4$ for imaginary experiences. This means that the agent is taking more exploratory steps in replays, since we want it to explore more parts of the environment that are expected to lead to less return in real experience. Moreover, learning from real interactions with the environment should have more priority than imaginary steps. So we used different learning rates for the actual and imaginary steps. A summary of the parameters used in our simulations is presented in table 6.3.

It is worthwhile distinguishing the different terms we use in this chapter: step, trial, run, and experiment:

- Each step is defined as taking an action and receiving its outcome, whether this action changes the agent's state or not. The word "step" is also equivalent to a time step, since at each time step the agent is taking exactly one action.
- A trial starts by placing the agent in the initial state, and ends when it reaches the goal state or has been in the maze for a certain number of steps (which we set to 300).
- A run consists of placing the agent in the maze for 100 trials and looking at the number of steps until it finds the goal at each trial.
- Each experiment is performed for 100 runs, and the results are averaged.

The computational model of the hippocampus proposed Foster et al. (2000) used the same TD learning update rule, but also assumed that place cells store the metric coordinates of the agent.Foster et al. (2000) discussed that when the reward position changes daily and rats show one-trial learning, they are apparently using their knowledge from previous trials. In order to use this knowledge, Foster et al. (2000) developed the metric coordinate system

and associate place cells to their coordinates, which adds the complexity of finding the origin each day. Compared to this model, we show that there is no need for the extra metric component, and simply re-learning from past experiences is sufficient to solve problem.

As previously mentioned, the most similar model to ours was proposed by Johnson and Redish (2005). They used the Dyna-Q algorithm, which uses previously experienced sequences to speed up the learning. They suggested that replaying past sequences is strengthened by experimental data from place cell reactivation, but only used this idea on a simple acquisition task. We extend this model in two different paths. First, we show how the replay idea can enable the TD learning models to solve the Tolman detour task in one trial. Second, we evaluate different strategies for generating replay sequences, while Johnson and Redish (2005) only considered random sampling from previous trajectories.

6.2 Performance Criteria

The four methods we use in the rest of this chapter to illustrate the behavior and performance of our agent are described below:

- **Action values**

  After the agent has completed learning a maze, the action values should be close to optimal and reflect this learning. We show the maximum action value of all states by a color ranging from white to black. Larger values have a darker color, so we expect the states that are closer to the goal to have a darker color, and this color should reflect the distance between each state and the reward location.

- **Average number of steps to goal**

  To eliminate the effect of the agent's random behavior and obtain more accurate results, we ran each experiment for 100 runs. In each run the agent was allowed to run through the maze for a maximum of 300 trials. We then averaged the number of steps until the agent reached the final state over the 100 runs. In each diagram the confidence interval is shown below and above the average line, which shows the standard error of the data. We expect that at the first episodes the agent takes longer to find the goal. This should gradually decrease to eventually reach the optimal solution (i.e. the length of the shortest path from initial state to the goal).

- **Average number of steps to learn the optimal path**

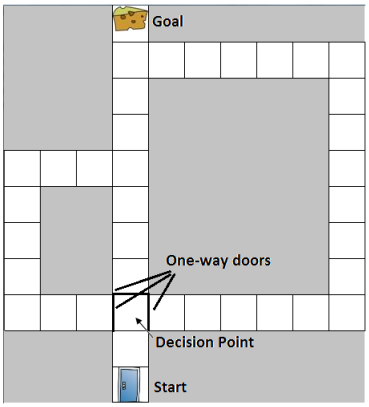


Figure 6.2: Simulated maze configuration of the Tolman maze. The agent starts at the initial state shown by a door and runs through the maze to find the reward (shown as a piece of cheese) at the goal state. Once it passes one of three one-way doors at the decision point, it cannot go back.

In some experiments, the environment changes and the agent has to learn the new optimal path in only one trial after observing the change. Therefore, to evaluate the efficiency of learning we look at the number of steps (in one trial) that the agent takes in order to correctly update all value functions so that the optimal path is found.

- **Action values of the decision point**

  In some experiments, there exists a decision point where the agent chooses to take a critical action. In such experiments, to show exactly how learning has occurred, we provide the action values of each action at this decision point before and after the test run of the experiment to be able to exactly show how learning has occurred.

6.3 Exploration and Goal Finding

In the first experiment we show two main behaviors of our model: (i) Adding replay steps in each trial improves the learning rate, and (ii) the more experience the agent has, the faster it learns.

We simulated the maze proposed by Tolman and Honzik (1930). In this experiment, Tolman and Honzik (1930) placed several groups of rats in a maze similar to that in Figure 6.2. They then let them find the shortest path from their start location to a goal, which was a food reward. In the original experiment, the decision point is surrounded by three one-way doors, so that once the animal has selected which path to choose, it cannot go back and change its decision. We used the same setting for our simulation, so that once the agent enters one of the three paths, the backward path is blocked.

