NEURAL NETWORK MODELS OF HIERARCHICAL MAP-BASED PLANNING IN THE BRAIN

HENRY O. C. JORDAN



A thesis submitted for the degree of
Doctor of Philosophy in Experimental Psychology

The Queen's College
University of Oxford
Michaelmas 2018 – submission version

Henry O. C. Jordan: *Neural Network Models of Hierarchical Map-Based Planning In the Brain,* A thesis submitted for the degree of Doctor of Philosophy in Experimental Psychology © Jan 2019

This thesis is dedicated to everyone who has supported me through my DPhil. Special mention goes out to my supervisor, Dr. Simon Stringer, and to my lab-mates past and present. I learned a great deal from you all and I'll always be grateful for the privilege of working with you. My thanks also to my friends and family, whose patient support made this work possible and whose companionship made it fun.

ABSTRACT

Many researchers have tried to model how model-based behaviour is produced in the brain, and specifically how environmental knowledge is learned and used in the form of cognitive maps (see [1] for a review). Previous work has proposed various cellular substrates that might encode a cognitive map, as well as several possible mechanisms for self-organising these maps. Current models also agree to some extent on the planning mechanism that produces goal-directed actions using a cognitive map, although they propose subtly different implementations of this planning mechanism, and a variety of mechanisms for extending it. We felt that this previous work was limited in several important ways: there was little consensus on how these cognitive maps were represented and how they were formed, the planning mechanism that was used seemed to be inherently limited to performing relatively simple tasks, and there had been little consideration of how these mechanisms would scale up beyond laboratory tasks.

This thesis makes several important advances. Firstly, the planning mechanism that is used by the majority of current neural models on this subject propagates a decaying signal through the environment to create a gradient that points towards the goal. However, this decaying signal will eventually decay to nothing, limiting the scale and complexity of tasks that can be solved in this manner [2]. Based on the idea, originally proposed by Ponulak et al. 2013 [3], that a wavefront propagating from a goal state can carry information about the direction of that goal state, we propose a novel planning mechanism that does not require decaying activity.

Unlike the model proposed by Ponulak et al., the mechanism that we propose is: able to work in a state-action map, able to output explicit actions to move towards the goal, able to perform planning without altering synaptic weights, able to plan in nondeterministic environments, and able to interface with a hierarchical mechanism.

We also propose several ways in which a network which implements this planning mechanism can self-organise during unsupervised exploration. We demonstrate that the neural substrate for the cognitive map, the mechanisms required to plan using this map and the cognitive map itself can all be learned using local Hebbian learning, in an unsupervised manner. We compare and contrast these mechanisms in terms of both their effectiveness and their biological plausibility.

Finally, we extend the model with a hierarchical planning mechanism: a layer of cells that can identify frequently-used sequences of actions and reuse them to significantly increase the efficiency of plan-

ning. This increase in efficiency is largest for tasks with longer, more complex solutions. We speculate that this may explain the apparent ability of humans and animals to perform model-based planning on both small scales (laboratory experiments) and on large scales (outdoor travel) without a noticeable loss of efficiency [4].

ACKNOWLEDGMENTS

I would like to express my very great appreciation to Dr. Simon Stringer, my supervisor, for his patient guidance, his encouragement and his even-more-patient editing. I would also like to acknowledge all of the support and training I received from Dr. Daniel Walters during the initial years of my DPhil, and to thank my Transfer and Confirmation examiners Dr. Sanjay Manohar, Prof. Christopher Summerfield and Prof. Nicholas Yeung for their patient and helpful critiques. Finally, I would like to thank the Oxford Foundation for Theoretical Neuroscience and Artificial Intelligence for providing me with funding over the last two years.

Further thanks to the inventor of ClassicThesis and Nicholas Mariette, Ivo Pletikosić for producing a LyX port.

CONTENTS

1 INTRODUCTION TO THIS THESIS 1
- 1.1 Statement of Research Question 1
- 1.2 Broad Rationale for Research Question 1
- 1.3 Brief Outline of Relevant Literature 2
- 1.4 Stimulus-Response and Cognitive-Map Theories of Planning 3
- 1.5 Algorithms for Stimulus-Response and Cognitive-Map Based Planning 11
- 1.6 The Role of Hierarchical Representations in Planning Behaviour 22
- 1.7 Hierarchical Representations in Algorithmic Planning 25
- 1.8 The Structure of this Thesis 31

2 CONTEMPORARY ALTERNATIVE NEURAL NETWORK MODELS 33
- 2.1 The Aims of The Proposed Model 33
- 2.2 Producing a Literature Review 35
- 2.3 Cuperlier 2007 38
- 2.4 Hasselmo 2005 42
- 2.5 Erdem & Hasselmo 2012 51
- 2.6 Schmajuk 2006 56
- 2.7 Martinet 2011 60
- 2.8 Matsumoto 2011 64
- 2.9 Friedrich 2016 67

3 NAVIGATION TO A GOAL LOCATION USING STATE-ACTION CELLS AND A WAVEFRONT PLANNING MECHANISM 71
- 3.1 Planning In Simple Grid Worlds 71
- 3.2 Planning in Grid Worlds With Noise 90
- 3.3 Planning With Multiple Sensory Inputs 96
- 3.4 Discussion 106

4 SELF-ORGANISING STATE-ACTION CELLS FROM SENSORY AND MOTOR INPUTS USING HEBBIAN LEARNING 109
- 4.1 Formation of SA Cells using Diluted Synaptic Connectivity 111
- 4.2 Formation of SA Cells using State Columns 122
- 4.3 Formation of SA Cells using Bandstop Inhibition 138
- 4.4 Learning SA Cells with Multiple Sensory Inputs 148

5 SELF-ORGANISING A STATE-ACTION COGNITIVE MAP USING TRACE LEARNING 155
- 5.1 Self-Organising a Cognitive Map In the Recurrent Connections Between State Cells 156

5.2 Simultaneous Self-Organisation of State-Action Cells and a Cognitive Map Encoded in the Recurrent Connections Between Them 168

6 SELF-ORGANISATION OF A NEURAL READ-OUT MECHANISM FOR ACTION SELECTION 185

6.1 Self-Organisation of Gating Cells Using Bandstop Inhibition 185

7 REPLICATION OF A DETOUR TASK 203

7.1 Planning With Moving Obstacles 204

7.2 Planning with Probabilistic Propagation 212

7.3 Description of the Detour Task 221

7.4 Detour Task Simulation Procedure 222

7.5 Results of the Detour Task 224

7.6 Discussion 227

8 A HARDWIRED MODEL FOR PRODUCING HIERARCHICAL BEHAVIOUR USING SEQUENCE CELLS 229

8.1 Testing Possible Sequence Cell Connectivities 231

8.2 Testing The Utility of Sequence Cells 249

9 ENCODING USEFUL BEHAVIOURAL SEQUENCES BY SELF-ORGANISING A LAYER OF SEQUENCE CELLS 265

9.1 Self-Organising the Connectivity Between Sequence Cells and State-Action Cells Using Trace Learning 265

10 DISCUSSION 285

10.1 The Aims of The Proposed Model 287

10.2 Aim 1: Learn and Use a Causal Cognitive Map 289

10.3 Aim 2: Remain Consistent with Known Neurobiology 313

10.4 Aim 4: Learn and Use a Hierarchical Planning Mechanism 329

11 CONCLUSIONS 339

BIBLIOGRAPHY 343

LIST OF FIGURES

Figure 1.1 Detour Task Maze 7
Figure 2.1 Cuperlier 2007 Neighbourhood Connectivity 38
Figure 2.2 Cuperlier 2007: Interaction between the cognitive map and transition representations 40
Figure 2.3 Hasselmo 2005: Components of each model minicolumn 44
Figure 2.4 Hasselmo 2005: Activity in an example linear track task 45
Figure 2.5 Hasselmo 2005: Encoding of necessary connections between a state and the action randomly generated at this state 47
Figure 2.6 Hasselmo 2005: Simple schematic illustration of the steps of synaptic modification during encoding 49
Figure 2.7 Erdem & Hasselmo 2012: Place cell to PFC cortical column network 52
Figure 2.8 The problem of planning with linear probes 56
Figure 2.9 Martinet 2012: Overview of the model architecture and connectivity 61
Figure 2.10 Martinet 2011: Decaying activity limits viable path lengths 62
Figure 2.11 An Example of a Gaussian Place Field 65
Figure 2.12 Matsumoto 2011: A mechanism to create distance-dependent GPC activity 66
Figure 3.1 Architecture of the Proposed Network Model 72
Figure 3.2 A Two-State Grid World 75
Figure 3.3 Illustration Of the Read-Out Problem 79
Figure 3.4 The Gating Problem 80
Figure 3.5 Connectivity of a Gating Cell 81
Figure 3.6 Layout of grid world environments. 81
Figure 3.7 Activity of State-Action Cells During a Planning Task 87
Figure 3.8 Choosing Actions to Move Towards a Goal 88
Figure 3.9 Performance of the model when using simple threshold inhibition or bandstop inhibition 97
Figure 3.10 SA Activation from XY Goal Coordinates 99
Figure 3.11 The SA layer activity during a sequence of actions when stimulated by a goal cell layer encoding the goal in xy coordinates 100
Figure 3.12 Expanded network architecture including SA Goal cells 101