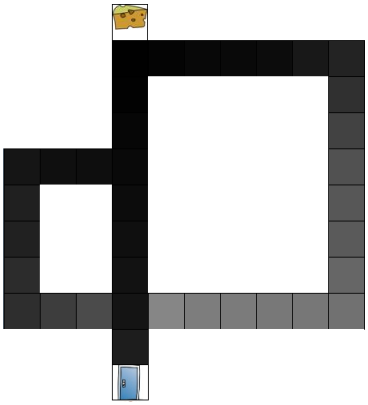


Figure 6.3: Illustration of the result of the replay model on a simple acquisition task. Colors show the maximum action value at each state after the agent has learned all three paths. Darker colors show higher value. Comparing the relative darkness of each path shows that the agent has learned the correct ranking based on each path's length.

We ran two different agents, one using our model, and one using only Sarsa (without replay steps) and compared the learning rate between them in Figure 6.4. Each trial starts with putting the agent in the initial state and letting it find the goal, or until a certain number of trials (which was set to 100 here) has passed. We repeated this for 100 runs (each run included 100 episodes) and the results show the average number of trials (over 100 runs) to the goal at each episode. The standard deviation of these values is also shown by the error lines below and above the average line.

Figure 6.3 shows the maximum action value (Q(s,a)) for each state after the agent using the replay model has completed its learning (100 episodes). As the figure shows, the values are higher near the goal and gradually decrease as we move away from the goal. Since we are using the ϵ-greedy action selection, we can guarantee that with enough trials all three paths are explored enough and the action values reach their optimal values.

Discussion

In this classical experiment, Dyna is used to speed up learning. The number of trials that the TD model needs to properly learn the maze is usually more than what animals need (Johnson and Redish, 2005). One way to improve the learning efficiency is to add replay steps as suggested in the Dyna algorithm. Comparing the two diagrams in Figure 6.2 shows that the number of trials necessary to update all action values and find the optimal path has considerably reduced in the case where the agent is allowed to replay.

Moreover, Figure 6.2 shows that, much like animal behavior, the number of steps the agent takes decreases over time until it reaches the optimum, which is 10 for the shortest

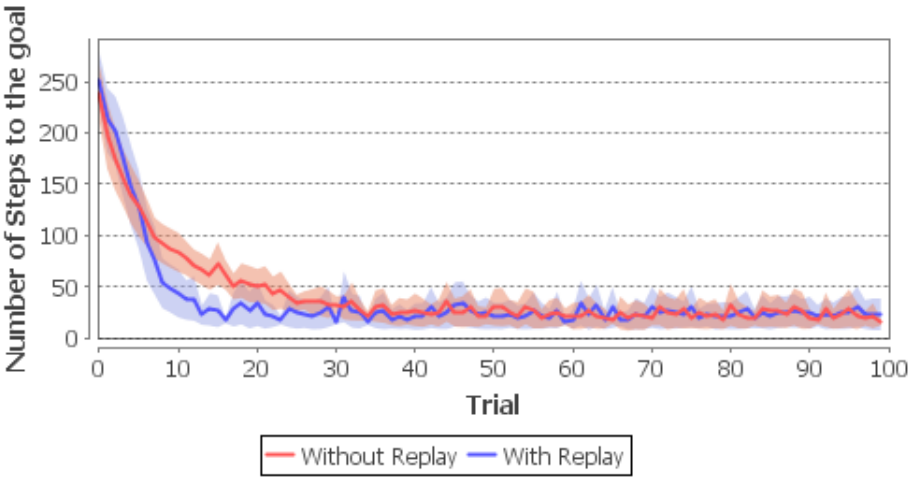


Figure 6.4: Performance of the replay model on the Tolman maze with and without having replay steps. Shown is the average number of trials to the goal on the Tolman maze shown in Figure 6.2. It shows that having replays speed up the learning process. The shaded area represents the standard error.

path.

6.4 Latent Learning

The goal in this experiment is to show that pre-exposure to the environment even without a reward being present affects the learning, as seen in the experiments on rats (Tolman and Honzik, 1930). For this experiment, we simulated a latent learning experiment proposed by Tolman (1948). Tolman used the term *Latent Learning* to describe the behavior when an animal does not show any signs of learning until after the reward is presented. The overall procedure is to let the agent first run through several trials where there is no reward present, and then present the reward.

6.4.1 Simulation results of our model

The parameters and the algorithm used are similar to those used in the previous section, except for the reward which is not presented right away for the test groups. We have three groups here: The control group (Group 0) receives the reward from the first trial and the two test groups receive the reward on 10th and 20th trials respectively (we call them Group 10 and Group 20). Figure 6.5 shows the average number of trials for 100 runs in each group. Group 0 shows a gradual decrease in the number of steps to the goal, and finds the optimal path after about 30 steps. Group 10 does not show any sign of learning before the reward

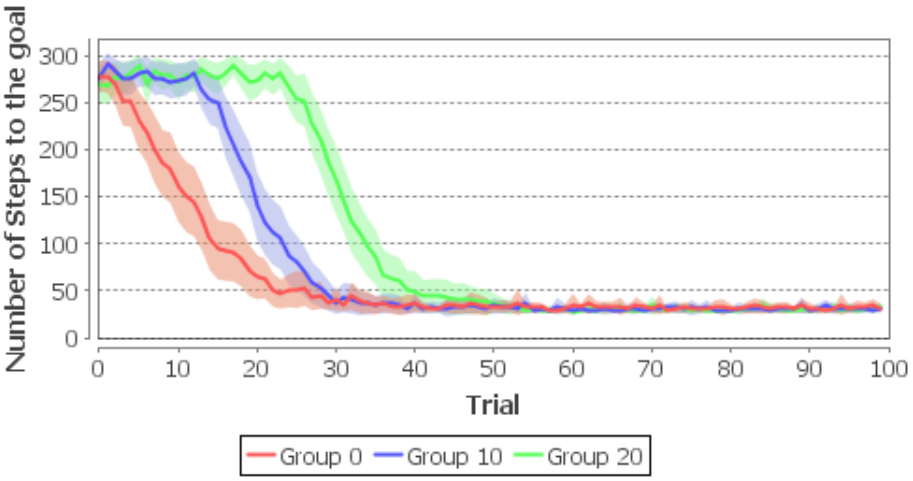


Figure 6.5: Performance of the replay model on a simulated version of the Tolman latent learning experiment. Group 0, 10, and 20 receive the reward at the first, 10th, and 20th trial respectively. The slope of the diagram increases as the agent has more time to develop the model before presenting the reward.

is presented in the 10th trial. However, once the reward is added, the number of steps to the goal quickly decreases to reach the optimal value. This takes about 15 steps. Similarly, the third group (Group 20) does not show any learning before the 20th trial, but shows a fast decrease in the number of steps to the goal and can find the optimal path in the 30th trial (after 10 trials from receiving the reward.) In order to compare the rate of learning for these groups, we ran a paired-t test on the slope of diagrams, and the results showed that the slopes of Group 10 and Group 20 are significantly more than the slope of Group 0. However the slopes of Group 10 and 20 were statistically similar. This shows that the world model is completely learned before the 10th trial, and the more trials the agent gets after learning the model does not affect the learning.

Discussion

When there is no reward present, all action values remain at 0. In this phase, the agent is only exploring the environment randomly. No sign of learning can be observed. However, the agent is learning the world model, which is represented by the transition matrix. Therefore, the more training the agent has before receiving the reward, the more it explores the environment and the more complete its world model becomes. Once the reward is added and the agent finds its location, it starts updating the value functions. Meanwhile, the agent is replaying previous experiences. The replay steps are retrieved from its world model but now have the reward at the goal state as well. Therefore, in the replay steps the agent recalls

all the paths it has explored before, and follows them to the goal, while updating their action values. Based on this explanation, there are two key aspects of the model responsible for showing the latent learning behavior:

- The world model, storing the transitions between states, which is being developed even before the reward is present.
- The replay steps after the reward is presented, which enable the agent to quickly update action values of all state-action pairs which are generated by the model learned in previous step.

Once all the transitions between states in the model are learned, additional training will not affect the agent's world model, so having Group 30 or more will not further speed up the learning.

6.4.2 Comparison to previous computational models

The model proposed by Zilli and Hasselmo (2008) focused on incorporating the role of memory in the model and using it to solve the tasks which involved a delay conditioning. The main learning algorithm was a TD algorithm with eligibility traces but the states also included the state of working and episodic memory as well. Therefore, the learning was driven only if the reward was present, so the model by Zilli and Hasselmo (2008) could not perform the latent learning task.

Burgess et al. (1994) modeled place cells by radial basis functions and used a five layer network. The connection weights of this network did not depend on the reward; they were updated as the agent explored the environment. Therefore, Burgess et al. (1994) could achieve the latent learning experiment on an open field environment. However, their model highly depended on the cue representation (the first two layers of the network mapped the input data from sensory cue to place fields), and the agent could not navigate through a maze with narrow pathways. Moreover, the goal cells also contained information about the agents head direction, which required the simulated rat to turn and explore all directions once reaching to the goal, and update their associated weights.

Voicu and Schmajuk (2002) proposed their model based on building a cognitive map which stored connections between states, and a motivation system which defined exploring undiscovered areas of the map as having a small reward. Therefore the agent could complete its world model even without an actual reward present, and could perform the latent learning task. However, the idea of adding an artificial reward signal to encourage

exploration does not have a strong biological basis. On the other hand, in our model the agent only explores the environment to find the goal location and does not require this extra assumption. Besides, they forced the agent to explore each path at least 10 times in the pre-exposure phase, while our model does this automatically due to the $\epsilon - greedy$ action selection policy.

6.5 Insight and Reasoning

This experiment was proposed by Tolman and Honzik (1930) to show evidence of insight and reasoning in rats, which then became a measure for computational models aiming to mimic animal behavior. The experiment procedure is as follows. The maze configuration is similar to Figure 6.2. As shown on the figure, there is a decision point with three one way doors, so that once the agent chooses a path it cannot change it.

The subjects have enough time to explore all three different paths, and eventually learn to quickly choose the middle (shortest) path. Then a blockade is added, as we show in Figure 6.6. After the subjects encounters this blockade, they learn not only to avoid it, but also to avoid the left path as well, although they never directly experienced that it was also blocked. Moreover, this update in their decision making occurred at the very next trial, having experienced the blockade only once. Therefore, this experiment suggests that a reasoning process is taking place in rats' brain which cannot be explained with classical conditioning learning theories.

6.5.1 Detour

We simulated this experiment with our model, and Figure 6.7 shows a maximum of action values both before and after the agent experiences the blockade. It can be seen that the values of both blocked paths has decreased while the values of states in the right path are unchanged.