Figure 3.13 Success rate of the model with SA-Goal cells if state and goal are represented by xy coordinates 105
Figure 3.14 Success rate of the model without SA-Goal cells if state and goal are represented by xy coordinates 105
Figure 4.1 Architecture for Self-Organising SA Cells 113
Figure 4.2 Diluted connectivity to an SA cell from the state and action layers 114
Figure 4.3 Single Cell Information Before and After Training Using Diluted Synaptic Connectivity 118
Figure 4.4 The effect of dilution parameters on the number of SA cells that display maximum single cell information 119
Figure 4.5 Diagram of a Single SA Minicolumn 126
Figure 4.6 Firing of SA Cells After Columnar WTA Learning. 128
Figure 4.7 Single Cell Information Before and After Training Using State Columns 128
Figure 4.8 Activation of SA cells over the course of the first timestep in the SA learning task 129
Figure 4.9 Comparison of Dilution and Columnar WTA Techniques for Self-Organising SA Cells 130
Figure 4.10 The effect of bounding the synaptic weight vectors using normalisation or rescaling 135
Figure 4.11 The effect of rescaling parameters on the number of SA cells that display maximum single cell information 136
Figure 4.12 Average Single Cell Information Stored in SA Cells After Bandstop Learning. 142
Figure 4.13 Firing of SA Cells After Bandstop Learning. 143
Figure 4.14 The effect of exploration time on the number of states and SA combinations encountered 143
Figure 4.15 Effect of moving the bandstop window on the level of information encoded by SA cells 144
Figure 4.16 Single cell information learned by SA cells in a grid task with distributed state inputs. 151
Figure 4.17 Firing of SA Cells After Training Using Distributed Representations 152
Figure 5.1 Ideal Connectivity of a Pure State Cell 157
Figure 5.2 Variations of the Trace Learning Rule. 161
Figure 5.3 Trace Learning between two locations 163
Figure 5.4 Precision and Recall In State-State Maps 166
Figure 5.5 The Effect of Learning Time on Precision and Recall In a Four-Room Environment 167
Figure 5.6 The Pattern of Connectivity Between SA Cells 169

Figure 5.7 Architecture for Learning a Cognitive Map in the Recurrent Connections Between Self-Organising SA Cells 171
Figure 5.8 Learned SA Connectivity 177
Figure 5.9 Precision and recall of the self-organising columnar SA model. 180
Figure 6.1 Illustration Of the Read-Out Problem 187
Figure 6.2 The Gating Problem 188
Figure 6.3 Connectivity of a Gating Cell 189
Figure 6.4 Architecture of the proposed model with self-organising gating layer. 191
Figure 6.5 Single Cell Information Analysis of Gating Cells After Learning. 195
Figure 6.6 The number of gating cells that encode each state-SA combination before and after training 196
Figure 6.7 Efferent synaptic connectivity of gating cells after self-organisation 197
Figure 6.8 The effect of exploration time on planning success 198
Figure 6.9 The effect of planning time on the percentage of SA combinations encountered 199
Figure 7.1 Desired Effect of Re-Learning on SA Connectivity. 206
Figure 7.2 Layout of grid world environments. 207
Figure 7.3 Architecture of the proposed network model to be used in simulations with dynamically changing or probabilistic environments. 207
Figure 7.4 Obstacle avoidance through hardwiring. 210
Figure 7.5 A diagram of the expected pattern of SA connectivity 211
Figure 7.6 Trial success on obstacle task using Simulation Procedure A 211
Figure 7.7 Obstacle avoidance through relearning. 213
Figure 7.8 Trial success on obstacle task using Simulation Procedure B 214
Figure 7.9 Decaying Activity in Martinet 2011 214
Figure 7.10 Two Gate Environment (8x8). As in Fig. 7.2, the agent can move between blue squares, which are free states, and cannot move into yellow states, which are walls. The green square represents the goal, and the red square represents the agent's starting state. The agent therefore starts equidistant from both gates, and can reach the goal from either gate in the same number of timesteps. 217

Figure 7.11 Planning in a nondeterministic environment 220
Figure 7.12 Detour Task Maze 221
Figure 7.13 Occupancy Grids for Tolman & Honzig Detour Task 225
Figure 7.14 Model Output vs. Experimental Data and Martinet 2011 226
Figure 8.1 Architecture of the Proposed Hierarchical Model Including Sequence Cells 232
Figure 8.2 Connectivity Between A Sequence Cell and the SA Layer. 238
Figure 8.3 Connectivity between the SA layer and a sequence cell in the 'all-to-all' condition. 239
Figure 8.4 Propagation of Activity Through SA Layer in simulation with connectivity condition 1 239
Figure 8.5 Propagation of activity through SA Layer without using sequence cells 240
Figure 8.6 Performance in 100 planning trials with connectivity condition 1: The sequence cell receives synapses from the last SA cell in a sequence and sends synapses to all of the SA cells in that sequence. At the beginning of each trial, the agent and the goal were put on two sides of an 8x8 open environment (see Fig. 8.5). We see that in this case, the network is able to achieve a 100% success rate. 240
Figure 8.7 Propagation of Activity Through SA Layer with connectivity condition 1, when the starting position of the agent and the position of the goal are switched. 241
Figure 8.8 Performance in 100 planning trials with connectivity condition 1, when the starting position of the agent and the position of the goal are switched (Fig. 8.7) so that the agent is moving westwards and the eastwards sequence is now maladaptive. We see that even in this case, the network is still able to achieve a 100% success rate. 241
Figure 8.9 Connectivity Between A Sequence Cell and the SA Layer. 242
Figure 8.10 'All-to-all' connectivity between the SA layer and a sequence cell. 243
Figure 8.11 Propagation of activity through the SA layer with 'all-to-all' connectivity 243
Figure 8.12 Results of 100 trials with connectivity condition 2: 'all-to-all' connectivity 244

Figure 8.13 Propagation of activity through the SA layer with connectivity condition 2, if the starting position of the agent and the position of the goal are reversed. We observe that because the sequence cell can be activated by any SA cell in the sequence it is now activated even though the goal-based activity is actually propagating in the opposite direction. The wrong SA cells are therefore activating, and the agent will now move in the wrong direction. 244
Figure 8.14 Results of 100 trials with connectivity condition 2: 'all-to-all' connectivity and the starting position of the agent and the position of the goal reversed 245
Figure 8.15 Connectivity Between A Sequence Cell and the SA Layer. 245
Figure 8.16 Connectivity between the SA layer and a sequence cell in connectivity condition 3. 246
Figure 8.17 Propagation of activity through SA layer in simulation with connectivity condition 3 247
Figure 8.18 Performance in 100 planning trials with connectivity condition 3 247
Figure 8.19 Propagation of activity through SA layer with connectivity condition 3, when the starting position of the agent and the position of the goal are reversed 248
Figure 8.20 Performance in 100 planning trials with connectivity condition 3, when the starting position of the agent and the position of the goal are reversed. We observe that in this situation (where the agent is moving west and the SA sequence is therefore maladaptive) the model is unable to successfully complete any tasks. Fig. 8.19 shows that this is because the agent is being forced to move in the wrong direction by a wrongly-activated sequence cell. 248
Figure 8.21 Wave Propagation with and without Sequences 252
Figure 8.22 Swarm Plot 253
Figure 8.23 The efficiency benefits of using a sequence-based hierarchical mechanism 254
Figure 8.24 Fitting curves to the efficiency benefits of using a sequence-based hierarchical mechanism 255
Figure 8.25 The use of sequence cells produces more stereotyped behaviour 256
Figure 8.26 Comparison of a Sequence Cell with a pre-SMA Cell 260