We also presented the action values of the decision point before and after the trial where the rat faces the blockade in Figure 6.8. It shows that after the training phase, the order of choosing actions is respectively up, left, right, and down based on the action values. However, after the blockade is added and the agent has had enough time for replay, the values of going left and up have decreased and going right has now become the best action.

It should be mentioned that action value of going down is always the second best option (both before and after adding the blockade). This is because of the three one-way doors which prevent the agent from changing its decision once it chooses right, left, or up. But

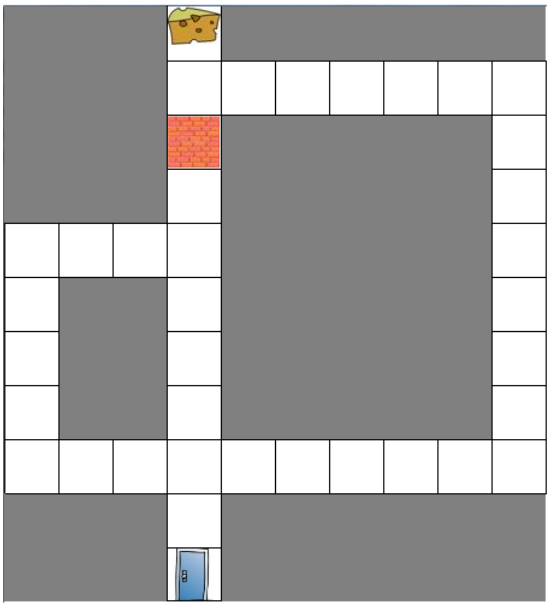
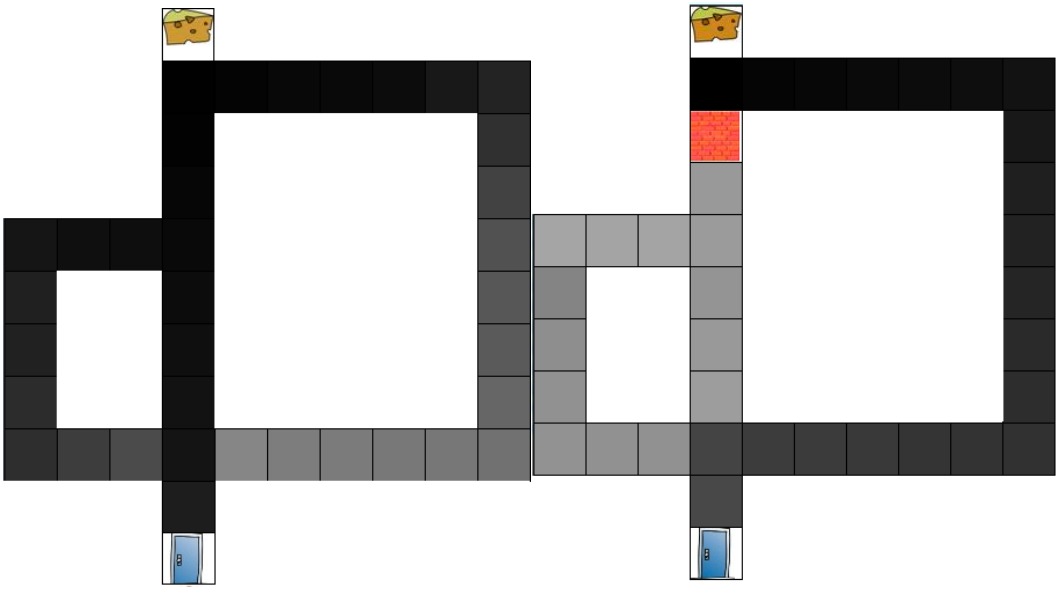


Figure 6.6: Simulated maze configuration for the Tolman detour experiment: The wall indicates the obstacle added to the common section of middle and left paths after the agent has explored all three paths sufficiently.



(a) (b)

Figure 6.7: Illustrating how the replay model reproduces the main effect of the Tolman insight experiment. Shown is them maximum action value at each state (a) before and (b) after adding the blockade for one trial. Darker colors show higher value. Decreased values in (b) show that the values of both blocked paths have decreased although the agent only experienced the newly added obstacle while being in the middle path.

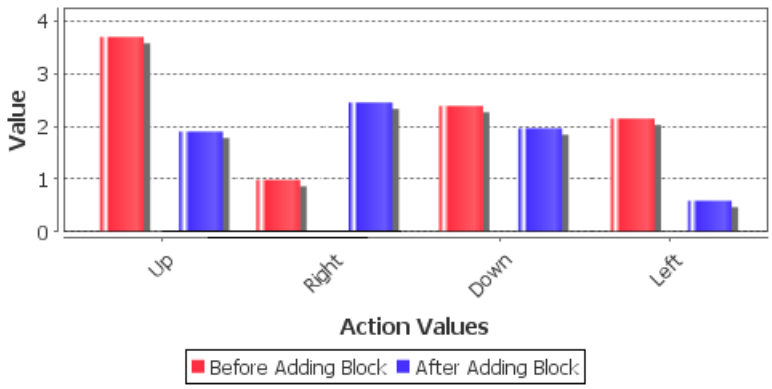


Figure 6.8: Showing how the replay model can correctly show the insight behavior in rats. Shown are the action values at the decision point before and after adding the blockade. Before the blockade is added going up is the best action, while after experiencing the obstacle for one trial its action value decreases and going right becomes the best action.

going down is not an irreversible choice, and only delays the reward for two steps. Therefore, going down is always the second best option for the agent. Moreover, the action value of going down from the decision point depends on the value of the best action, and therefore is decreased after adding the blockade.

Discussion

The model is updated when the agent faces the blockade and is then stuck between it and the one-way door, so that it cannot explore other parts of the maze. However, it continues to retrieve trajectories from its updated world model and reprocess them. This includes reprocessing paths leading to the blockade, as if it is experiencing it again, so it will result in a decrease in the values of both blocked paths. In this experiment, replay is an essential part. Without replay steps, the agent would only update the action values of the path which it is currently in, and could not generalize it to the other path. Replay steps here provide a mechanism for reasoning and remembering past experiences to integrate them with the new information.

6.5.2 Comparison to previous computational models

As previously mentioned, the model proposed by Zilli and Hasselmo (2008) included the working and episodic memory in order to explain delay conditioning tasks. The update rule was TD, which needs more than one trial to converge to the optimal solution. Therefore, Zilli and Hasselmo (2008) could not achieve the one-trial learning observed in rats.

The computational model proposed by Burgess et al. (1994) has two main limitations in explaining this experiment. First, as mentioned earlier, the model cannot be used in a maze environment with narrow pathways. The radial basis functions provide a mechanism for the agent to find the shortest path, but do not work well if there are walls and blocks in the path. Second, one trial is not sufficient for all the necessary connections to be updated.

The model proposed by Voicu and Schmajuk (2002) is capable of solving this detour problem by noticing that the world model is stored as a canvas. Thus, when the agent encounters the blockade, this changes all the connections weights of nodes leading to it in the network. Therefore, the update process takes place at the same time for all the nodes. However, this is slightly different from experimental data that showed that the place cells which are spatially close to each other fire at the same time. Therefore, only the connection weights of one trajectory are updated each time. That is the inspiration behind our model which updates the weights by replaying each sequence separately, and not all value functions at the same time.

6.6 Replay Content

The question we focus on in this part is the content of replay, how it is related to experience, how it affects learning efficiency, and how patterns observed in empirical data can be generated by our model. So far, replay scenarios have been chosen randomly with a uniform distribution over all states previously experienced, which have thus been incorporated into the world model. However, experimental results (Gupta et al., 2010) suggest that the content of replayed trajectories is not always randomly chosen from past experiences. Gupta et al. (2010) tested the subjects on a maze which had two loops. The rats received the reward only if they alternated between these two paths and did not choose the same direction at each trial. By looking at the recorded replay patterns of the rats' brains both at the pause intervals during the task and at rest intervals after it, Gupta et al. (2010) found three main patterns that had not been reported in any of the studies that we briefly repeat here:

- The replay trajectories sometimes started from the animal's current location and moved forward toward the goal, and sometimes started from goal locations and continued backward, even if the animals had only experienced the maze in one direction.
- Although the rats learned to alternate between the left and right paths, their replay patterns included trajectories of the other side of the maze, although they had not experienced it recently.

- There was a shortcut between the two reward locations of the maze which the rats never experienced because each time they reached at the goal locations they were picked up and placed at the initial state of the maze again. However, the replay patterns included this shortcut path which was a novel trajectory.

Previous studies had observed different patterns:

- The first evidence of replays in the hippocampus were associated with the frequency that each of these paths had fired during the experiment (Pavlides and Winson, 1989; Jackson et al., 2006). This suggests that the start state of replays can be set to be the one that had been visited the most.

- Singer and Frank (2009) reported that the paths that were associated with a reward were replayed more often than the other paths. This suggests that we choose the one with higher value as the start state of replay trajectories and move backward.

None of the previous computational models of the hippocampus included various replay patterns except for the TD learning model of the hippocampus proposed by Johnson and Redish (2005). However, they only considered random sampling from previous trajectories and did not consider other strategies for choosing replay states.

In this section we evaluate different replay schemes and show their effect on learning. We use the same setting as the Tolman detour task as in previous section, and compare the number of steps until the correct values at the decision state have been learned (by correct value we mean that the value of going right is higher than other actions). The only difference between an imaginary and real experience is that we set the $\epsilon = 0.4$ for replay steps and $\epsilon = 0.1$ for real experience. This is because real experiences might be costly to obtain, so it would be wise to exploit more, and let the imaginary experiences have a more exploration rate to choose rare samples that have not yet been experienced often.

We compared different strategies for choosing replay content in this part. First, the agent starts its imagined experiences from its current location and replays forward until it meets the goal or a maximum number of 30 replay steps have passed. More formally, in each trial, after the agent executes an action and observes the next state and the immediate reward, it starts simulating a path from its internal model of the world. This path starts from its current location and it simulates next actions and rewards similar to the real experiences until it reaches the goal. This pattern was seen in actual recordings from reactivation of hippocampal place cells during rest states (Gupta et al., 2010) and was thought to be the main pattern of replay.