Figure 9.1 Examples of two learned sequences 274
Figure 9.2 Success Rates in Planning Task When Using Learned Sequences 276
Figure 9.3 The occupation of different states during the sequence-learning period. 276
Figure 9.4 The effect of altering eta on the length of the sequences learned 277
Figure 9.5 Correlation of Planning Time to Path Length 281
Figure 9.6 The Model Replicates Brunye 2017 Results 283
Figure 10.1 The Size of Network Required to Learn Environments of Different Sizes 300
Figure 10.2 Architecture of the Proposed Network Model (Without Hierarchical Mechanism) 322

LIST OF TABLES

Table 3.1 Table of Parameters for Section 3.1 85
Table 3.2 Table of Parameters for Simulations With Noise Described in Section 3.2 96
Table 3.3 Table of Parameters for simulations with multiple sensory inputs described in Section 3.3 104
Table 4.1 Table of Parameters for Section 4.2 127
Table 4.2 The conditions required for the network to self-organise columnar SA cells 133
Table 4.3 Table of Parameters for Section 4.3 142
Table 4.4 Table of parameters for the model described in 4.3. 151
Table 5.1 Table of Parameters for Section 5.1 160
Table 5.2 Table of Parameters for self-organising model incorporating layer of SA cells described in Section 5.2 176
Table 6.1 Table of Parameters for Section 6.1 193
Table 8.1 Table of Parameters for the hardwired model with sequence cells described in Section 3.1 237
Table 9.1 Table of parameters for the learning of sequence cells as described in Section 9.1. 273

1

INTRODUCTION TO THIS THESIS

1.1 STATEMENT OF RESEARCH QUESTION

How might a biologically plausible neural network based on the architecture and operational principles of the brain learn to perform hierarchical model-based action selection as it explores a sensory environment?

1.2 BROAD RATIONALE FOR RESEARCH QUESTION

The experiments of Tolman [5–7] suggested that rats attain an internal model of their environment, called a cognitive map. Furthermore, they suggested that this cognitive map was necessary to respond quickly to changes in the reward available in an environment or to changes in the structure of the environment.[1] Examples of these are latent learning tasks and detour tasks, described more fully in Section 1.4.

Extensive modelling efforts [1] have validated this claim. Despite the development of sophisticated 'model-free' algorithms it has thus far been impossible to replicate certain observed behaviours without using a cognitive map representation [11, 12].

Several neural network models have studied the formation and use of cognitive maps.[2] However, many of the major questions around map-based planning have not yet been fully answered. How are spatial and non-spatial cognitive maps formed, stored, retrieved and used at the neural level? These problems are considered in Chapters 3 to 7 of this thesis. Furthermore, how are these processes affected

1 This was originally demonstrated in spatial tasks [6, 7] but proved to be true for more abstract decision making as well [8–10].

2 See Ch. 2: CONTEMPORARY ALTERNATIVE NEURAL NETWORK MODELS for a review of contemporary neural network models that learn and use cognitive maps.

by scale? It is uncertain how the storage and usage of cognitive maps differs between the small-scale mazes used in rat-based experiments and the large-scale maps of, say, a town, that a human would use to navigate. These problems are considered in Chapters 8 and 9 of this thesis.

There is some evidence that map-based planning can take place on very short timescales but also on much longer timescales, frequently abstracted from much detail about individual muscle movements [13, 14]. Behavioural research is currently investigating such 'high-level' or 'hierarchical' environment representations and map-based planning, and the results of these investigations are detailed in Sec. 1.6. Furthermore, a considerable amount of work in artificial intelligence research has been dedicated to investigating different forms of hierarchical planning in the context of Markov Decision Processes (MDPs) and Reinforcement Learning (RL); this research is discussed further in Sec. 1.7.

We believe that by modelling hierarchical, map-based planning in a biologically plausible neural network architecture—one which relies on local Hebbian learning to model synaptic plasticity and which self-organises its connectivity from sensory and motor inputs as the simulated agent explores its environment—we can begin to produce predictions of the neural and behavioural correlates that might accompany this form of planning. In particular, to produce predictions about how an unsupervised neural architecture could learn to represent useful sections of previously experienced trajectories and use these to plan at larger scales with greater efficiency, yet still in a biologically plausible fashion. Furthermore, by investigating the constraints of such a model, we can predict the features that a planning model needs to have in order to begin planning hierarchically, therefore constraining the space of possible action selection models.

1.3 BRIEF OUTLINE OF RELEVANT LITERATURE

The remainder of this introduction and literature review will cover the following topics in order:

- How animals produce behaviour, based on the presence or absence of cognitive maps (Sec. 1.4).
- Some algorithms that are frequently used to model planning behaviour (Sec. 1.5).
- Evidence for the role of hierarchical representations in planning behaviour (Sec. 1.6).
- Some algorithms that replicate hierarchical action planning (Sec. 1.7).

1.4 STIMULUS-RESPONSE AND COGNITIVE-MAP THEORIES OF PLANNING

This thesis describes a model of cognitive map-based planning. This section will therefore introduce cognitive map theory in more detail. It will also introduce the previously dominant theory of behaviour production, stimulus-response theory, which remains an important influence on thought about this topic.[3]

1.4.1 *Stimulus Response Theory*

Stimulus-response theory essentially states that all behaviour, however complex, deliberate and novel it might appear to be, is simply the summed output of finely tuned stimulus-response associations.

Early forms of this theory were heavily inspired by the experiments of Pavlov [15] and Thorndike [16] on classical and operant conditioning. Pavlov demonstrated that animals learned to associate a signal predicting a biologically significant event with the event itself, and would eventually respond to the signal in the same way that they responded to the event. He demonstrated this using hungry dogs: these animals were lightly restrained and a cue (such as a ticking metronome) was played for a few seconds before the delivery of food. The dogs did not initially respond to the metronome, but after repeating this procedure for some time, the dogs began to salivate copiously

3 Dolan and Dayan give a detailed review of this area in their 2013 paper, Goals and Habits in the Brain [1].

even though the food has not yet been presented. In other words, they had begun to respond to the cue (the metronome) as though it were the event (the food). This process has come to be called Pavlovian or classical conditioning [15].

Rather than studying the learning of associations between a cue and an event, as Pavlov did, Thorndike studied the association between a stimulus and a response, mediated by a reward. He found that animals which were rewarded for performing a particular action in a particular circumstance would learn to produce this action more frequently. A typical experiment placed a cat inside a cage with a bowl of food outside. This cage would only open if the cat performed a particular action, for example, pulling a lever. Thorndike found that it initially took a considerable period of time before the cat (accidentally) performed the appropriate action to release itself and reach the food. However, every time the cat was placed back in the cage, it would perform the action to release itself more quickly. Thorndike concluded that the reward was strengthening the association between the cat's situation (trapped inside the cage) and the action required to produce reward (pushing the lever) [16]. This form of conditioning is not quite the same as the classical conditioning described previously: Pavlov's work showed that animals which *naturally* produced Response R for a biologically significant Cue A would come to produce the same response for neutral Cue B, provided that B predicts A, even if the animals are not directly rewarded for producing that response. Thorndike's work showed that an animal could be *trained* to produce Response R for a previously neutral cue if it received a reward for producing that response. In other words, Thorndike showed that an animal could be trained to associate a neutral stimulus directly to an action response using a system of constant or intermittent rewards. This is known as instrumental or operant conditioning [17].

Following further study, a body of work developed that attempted to explain all animal behaviour (including classical conditioning) in terms of operant conditioning, so that all behaviour resulted from the interaction of environmental features and learned responses [1, 17].

1.4.2 *Challenges to Stimulus-Response Theory and the Development of Cognitive Map Theory*

The 'strong' stimulus-response hypothesis—that all animals solve problems purely by trial-and-error learning of stimulus-response associations—was challenged by a growing body of evidence showing that certain behaviours seemed to result from an internal 'understanding' of an environment or task rather than purely from learned associations.

1.4.2.1 *'Insight' Learning*

As the body of stimulus-response work developed, researchers such as Kohler and Epstein argued that the experiments which formed the basis of the stimulus-response hypothesis were too restrictive and unnatural, so that animals could not display any more complex signs of reasoning. They provide examples of animals producing sudden solutions to complex problems without trial and error, which is sometimes called 'insight' learning. For example, a dog inside a room sees a man in the same room throw food out of a window and after a moment's pause, immediately runs outside and finds the food lying underneath the window. Other examples include a caged ape which attempts to use various tools lying nearby to reach a fruit. After a series of failures and an extensive pause 'for thought', during which the ape gazes around the room, it suddenly uses a small stick to reach a longer stick that it could not reach by hand, and uses the longer stick to reach the fruit [18].

These and other experiments appeared to show purposive, reasoned animal behaviour, which draws upon abstract reasoning and prior experience to produce the answer to a problem without extensive trial and error. However, the complexity of these experiments and the requirement for general environmental knowledge (the ape, for example, had played with sticks extensively before the experiment in question) make it almost impossible to prove that these behaviours did not arise from some combination of learned stimulus-response associations. The trade-off between the 'naturalness' of an experiment

and the control of extraneous factors is a constant problem in the study of animal cognition, and makes it very difficult to produce complex experimental behaviours that cannot be reduced to stimulus-response learning.

1.4.2.2 *Tolman, Latent Learning and Cognitive Maps*

Researchers also challenged the idea that all learning was reward-driven, and therefore fundamentally an extension of operant conditioning. In particular, Tolman argued that much of animal learning was simply associative. He argued that classical conditioning was not an extension of operant conditioning but was as Pavlov had initially described it: the association of an earlier stimulus with a later stimulus. Tolman suggested that neither the associated response (salivation, in the case of the dogs) nor the reward (food) were required for this form of learning. Furthermore, he argued that animals could learn stimulus-response associations simply by exposure, without reward [19].

Essentially, Tolman suggested that animals learned constantly, even without the presence of reward, and that they actively used the information to produce "a tentative cognitive-like map of the environment ... including routes and paths and environmental relationships, which finally determines what responses, if any, the animal will finally release" [5]. Tolman and other researchers conducted a variety of more controlled experiments to provide support for this hypothesis.

In one study he demonstrated that rats could learn about their environment even when no reward was present, and utilise this learned knowledge when it became useful [5]. Specifically, he allowed rats to explore a maze without reward; when a reward was put into the maze, the rats which had already explored the maze were able to reach the reward faster than rats which lacked this experience, suggesting that the rats had learned an internal map of the maze during unrewarded exploration and were now using it to navigate. In another paper Tolman demonstrated that rats experienced indecision at choice points in a T-shaped maze, thus suggested that they were con-

sciously juggling different possibilities [6]. Finally, he demonstrated that rats could deduce shortcuts through a familiar space even if those shortcuts were never rewarded and so had no associated stimulus-response mapping [7], which again suggested the creation of an internal map. To do this he put rats in a maze that required them to initially take a long and convoluted path to reach the goal, one that initially moved them in the wrong compass direction. He then blocked that path, and allowed the rats to choose between 18 straight radial paths, finding that rats tended to choose the path that led directly towards the goal.

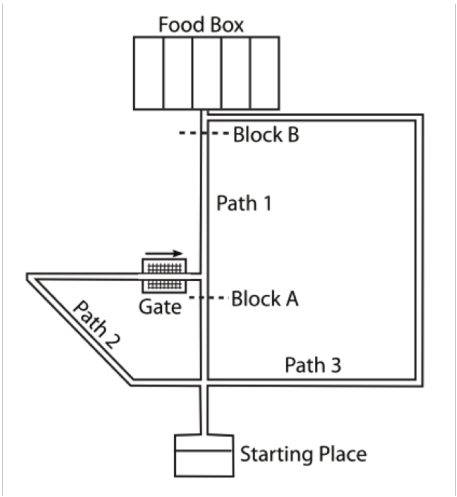


Figure 1.1: The maze used in the Tolman & Honzig detour task [2].

An important and influential experiment of Tolman's was the Detour task. This task (illustrated in Fig. 1.1) places rats in a maze with three paths of varying length: Path 1 (short), Path 2 (medium) and Path 3 (long). At the end of the maze is a food source (a goal state in the model). Put briefly, rats are allowed to explore the maze and are then expected to seek the reward. Path 1 is shortest; if it is open, rats will take it. If Path 1 is blocked at point A, so that only Path 2 and Path 3 are accessible, rats will then take Path 2 the majority of the time, without extensive retraining. If Path 1 is blocked at point B, rats take Path 3. This behaviour happens quickly, without extensive exploration, suggesting that rats are relying on a previously learned map of their environment to predict the outcomes of alternative trajectories through the environment.

Tolman's work helped to form the *cognitive map hypothesis*, which stated that animals such as rats were capable of forming and storing an internal map of a familiar environment and then leveraging that knowledge to make formal deductions about the space. This internal map could therefore be used to produce novel goal-directed behaviour.

Unfortunately, Tolman's experiments were conducted more than seventy years ago and from a modern perspective Tolman's experiments were considerably less conclusive than he claimed and his contemporaries believed. As stated above, it is difficult to conduct animal experiments in such a way as to rule out behavioural 'hacks' that use stimulus-response learning to produce the appearance of reasoned behaviour. For example, the experiment in which Tolman demonstrated that rats could select novel shortcuts through a known environment [7] was conducted with a light bulb above the goal, visible from all points of the maze, and so the rats may have simply associated the light with food and moved directly towards it, taking the novel shortcut as a side-effect of this process. However, most of his core findings were later replicated by more rigorous experiments [12, 20, 21].

1.4.2.3 *Neurological Investigation of Cognitive Maps*

Although Tolman's experiments were not conclusive, the cognitive map hypothesis has been greatly strengthened by further experimental evidence since it was originally proposed. In particular, the discovery of place cells in the hippocampus [22] gave a neural substrate in which the cognitive map might be stored. The later discovery of grid cells in the entorhinal cortex [23] gave rise to theories that combinations of grid cell inputs might give rise to place cell responses [24, 25] and so about how the cognitive map might arise. A little later still, the work of Johnson and Redish 2007 [26] showed apparent neural correlates of map-based planning in place cells: rats at a choice point sent waves of activity along the place cells that represented various

future paths, apparently selecting between them in a process dubbed 'mental time-travel'.[4]

Several experiments have provided indications that cognitive maps also exist for non-spatial domains. Hampton et al. [8] measured the activity of the ventromedial prefrontal cortex (theorised to represent value information) using fMRI as human subjects performed a two-choice task. In this task, subjects were repeatedly presented with the same two visual stimuli. One of these stimuli was designated 'correct' and the participant was rewarded for selecting it; the other stimulus was 'incorrect' and the participant was punished for selecting it.[5] At semi-random intervals, the reward structure was reversed, such that the 'correct' stimulus became the 'incorrect' stimulus, and the 'incorrect' choice became the 'correct' choice. Hampton et al. showed that when the reward structure reversed, participants quickly and correctly predicted (via ventromedial PFC activity) that the previously 'incorrect' (but now 'correct') action now had high value, indicating that the subjects recognised the reversal and could predict the new values of each stimulus without relearning them. In other words, they had recognised the move from one abstract state (normal) into the next (reversed), indicating that they are able to encode the presence of abstract hidden state.

A later experiment by Kurth-Nelson et al [9] analysed whole-brain magneto-encephalographic (MEG) data while subjects performed a non-spatial navigation task[6] and found that (a) the current state could be reliably decoded from MEG data and that (b) once the task had

4 It is not clear whether such place cell activity is responsible for producing map-based behaviour or whether it represents an input-to/output-from/reflection-of planning processes occurring in the basal ganglia and prefrontal cortex. Chersi et al. [27] explore the incongruities between hippocampal pre-play (as described by Johnson & Redish [26]) and contemporary neural network models of map-based planning (as described in the next chapter).

5 'Correct' stimulus: 70% chance of 25 cents, 30% chance of -25 cents. 'Incorrect' stimulus: 40% chance of 25 cents, 60% chance of -25 cents.

6 States are defined by displaying a unique visual object and were each associated with a varying reward. From each state the model could move 'up' or 'down', moving it to a different state that was identified by a different visual object. Participants were trained to learn the structure of the environment in advance and were then asked to enter a sequence of four moves with the goal of collecting as much reward as possible. Kurth-Nelson et al. said that "At debriefing, no participant reported conceiving the relationships between objects in a spatial manner" and furthermore "all participants reported a subjective experience of deploying knowledge of transitions for planning".

been learned, spontaneous MEG activity encoded reverse paths up to four states long. In other words, when the MEG activity of participants during the planning task was decoded using a trained regression model, the MEG activity encoded a series of four state transitions in reverse. To quote: "If a neural representation of state k was active at time t, then a representation of one or both of the states with transitions to k was active around t + 40ms." These four-state sequences were legal moves according to the transition structure of the environment, and appear similar to the replay that appears in rodent hippocampus after spatial navigation tasks, except for the fact that they appear in reverse.