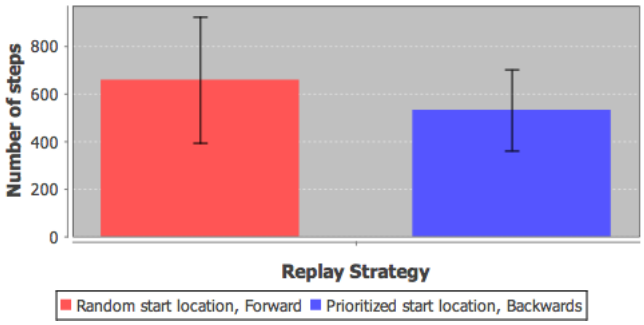


Figure 6.9: Evaluating the performance of our replay model with different retrieval strategies. Figure shows the number of steps until correct path is found in the Tolman detour task with forward replays starting from a random state and backward replays with prioritized sweeping

The second option would be to choose the states that have been visited the most. These are the states in the shortest path, which is now blocked. By starting a trajectory from this location and moving either backward or forward, the agent will not be able to replay the values of left path, so this strategy is not helpful in this case.

Another option is to start the imagined experience from the reward location and move backward. What is interesting about this situation is that the agent has only experienced the environment in one direction. So in order to move backward from a sample state s, the agent has to search the model to find which states have led to s. It then chooses one of them (s') and executes the action (a'), taking it from s' to s. This process is then repeated for s' until it is equal to the maze's start location, or 30 replay steps has passed. This behavior was reported in the experiments done on rats as well (Gupta et al., 2010). Although rats were not allowed to move in the opposite direction in the maze, they could observe this backward reactivation of hippocampal place cells. In this case, learning the correct values is equivalent to updating the value function of both the left and middle paths. Both paths are blocked and do not lead to the reward location. Therefore, regardless of how many times the agent replays from the reward location, it cannot update the values of these two paths. So this strategy is not suitable for this case.

However, we can use the prioritized sweeping strategy instead. Prioritized sweeping is suggested as one of the strategies for sampling in the Dyna algorithm (Moore and Atkeson, 1993; Lin, 1992). The idea is to start replays from the states that were considered to be more *surprising*. More formally, the agent maintains the latest value of all prediction errors, and in each trial chooses to start replaying from the largest one. It continues simulating

imagined experiences from that state until it reaches the goal, or a maximum number of 30 steps have passed. In this way, if the environment changes, the agent is more frequently retrieving the past experiences that have been in contradiction to its world model.

We compared these two strategies on the Tolman detour task. The results are in Figure 6.9 which shows that prioritized sweeping in this maze environment has led to faster learning, because it directs the replays towards the states that need more learning.

6.7 Summary and Conclusion

In this chapter we provided our implementation details and our environment settings that we used to evaluate the performance of our replay model in two behavioral tasks including the Tolman detour task, and latent learning. We then discussed the advantages of our replay model over previous computational models of spatial navigation. Previous models (presented in Chapter 4) had two main limitations: Some lacked a sufficient neurobiological basis, and relied on computational methods that made them rather complicated and reduced their robustness in explaining new behavioral experiments. Others were based on the evidence in rats' brains, but could not explain some of the complex phenomena such as latent learning or detour. In contrast, in this section we illustrated how our model uses the simple and intuitive idea of replay, which is also in accordance with empirical data, and can explain the behavioral tasks mentioned.

Chapter 7

Conclusion and Future Directions

In this thesis we proposed a computational model for spatial navigation based on sample-based RL algorithms. Our model was inspired by the experience replay seen in hippocampal neurons. We reviewed some recent experiments that showed how neurons that were active during a learning task tend to continue firing when the animal is sleeping or even during the rest intervals between trials. This inspired us to use a similar replay idea proposed in Dyna to model spatial navigation. Our model worked by keeping previous experiences in a world model and using them to generate imaginary experiences. This process of generating imaginary sequences was then continued, and learning conducted on both real and imaginary experiences.

The highlights of our work can be summarized as follows:

- We established experience replay as an idea in RL with promising neurobiological evidence that can bridge the gap between AI and neuroscience.
- We showed, through simulations, that adding the experience replay extension to sample-based learning algorithms can explain many previously hard-to-explain behavioral navigational experiments such as latent learning or insight experiments.
- Our computational model was flexible in terms of parameter settings, and the behavioral results we presented did not depend on specific parameters.
- We systematically investigated several replaying schemes and assessed them both computationally and according to existing empirical evidence.
- We simulated different replay schemes observed in recent findings, which have not been considered in any previous computational model of navigation.

- Our model re-used the same learning algorithm on both real and imaginary experiences, which enhanced its simplicity and made it a more promising model for explaining the learning process in the brain.

- Learning from imagination can be an inspiration to other RL algorithms trying to model animal behavior. Its computational benefit was proved previously, and in thesis we presented potential neural and behavioral evidence for it.

Based on our model, we make several testable predictions that can be justified through experiments, and also propose potential paths for future computational models of spatial navigation:

- Relationship between experience frequency and replay

  Previous theories suggested that the re-firing of place cells in the hippocampus will increase with experience, and considered these replays as a passive function of previous experiments. However, we suggest that these replays play an active role in learning a behavioral task. For example, the idea of prioritized sweeping suggests that the trajectories which need more learning are replayed more often. Therefore, observing the patterns of replay during the training trials of subjects on a T-shaped maze can verify whether they are only replaying the most frequent sequence (main stem) or the decision point (which is more important in the learning process).