Finally, a recent experiment by Aronov et al [10] seems to have shown the existence of 'place cell' representations for non-spatial state-spaces. Mice were given the ability to alter auditory stimuli along a continuous frequency axis and cells developed in the hippocampal CA1 and the medial entorhinal cortex that have discrete firing fields along particular areas of the frequency axis. These cells overlapped with spatial place cells and grid cells but not in any organised fashion, which suggested that some spatial cells were being randomly repurposed for this new representation. Interestingly, these neurons appeared to be task-selective; when Aronov et al. presented the same mice with the same stimuli (sweeps of the same frequency and duration) outside the context of the task, they found very little activity in the cells that had earlier fired reliably at different parts of the same axis.

1.4.3 *Recent Interactions Between Model-Free and Model-Based Theories of Behaviour*

In the last ten years models have emerged which posit that both model-free and model-based planning mechanisms coexist in the brain and co-operate or compete to form observed animal behaviour. Usually these models posit that stimulus-response action selection provides semi-optimal habitual responses which are used for quick and/or efficient planning under constrained conditions while a cognitive map

provides more optimal behaviour but at a higher cost in terms of mental resources and planning time [1, 28].

1.5 ALGORITHMS FOR STIMULUS-RESPONSE AND COGNITIVE-MAP BASED PLANNING

Following the previous section's description of stimulus-response and cognitive-map based theories of planning, this section will give a brief overview of how these behaviours are typically modelled. Although this thesis models map-based planning at the neural scale, this section will introduce the *algorithmic* formulations that are typically used by engineers and researchers to model map-based planning, acting as a precursor to the review of biologically plausible[7] *neural network models* in the next chapter.

The section begins by describing how Markov Decision Processes (MDPs) are typically used to formulate and solve planning problems. It will go on to describe model-based and model-free reinforcement learning and how they can be used to model cognitive-map and stimulus-response planning mechanisms. Note that a Markov Decision Process (MDP) by definition describes the full structure of a problem, and can be solved to produce a solution to that problem; by contrast, *both* model-based and model-free reinforcement learning are ways to solve an MDP that is initially unknown. In other words, when an agent does not know the states that it can visit, and does not know the transition structure that connects those states, it can use a reinforcement learning technique to learn and solve that MDP. The simplest way to do this is to fully learn the MDP and then solve it using the techniques described in Sec. 1.5.1. This can be thought of as a form of model-based learning (see Sec. 1.5.2). Most forms of reinforcement learning attempt to (partially) discover and (approximately) solve the MDP simultaneously, using a mixture of exploration and exploitation, and these methods are described throughout Secs. 1.5.2 to 1.5.5.

7 According to the definition given in Secs. 1.2 and 2.1.

1.5.1 *Markov Decision Processes*

Beyond initial forays into geometric planning and the use of dynamic control theory for motor control, most planning algorithms derive from some form of Markovian decision process. In a Markovian formulation, an agent occupies a particular 'state'[8] s at a given point in time t, based on the condition of the environment, the task and the agent itself. If the agent takes an action a, that action will alter the agent's state in a partially predictable fashion. Specifically, the action will transition the agent from its current state s to a new successor state s', with a certain probability $T\left(s, a, s'\right)$ taken from a transition matrix T which describes the relationship between states, actions and successor states.[9]

Two methods for solving a Markovian Decision Process, calculating an optimal policy from the transition matrix and reward function, are policy iteration and value iteration.

POLICY ITERATION An agent visits states according to its policy. This policy is calculated to maximise reward by visiting the most valuable states. However, the value of each state depends on the agent's policy! This is because the value of a state incorporates the expected reward that the agent will receive after leaving that state (eq. 1.1). Even if the agent is not directly rewarded for visiting a given state, that state may still be valuable because under the agent's current policy it will cause the agent to reach another state which does produce direct reward. The value of a state in policy iteration is given by the following equation:

$$v(s) = \sum_{s'} p(s' \mid s, \pi(s)) \left[r(s, \pi(s), s') + \gamma v(s') \right] \tag{1.1}$$

where $v(s)$ is the value of state s, s' is any other state that the agent may enter in the future, $p(s' \mid s, \pi(s))$ is the probability that

8 States can be discrete or continuous.

9 The fundamental Markovian assumption is that your transition probability depends *only* on your current state and action, although this assumption is frequently relaxed in more sophisticated Markovian formulations. For example, the current 'state' may be considered to be a vector of the last 5 states.

an agent will move to any s' given its current state and its policy $\pi(s)$ for that state, and the function in square brackets denotes the total future reward that the agent expects to receive over all states if it moves to state s', discounting future reward to some extent. The γ term defines the discounting of reward obtained in future states and therefore defines the value of predicted reward. If γ is low, then long-term reward is considered much less valuable than short-term reward. If γ is high, then future reward is considered to be of nearly equal value to short-term reward.

Since the value of states depends on the agent's policy, and the agent's best policy depends on the value of states, the agent can improve its policy if it has a better estimate of state values. At the same time, the agent's estimation of state values assumes that the agent will continue to operate on its current policy (eq. 1.1). If an agent finds a more optimal policy, it can therefore improve its estimation of state values by recalculating them based on this new policy. This is the basis of policy iteration.

Policy iteration has two phases: policy evaluation and policy improvement. The agent begins with an arbitrary policy. During *policy evaluation*, the agent determines the value of each state (the value function) based on its initial policy. During *policy improvement*, the agent produces a new policy designed to maximise the amount of reward it will receive according to the current value function.

The agent iterates between policy evaluation and policy improvement until both the policy and the value function become optimal [29].

VALUE ITERATION Value iteration is a variant of policy iteration that uses less computational resources per timestep but usually requires more iterations to converge. In policy iteration, the value of a state depends on the agent's policy, and the agent's optimal policy depends on the value of each state. Both of these can therefore be updated iteratively.

By contrast, in value iteration the assumption is made that the agent will always move to the most valuable adjacent state. This

means that it is no longer necessary to know the agent's policy in order to calculate state values—the agent's policy π is literally removed from the equation:

$$v(s) = \max_a \sum_{s'} p(s' \mid s, a) \left[r(s, a, s') + \gamma v(s') \right] \tag{1.2}$$

The value of a state now depends purely on the values of nearby states, and the agent can therefore calculate the optimal value function directly, by iteratively improving its estimate of state values. Once the agent knows the optimal value function, it can then use that to generate an optimal policy in a single step [29].

1.5.2 *Model-Based Reinforcement Learning*

The term 'model-based reinforcement learning' can refer to any reinforcement learning paradigm that relies in some way on learning a model of the environment and/or task. This includes approaches where a model is learned and then analysed to produce a policy that maximises reward, as well as approaches where a model is learned and then used to produce experience in a virtual recreation of the task. The latter approach is primarily used in the robotics literature, where exploration is dangerous and expensive due to physical wear-and-tear, the risk of destruction etc. and so data is hard to obtain [30], and will not be discussed in this thesis.

The simplest model-based reinforcement learner has two phases: a learning phase and a planning phase. In the learning phase the agent explores its environment and learns a transition matrix (how states are connected) and a reward function (where rewards are located), producing a Markov Decision Process (MDP). In the planning phase the agent solves the resulting MDP to produce an optimal policy, using value iteration or policy iteration methods (see Sec. 1.5.1). Algorithms which use this two-stage approach are sometimes called *certainty equivalent methods [1, 31]* because the algorithm assumes that once it has completed the learning phase then its knowledge of the environment is so extensive that it can produce optimal behaviour

without further learning.[10] A safer but computationally demanding variation on this idea is for the model to learn continuously through the agent's lifetime and recalculate the optimal policy and value function at every timestep. This has the additional benefit that the agent is more likely to experience hard-to-reach areas of the state space, which can only be reached by a controller that is already reasonably sophisticated.

Model-based planning has certain advantages: if the transition matrix is wholly or mostly known, the value function and policy can be calculated directly, without further exploration. In general, model-based techniques can be highly sample-efficient: it is often the case that considerably fewer samples are required to model a task (i.e., learn its transition structure) than to learn a policy using model-free methods (see next section). There is a considerable resultant processing burden required to calculate an optimal policy from the learned model, but if sampling efficiency is more important that processing efficiency then model-based techniques are often desirable.

Model-based reinforcement learning can also be more flexible than model-free learning. If the transition structure of the environment is known (or well-approximated) this knowledge can be leveraged to quickly update or replace the agent's policy if new reward information is revealed. This is in contrast to a model-free approach, which would need further real-world samples to learn the new state/action values and thence form a policy. Similarly, if the transition structure of the environment changes, such that previously optimal policies are no longer useful, a model of the new transition structure can be used to quickly update or reproduce the agent's policy. This flexibility allows model-based approaches to perform well in revaluation tasks and contingency change tasks such as the Tolman detour task (see next section).

Having obtained a model of the transition structure, producing and updating the agent's policy are the most computationally expens-

10 The term "certainty equivalence" originates from the control theory literature. The certainty equivalence principle asserts that control under certain conditions can be considered optimal, as if the agent's knowledge of its environment is certain, without error [32].

ive elements of the algorithm. Many initial improvements to model-based reinforcement learning therefore centred on optimising updates. For example, Dyna [31] was a version of model-based reinforcement learning that performed efficiently by reducing the rate at which it updated its policy. Where a naive model-based reinforcement learning would update its entire policy when it received a new reward, Dyna only updated the value of the current state and k randomly chosen state-action pairs every timestep, leaving the rest unchanged. The updates were performed using a learned transition matrix $T\left(s, a, s^{'}\right)$ and so Dyna was fundamentally a model-based reinforcement learner but increasingly behaved according to learned associations as the ratio of map size to the number of randomly chosen pairs k increased.

A further update to model-based reinforcement learning came with Prioritised Sweeping and Queue-Dyna [31], which both seek to optimise policy updating in very similar ways. Prioritised Sweeping gives every state a priority value which determines how likely the algorithm is to re-update that state's value; discovering a surprising state (which has experienced a large change in value) causes the algorithm to prioritise the surrounding states.

Markovian decision trees seem to be a nice fit for many elements of the cognitive map hypothesis. However, computer scientists quickly observed that the Markovian decision tree method of action selection did not scale well. As the scale of the world and the complexity of the agent increased, the algorithm was required to propagate values over an exponentially growing number of branches. Watkins and Dayan [33] observed that this problem could be circumvented by calculating a policy for each state based purely on the average discounted reward received after making each possible action in that state, effectively associating each stimulus with a highly rewarded response. This is known as model-free reinforcement learning or q-learning.

1.5.3 *Model-Free Reinforcement Learning*

Model-free reinforcement learning or Q-Learning arose from the realisation that Q-values (the value of particular state-action com-

binations) could be calculated directly from discounted reward. In the simplest form, a state-action's Q-value is calculated simply as the average reward (discounted by time) received after performing that action in that state. There is no need to store or calculate over a transition matrix to create a policy; the agent simply selects whichever action bring the highest average reward.

Q-learning is efficient because it discards the large transition matrix that forms its model of the world. However, this frequently gives rise to a *credit-assignment problem.* If an agent has no model of its environment then that agent can have no basis to reason about *how* or *why* a particular state-action combination leads to a later reward. A model-free algorithm therefore finds it difficult to correctly separate the state-action combinations that *cause* reward from those that are *correlated* with reward and disproportionately assigns credit to state-action combinations that are close in time to reward. Experimentalists have observed similar false attribution in laboratory animals [34] and so this may partially reflect a true aspect of planning in the brain but nevertheless demonstrates the limitations of using model-free reinforcement learning as a general theory of planning.

The temporal-difference method is a technique for improving credit assignment. This algorithm does not update Q-values directly from observed reward; instead it maintains a prediction of discounted future reward and updates the Q-values to match that prediction. This means that each Q-value can be updated immediately according to currently available information, rather than trying to back-propagate future reward.

As people began trying to use model-free reinforcement learning to solve larger, more complex tasks, it became clear that (a) most tasks have continuous state-spaces that cannot be nicely delineated for a discrete learner and (b) that model-free learners will probably not encounter the same state twice. Model-free learners could no longer afford to simply rely on a look-up table of average discounted reward for all states. There was consequently a rising interest in Q-function predictors, which could apply a function to an unexplored state and accurately predict the q-values of different actions taken in this state.

Work by Mnih et al. [35] demonstrated that a neural network could be used to perform end-to-end Q-learning on ATARI games and so learn a continuous Q-function that outperformed humans in many cases.

In conclusion, model-free learning has been greatly improved and extended since its inception but is still mostly applied to single problems with stable reward functions. Because it learns no transition matrix, its ability to perform is tied closely to reward and so suffers when the reward function changes, necessitating heavy re-learning.

Historically, two kinds of animal behaviour tasks have been presented as arguments against purely model-free theories of planning in the brain: revaluation tasks, which examine whether animals adjust their behaviour appropriately following changes in the reward function, and contingency change tasks, which examine whether animals change their behaviour appropriately following changes in the transition structure of the environment (for example, a blocked or opened passageway in a maze) [5, 11]. Model-free reinforcement learners perform poorly in these tasks because they cache the cumulative expected rewards that they expect from different state-action combinations and—without a transition model of the environment—have no way of updating these cached values post-manipulation except to relearn its Q-function. In comparison, model-based reinforcement learners can immediately update their value-functions and policies, continuing to produce adaptive behaviour in the face of environmental changes or altered rewards [11]. Russek et al. 2016 [11] found that a successor representation approach could only solve these tasks if mixed with some form of explicitly model-based reinforcement learning.

1.5.4 *Deep Reinforcement Learning*

Deep reinforcement learning is the name that has come to be attached to the various uses of a deep neural network to approximate a reinforcement learner using the backpropagation of error. Such 'Deep RL' networks have achieved considerable success in a wide variety of domains [36]. Their main strength is that they are able to learn the

complex nonlinear functions required for performing continuous reinforcement learning tasks. Although the use of neural networks to perform reinforcement learning predates the rise of Deep RL methods, initial approaches suffered from instability. When a nonlinear function approximator (for example, a neural network) was used to represent the value of state-action combinations (a Q-function), researchers found that the results did not converge to a high-reward stable state but remained unstable or even diverged. Mnih et al. [35] blamed this on several factors in their seminal 2015 paper: sensory inputs were received in temporal sequence and so contained many irrelevant correlations, small updates to the network could result in large policy changes, small updates to the network could result in large input changes (as a result of those policy changes) and the learned Q-values and target Q-values were inherently correlated [35].

Minh et al. showed that a deep learning network could learn to achieve a high score in a variety of Atari video games by implementing a form of temporal difference learning. In particular, they showed that the same method and hyperparameters produced near- and super-human performance on a large number of very different games even if the agent was only given the pixel values and score; the agent did not have to be carefully tailored for each different game [35]. The power and flexibility of this approach was clearly apparent and a large number of papers were published that either refined this method (proposing extra/alternate neural mechanisms and training procedures) or showed its applicability in different domains [36]. For example, van Hasselt et al. (2016) published a technique that corrected the tendency of Mnih et al's 'Deep Q Network' to overestimate the value of certain state-action combinations [37]. Schaul et al. (2016) proposed a technique for ensuring that the Deep Q Network spent more time analysing 'significant' experience [38].[11] Neural network methods have also been applied to other RL methods, such as actor-critic RL [39] and policy gradient RL [40]. Li 2017 [36] provides a more detailed overview of this field (allowing for the field's rapid ex-

11 'Significant' in this case is measured by a high error between the predicted value of a state-action combination and the actual value.

pansion) and Henderson et al. 2017 [41] provides a statistical critique of the reproducibility of results in this space.

Much of the research in the field of Deep RL is focused on model-free methods [36, 42] but several researchers have noted the data inefficiency of Deep RL (inherited from the model-free principles that conventional Deep RL is based on) and there has been some interest in applying deep learning to model-based methods [36, 42]. Model-based reinforcement learning algorithms in continuous state spaces have conventionally used function approximators or bayesian methods to learn a dynamical model of the system with relatively few samples, but these methods are difficult to apply to complex, high-dimensional tasks, and neural networks have not performed well as a replacement. Furthermore, model-based deep RL usually has a low final performance compared to model-free approaches, because the learned model is inevitably imperfect and propagates inaccuracies through the network [43, 44].

Some current work therefore focused on combining model-free and model-based RL in various ways. An example of this work is Nagabandi et al. 2017, which demonstrates techniques for improving the use of neural networks as function approximators in deep model-based RL. Nagabandi et al. further show that deep model-based RL and deep model-free RL are complementary, using deep model-based RL to learn a 'good-enough' policy for a complex system[12] and then using model-free techniques to fine tune the resulting policy. This transition from model-based to model-free methods allows the learner to continue improving its performance to the expected level for a model-free learner—correcting for the imperfections of the original model—while producing efficient initial learning [43]. Much like other deep RL methods, this is very recent work, and the applicability of these methods is not yet clear.