- Effect of small changes in the environment

  Another experiment is to present a small (which can also be negative) reward in another location in the maze, after the agent has completely learned the environment. Such a small surprising event will not change the agent's behavior, but will make the agent think more about it. The result is a small prediction error around that particular state in the maze, while there is almost zero error in all other locations because the learning has been completed. In this case, if the patterns of replay follow a prioritized sweeping strategy, they should occur around this new reward location. However, if the agent is replaying the most rewarding parts of the maze, this change should not affect its replay content since the most rewarding part is still the old higher reward value.

  This small reward should have two features: First, it should only be presented for a limited time and not end the trial. Otherwise the agent will spend the rest of the trial at this location and be unwilling to move on to the main reward. Second, the reward

value should be determined in such a way that during the steps from this location to the main reward, its discounted value goes below the threshold which is set to 0.001 for our experiments.

This procedure can be verified by adding a small surprising event (e.g., a shock) in the other loop of the maze. The rat does not normally take this path, but will eventually experience it because of its exploratory actions. Recording hippocampal place cells firing during the pause periods after receiving the final reward can verify whether the rat is imagining that small event, or imagining the most rewarding parts of the maze.

This experiment allows us to suggest that if we come across an unexpected event which is not as important as other parts of our usual daily patterns, we do not need to repeat that action and receive the same unpleasant reward. It is sufficient to think about it until the event and its consequences are no longer considered surprising and unexpected.

- Hippocampal Lesions

  As described previously, replay is the main reason that our agent updates its world model correctly and learns to avoid both blocked paths in the detour task (section 6.5.1) without explicitly experiencing them. If the agent is not allowed to replay, we predict that it cannot learn the correct path and quickly avoids both blocked paths. This hypothesis can be verified in two ways. First, causing a temporary and reversible lesion in the hippocampus after a learning task, will prevent the animal from replaying. By removing this lesion the next day, the animal will be able to navigate through the maze again, without having done any replay steps. There are two drawbacks to this method: causing damage to the hippocampus, even if it is a reversible damage, is ill-advised and could permanently damage some place cells. Therefore, not being able to solve the maze does not imply the necessity of replay steps. Besides, the agent still can perform some replay steps during the task, and causing the hippocampal lesion does not prevent all replays.

  The second approach is to prevent the animals from experience replay by keeping them busy in another task that needs constant attention. This method is more feasible and also can be done on humans as well as other animals. The procedure should begin by letting the subjects experience the main experiment, and then presenting another competing task that requires their active attention. This will eliminate the replay steps that are assumed to be occurring most often during rest intervals. A control

group is allowed to rest after the main task. This leaves the control group free to have replays. Evaluating and comparing the two groups' performance after this period will be a good indicator of the experience replay effect. Moreover, there are very few experiments in this field performed on humans. Therefore such an experiment can lead to very interesting results.

To sum up, this work is only the start of a path to building more human-like agents and discovering the underlying mechanism of the brain. Our work suggests that the replay idea has a potential neurobiological and computational basis, and can be a promising approach for future research.

Bibliography

A. Alvernhe, T. Van Cauter, E. Save, and B. Poucet. Different CA1 and CA3 representations of novel routes in a shortcut situation. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 28(29):7324–7333, July 2008. ISSN 1529-2401.

A. Alvernhe, E. Save, and B. Poucet. Local remapping of place cell firing in the tolman detour task. *European Journal of Neuroscience*, 33(9):1696–1705, 2011. ISSN 1460-9568.

K. Balakrishnan, R. Bhatt, and V. Honavar. Spatial learning and localization in animals: A computational model and behavioral experiments. Technical report.

A. Blaisdell, L. M. Gunther, and R. R. Miller. Recovery from blocking achieved by extinguishing the blocking cs. *Animal Learning and Behavior*, 27:63–76, 1999.

H. C. Blodgett. The effect of the introduction of reward upon the maze performance of rats. *University of California Publications in Psychology*, 4:113–134, 1929.

M. A. Brown and P. E. Sharp. Simulation of spatial learning in the morris water maze by a neural network model of the hippocampal formation and nucleus accumbens. *Hippocampus*, 5:171–188, 1995.

N. Burgess, M. Recce, and J. O'Keefe. A model of hippocampal function. 1994.

N. Burgess, J. Donnett, K. Jeffery, and J. O'Keefe. Robotic and neuronal simulation of the hippocampus and rat navigation. *Philos Trans R Soc Lond B Biol Sci*, 352(1360): 1535–43, 1997.

D. R. Euston, M. Tatsuno, and B. L. McNaughton. Fast-forward playback of recent memory sequences in prefrontal cortex during sleep. *Science*, 318(5853):1147–1150, 2007. doi: 10.1126/science.1148979. URL `http://www.sciencemag.org/content/318/5853/1147.abstract`.

D. J. Foster and M. A. Wilson. Reverse replay of behavioural sequences in hippocampal place cells during the awake state. *Nature*, 440(7084):680–683, Feb. 2006. ISSN 0028-0836.