12 They claim that less than four hours of random action data was needed to do so, making the technique potentially very useful in real-time robotics applications.

1.5.5 *Successor Representations*

Purely model-free reinforcement learning does not appear to provide a full explanation of animal behaviour because it responds poorly to changes in transition probabilities and provided rewards [1, 5, 11]. Recognising this, researchers have tried to merge model-free and model-based planning in various ways. One of these is to posit that these mechanisms coexist and interact, either co-operatively or competitively. This work is described in sec. 1.4.3.

Another approach, which has recently had considerable success, is based on the *successor representation.* The successor representation is, technically speaking, the expected discounted future occupancy of all states s' considering the agent's current position s. In other words, the successor representation is a two-dimensional matrix $M(s, s')$ that represents the probability that an agent will eventually occupy any state s' given a starting state s, discounted by the amount of time until that occupation is predicted to occur [45]. The successor representation literally represents the expected successor states.

In Sec. 1.5.2 we explain that the value of a state is determined by the expected cumulative reward, which is determined by the combination of the reward available at a state and the (discounted) likelihood that the agent will occupy that state (eq. 1.1). In Sec. 1.5.2 this likelihood is calculated from the agent's policy but here this likelihood is encoded in the successor representation[13]. The value of a state can therefore be represented as:

$$v(s) = \sum_{s'} M(s, s')R(s') \tag{1.3}$$

where M is the successor representation and $M(s, s')$ is the expected discounted future occupancy of state s' given that the agent is in state s and $R(s')$ is the reward that the agent will receive in state s'.

The successor representation implicitly encodes the structure of the environment, because the likelihood that an agent will occupy a space

13 The successor representation is learned using a variant of temporal difference learning that we will not cover in detail. See Dayan 1993 [46] or Stachenfeld et al. 2014 [47] for a full explanation of this process.

is partly a function of the environment's transition structure. As such, the use of a successor representation creates a reinforcement learner that combines some of the advantages of model-free and model-based algorithms. The agent is able to form a policy efficiently (as in model-free reinforcement learner) but at the same time it is sensitive to changes in the task's reward structure, in the manner of a model-based reinforcement learner [47].

It is possible, therefore, that the use of a successor representation may allow a reinforcement learner to perform tasks that defeat a purely model-free learner without using full model-based techniques. Russek et. al therefore reinterpreted a set of revaluation and contingency change tasks—which respectively change the reward function and transition structure of the task—as grid world tasks. They then tested various reinforcement learners on these tasks. Russek et al. 2016 found that augmenting temporal difference learning with a successor representation (TD-SR) was not enough to replicate animal and human behaviour on all of the tasks that they examined. Specifically, they found that the TD-SR learner was not able to perform transition revaluation and detour tasks, and in particular fit poorly to the results of a detour task performed by humans [11, 48].

1.6 THE ROLE OF HIERARCHICAL REPRESENTATIONS IN PLANNING BEHAVIOUR

An important objective of this thesis is to investigate the production and use of *hierarchical* map elements in a biologically plausible neural network model. Accordingly, this section will discuss the nature of these hierarchical elements *in vivo* as far as it can be deduced from behavioural studies.

Extensive evidence shows that humans represent space in a regional and hierarchical fashion. Hirtle and Jonides 1985 [49] describe an experiment showing that participants make systematic errors when judging the spatial relationship between American cities, because their judgements of the lower-level relationships between cities are influenced by the higher-level relationship between American states. Holscher

et al. showed that human participants represented a complex public building regionally, in terms of floors and building sections, when required to search for a given room [50].

Regional effects seem to occur even in spaces whose hierarchical structure is not explicitly defined: participants produce faster judgements when verifying directions between two spatial clusters than between two points (of identical distance) within the same spatial cluster [49].

Noack et al. recently demonstrated that sleep is partially responsible for this regionalisation, increasing awareness of implicit and explicit semantic regions in a virtual environment [51].

Experimental work has also indicated that humans plan routes at different levels of the hierarchical representation. When giving route directions, either in advance or during travel, participants reliably produce high-level route segments in order to optimise information-to-memory-load [52]. Furthermore, participants reference elements of a city in order from the most general to highly specific local references [53], signifying that most people store hierarchical spatial information in the same way and are able to predict what level of information other people are likely to recognise. Timpf et al. (2003) break down highway navigation into three levels of abstraction: *route planning* operates on a very high level, finding routes given certain constraints; *driving* is performed at a fine level; and people produce *driving instructions* at a level partway between the two, dividing the route into segments that amalgamate a certain amount of driving activity [54]. Wiener & Mallot (2003) similarly show that humans appear to use a fine-to-coarse planning heuristic when navigating: a route plan contains fine-space information for the close surroundings and coarse space information for the rest of the route [55]. There is also evidence that route familiarity is an important element in human navigation [56].

It also seems clear, although there is a paucity of work on the subject, that the scale on which high-level route segments are generated and used depends on the size of the task in question. People give directions of a roughly similar length for routes whose lengths differ

by orders of magnitude, provided that route fragments of the appropriate scale exist.

At the same time, planning tasks performed at the same level of abstraction show planning times at a given level of abstraction are partially proportional to absolute path length [57]. We believe that a model of map-based planning should explain both the generation of high-level path fragments as well as how planning operates at any given level of abstraction.

The problem of hierarchical planning is similar, but not identical, to the problem of hierarchical motor control. The latter arises from the physiology of the motor system. Complex voluntary motor behaviour appears to be produced in a hierarchical fashion, being controlled at an abstract level by the premotor cortex, and at a progressively less abstract level in the motor cortex, the brain stem and the spinal cord. For example, a well-known study showed that the premotor cortex encodes the intended direction of reaching movements during reaching tasks whilst the primary motor cortex encodes specific directions and amounts of muscular force intended to bring this movement about [58]. Meanwhile at the bottom of the hierarchy, the spinal cord encodes a variety of low-level reflexes and central pattern generators [59]. Recent studies also seem to indicate that the ventral premotor cortex is itself hierarchically organised. Several areas (F4d, F4v and F5p) in the dorsal part of the ventral premotor cortex appear to be responsible for sensorimotor transformation and the production of coordinated hand-eye behaviour, while more ventral areas (F5c and F5a) are responsible for higher level decision making and the production of highly trained repetitive behaviours [60]. Although this hierarchy is simplified, it suggests that complex voluntary motor movements are formed from complex combinations and modulations of stereotyped motor primitives. Similar to the behavioural research described above, these physiological findings further suggest that planning can take place at different spatial and temporal scales, from very quick reflexive contractions to long 'high-level actions' that ignore the specific complexities of muscular implementation.

1.7 HIERARCHICAL REPRESENTATIONS IN ALGORITHMIC PLANNING

The last section described the nature of hierarchical map representations demonstrated in human behaviour so far as they are known. This section will describe how hierarchical elements are typically learned and used in planning/navigation algorithms.

1.7.1 *Hierarchy in Markov Decision Processes*

If we consider MDP solving—and behavioural planning more generally—as the process of searching for a nearly optimal solution to a problem in a very large space of potential actions, it becomes clear that as that space grows, the process of searching for a solution becomes more difficult, until the problem becomes computationally intractable for a naive planner.

For an algorithm to successfully obtain its goals within a very large state space, it must either find a way to shrink that space or to search through that space more efficiently. An example of the former is the set of algorithms that try to abstract away irrelevant state attributes to describe the task in the minimum number of states [61]. An example of the latter is hierarchical planning algorithms.

Several hierarchical planning algorithms operate using the MAXQ graph created by Dietterich [62] as a base. This can be seen as a sort of predecessor to Sutton and Barto's Option Framework. It presents any task/environment-space as a set of subtasks. Effectively it's a graph with two types of nodes: Max nodes and Q nodes. Max nodes denote subtasks, Q nodes denote actions that may fulfil the subtasks. It is possible to calculate a hierarchical policy based on this graph. However, Dietterich like Sutton & Barto used a hand-produced graph for the particular Taxi task that he was interested in. Elfwing et al. 2007 then use genetic programming to extend this work by encoding the MAXQ graph structure as a genome and using genetic algorithms to evolve the MAXQ graph towards the most useful representation of the task structure [63]. The MAXQ graph lends itself to this ap-

proach because it keeps some value information even if the structure of the tree alters, so that evolving the tree does not require the agent to lose all progress. A separate body of work by Mehta et al. 2008 used dynamic Bayesian networks to accomplish the same objective of discovering useful MAXQ graphs and therefore useful task hierarchies [64]. Given a task, they assume that a similar task has already been solved and the various elements of that task modelled using dynamic Bayesian networks, and use the successful trajectory and models to solve a similar problem faster. They annotate the successful trajectory with various pieces of information about how different actions in the successful trajectory changed environmental/state variables to create a "causally annotated trajectory". They then scan this trajectory and partition it to find useful subtasks, using the causal information that has been annotated to it.