D. J. Foster, R. G. M. Morris, and P. Dayan. A model of hippocampally dependent navigation, using the temporal difference learning rule. *Hippocampus*, 10:1–16, 2000.

A. S. Gupta, M. A. A. van der Meer, T. D. S., and A. D. Redish. Hippocampal replay is not a simple function of experience. *Neuron*, 65(5):695 – 705, 2010. ISSN 0896-6273.

K. Haberlandt, K. Hamsher, and A. W. Kennedy. Spontaneous recovery in rabbit eyeblink conditioning. *Journal of General Psychology*, 98:241–244, 1978.

T. Hafting, M. Fyhn, S. Molden, M.-B. Moser, and E. I. Moser. Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436(7052):801–806, June 2005. ISSN 0028-0836.

J. C. Jackson, A. Johnson, and A. D. Redish. Hippocampal Sharp Waves and Reactivation during Awake States Depend on Repeated Sequential Experience. *J. Neurosci.*, 26(48): 12415–12426, Nov. 2006. doi: 10.1523/JNEUROSCI.4118.

A. Johnson and A. D. Redish. Hippocampal replay contributes to within session learning in a temporal difference reinforcement learning model. *Neural networks : the official journal of the International Neural Network Society*, 18(9):1163–71, Nov. 2005. ISSN 0893-6080.

A. Johnson and A. D. Redish. Neural ensembles in CA3 transiently encode paths forward of the animal at a decision point. *Journal of Neuroscience*, 27(45):12176–12189, Nov 7 2007. ISSN 0270-6474.

W. Li, J. Li, and J. D. Johnson. A computational model of sequential movement learning with a signal mimicking dopaminergic neuron activities. *Cognitive Systems Research*, 6 (4):303 – 311, 2005. ISSN 1389-0417.

L. J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. In *Machine Learning*, pages 293–321, 1992.

R. E. Lubow. Latent inhibition. *Psychological Bulletin*, 79:398–407, 1973.

R. E. Lubow and A. U. Moore. Latent inhibition: The effect of non-reinforced preexposure to the conditional stimulus. *Journal of Comparative and Physiological Psychology*, 52: 415–419, 1959.

P. Maquet. The role of sleep in learning and memory. *Science*, 294(5544):1048–1052, 2001.

H. B. McMahan and G. J. Gordon. Fast exact planning in markov decision processes. In *ICAPS*, pages 151–160, 2005.

A. W. Moore and C. G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. In *Machine Learning*, pages 103–130, 1993.

R. M. Napier, M. Macrae, and E. J. Kehoe. Rapid reacquisition in conditioning of the rabbit's nictitating membrane response. *Journal of Experimental Psychology: Animal Behavior Processes*, 18:182–192, 1992.

J. O'Keefe and N. Burgess. Dual phase and rate coding in hippocampal place cells: theoretical significance and relationship to entorhinal grid cells. *Hippocampus*, 15(7):853–866, 2005. ISSN 1050-9631.

J. O'Keefe and D. H. Conway. Hippocampal place units in the freely moving rat: Why they fire where they fire. *Experimental Brain Research*, 31:573–590, 1978. ISSN 0014-4819. 10.1007/BF00239813.

C. Pavlides and J. Winson. Influences of hippocampal place cell firing in the awake state on the activity of these cells during subsequent sleep episodes. *Journal of Neuroscience*, 9(8):2907–18, 1989.

J. Peng and R. J. Williams. Efficient learning and planning within the dyna framework. In *Proceedings of the second international conference on From animals to animats 2: simulation of adaptive behavior: simulation of adaptive behavior*, pages 281–290, Cambridge, MA, USA, 1993. MIT Press. ISBN 0-262-63149-0.

R. A. Rescorla and A. W. Wagner. *A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement*, chapter 3, pages 64–99. Appleton-Century-Crofts, New York, 1972.

E. T. Rolls, S. M. Stringer, and T. Elliot. Entorhinal cortex grid cells can map to hippocampal place cells by competitive learning. *Network*, 17(4):447–465, dec 2006. ISSN 0954-898X.

D. Silver, R. S. Sutton, and M. Müller. Sample-based learning and search with permanent and transient memories. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 968–975, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4.

A. C. Singer and L. M. Frank. Rewarded outcomes enhance reactivation of experience in the hippocampus. *Neuron*, 64(6):910–921, 2009.

W. E. Skaggs and B. L. McNaughton. Replay of neuronal firing sequences in rat hippocampus during sleep following spatial experience. *Science*, 271(5257):1870–1873, 1996.

R. S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming, 1990.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, Mar. 1998. ISBN 0262193981.

E. C. Tolman. Cognitive maps in rats and men. *Psychological review*, 55(4):189–208, 1948.

E. C. Tolman and C. H. Honzik. Insights in rats. *University of California Publications in Psychology*, 4(4):215–232, 1930.

H. Voicu and N. Schmajuk. Latent learning, shortcuts and detours: a computational model. *Behavioural Processes*, 59(2):67 – 86, 2002. ISSN 0376-6357.

M. Wilson and B. McNaughton. Reactivation of hippocampal ensemble memories during sleep. *Science*, 265(5172):676–679, 1994.

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