A relative of MAXQ that is more relevant to the work in this thesis is the option framework, defined by Sutton and Barto in their 1999 paper "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning" [65]. The original formulation of the option framework speeds up the planning process in an MDP through the provision of *Options,* little chunks of policy that can be predefined by the coder or learned/imported from previous tasks. Each option consists of an initiation set (states where the option may be used), a termination set (states where the option ends and the agent starts choosing actions normally again) and a policy (a set of state-action combinations that moves the agent towards a state in the termination set).

Hierarchical Reinforcement Learning, as seen in Botvinick 2009 [13], implements options in order to allow the agent to explore the environment in a faster and more rewarding fashion. Early papers show that this worked but the technique was limited by the necessity to provide hand-crafted options. Accordingly, a body of research developed over the last fifteen years to find efficient ways of learning useful options.

The two main approaches can be summed up as sub-goal based or policy based.[14]

1.7.2 *Subgoal-Based Option Discovery*

Subgoal-based option discovery uses various techniques to identify useful states as subgoals. These states are usually either bottlenecks (states that connect together two otherwise-isolated sets of states) or frequently visited states. Having identified these subgoals, a secondary reinforcement learning process is used to create a policy for reaching those subgoals, rewarding the agent with a pseudo-reward when it reaches those subgoals.

A well-known example of hierarchical reinforcement learning based on subgoals is Botvinick et al. 2009 [13]. Botvinick et al. considered the task of an agent moving through a discrete grid world that contained four rooms that are connected by four doors (taken from Sutton et al. 1999 [65]). The goal was located somewhere within these four rooms, and Botvinick et al. recorded the number of steps that the agent took to reach it in each episode.

Botvinick et al. added two 'options' in each room. Each option was composed of three elements: a set of states where the option could be selected (instead of selecting an ordinary action primitive), a set of weights that determine how likely the agent is to choose a given primitive action in each state while the option is active (essentially an option-specific policy), and a state that would cause the option to end when it was reached. An option can here be thought of as an extra policy that can be overlaid on top of the agent's usual policy when the agent reaches an initiation state and controls the agent's actions until it reaches the termination state.

In the case of Botvinick et al. the doors between rooms were manually designated as being subgoals and were therefore made into the termination state of the option. Furthermore, Botvinick et al. created

14 Note that although these papers usually specifically reference the Option Framework or another hierarchical formulation, the techniques described are usually more general. This makes sense as they are essentially ways to find useful substructures within a state space.

a pseudo-reward that the learner received on reaching the option's termination state. This pseudo-reward *only* affects the weighting of actions *within* the option. In other words, each option essentially comprised an explicit reinforcement learning problem within the overarching reinforcement learning problem, training the option weights (the option-specific policy) to efficiently move the agent from the initiation states to the termination states.

Botvinick et al. designate the termination states manually to suit their task, and so although this work nicely illustrates subgoal-based reinforcement learning it does not show the process of subgoal discovery. An example of the latter is given by Taghizadeh and Beigy 2013 [66], who use graph analysis techniques to algorithmically identify bottleneck states before learning option policy using pseudo-reward in a similar fashion to Botvinick et al.

Several papers have investigated the combination of deep reinforcement learning with subgoal-based options [67–69] and demonstrated promising results in challenging environments like Minecraft and various Atari games. However, like Botvinick et al. 2009 these papers use pre-defined subgoals, and the question of subgoal discovery is considered to be an open challenge [69–71].

1.7.3 *Policy-Based Option Discovery*

Whereas subgoal-based option discovery focuses on the termination of an option, identifying a useful sub-goal and then learning an option policy that can lead the agent to that subgoal, policy-based option discovery ignores subgoals and tries to identify useful sub-policies directly, rather than designating sub-policy learning as an explicit reinforcement learning problem within the overall reinforcement learning problem. One common approach is to search many solutions for common policy elements. Girgin et al. apply this approach by examining the sequences of actions that occur after reaching certain states [72]. They use this data to create a 'suffix tree' which represents all of the sequences following that state as strings of actions sorted by mutual commonality and then search this suffix tree for strings/his-

tories that are 'useful' within any given criteria e.g., highest reward. This gives an option with a defined initiation state, policy, and implied termination state. They then use a classification state to find states similar to the initiation state and add those to the initiation set. It could be argued that this is not quite an option because it finds macro-actions (sequences) rather than policies. This methodology is extended slightly further in subsequent papers, which explore how these trees can be created, grown and used [73, 74].

The PolicyBlocks algorithm [75] performs a similar sort of reasoning. It takes lots of successful solutions to a task and applies a three-step process: it identifies areas of these sample policies which match, it scores them according to how much of the solution they 'explain', and it removes 'explained' areas of policy from the solutions. In this way PolicyBlocks tries to bring about a selection of options which reproduce as much of the matching solution policies as efficiently as possible.

Florensa et al. [71] have produced a hierarchical deep reinforcement learner along policy-based lines by producing a framework for learning skills and applying them in different tasks. Their framework consists of an unsupervised procedure to learn a repertoire of skills using proxy rewards, as well as a hierarchical structure for reusing those skills in future tasks. Skills appear to be represented as policies, similar to other approaches in this subsection.

1.7.4 *Variable-Resolution Maps*

One approach to simplifying planning in large state spaces is to produce maps at different resolutions. For example, one map may describe a single room centimetre by centimetre, another may describe a building room by room. Building-level planning would then take place using the room level map, while navigation through each room would use the centimetre map. It is common for autonomous robotic architectures to decompose planning such that long-term, large-scale planning is performed separately from short-term, close-range plan-

ning and obstacle avoidance [76], with an executive controller interfacing between these modules.

It is also common to represent environments using a low resolution (fixed decomposition) map that abstracts away excessive detail to allow faster and more efficient planning. The 'occupancy grid' representation is a common variant of this that represents an environment using a discrete grid, where each cell is either filled (part of an obstacle) or empty (free space) [76, 77]. This representation is similar to the logic of the grid world simulations used in later chapters.

The occupancy grid representation has two important problems: firstly, the amount of memory required grows in proportion to the size of the environment, and secondly, there is a tradeoff between grid cell size and representational accuracy. Using a large grid cell size allows an agent to use less memory when representing an environment, but may remove salient details and exaggerate the size of obstacles [76]. This is also true for topological maps, which represent positions as nodes and empty space as edges between those nodes. Having fewer nodes further apart reduces memory requirements and allows more efficient planning, but potentially loses information about the environment and how to interact with it at lower levels.

Wiener et al. [55] sketch two algorithms for planning using multilevel topological maps: 'coarse-to-fine' planning and 'fine-to-coarse' planning. In both formulations, an environment is divided into regions. A full 'fine' map is available and so is a regional 'coarse' map. In the 'coarse-to-fine' formulations, a coarse plan is initially generated using the coarse map (i.e., by solving the coarse MPD) and as the agent reaches each waypoint in this coarse plan it produces a local plan to move it to the next waypoint. In the 'fine-to-coarse' planning method, the agent generates a partial map that contains local topology at a fine resolution and far-off topology at a low resolution (e.g., 100 nodes from the local region and one node from each other region) and then solves this MDP.

1.8 THE STRUCTURE OF THIS THESIS

This thesis is intended to model a neural mechanism for cognitive map-based planning, and in particular the formation of hierarchical map representations and their role in the planning process. We have kept the model as biologically plausible as possible in order to investigate how these mechanisms might occur in the brain.

In the next chapter (Ch. 2) we clarify this goal in order to produce a set of concrete aims that govern this work. We then review the most relevant previous models.

Seven chapters of simulations follow the literature review:

The first simulation chapter (Ch. 3) proposes a novel planning mechanism, which uses the relative speed of activity propagation through different areas of the cognitive map to indicate actions that will move an agent towards a randomly chosen goal state. We show how this planning mechanism can operate and how a gating mechanism can read off actions during this process. We also show how this mechanism can operate when synaptic transmission is made noisy, and when the agent's state is defined as a combination of several pieces of information. This model of planning is instantiated as a biologically plausible neural network.

The next three chapters (Chs. 4, 5 & 6) describe how the cell types present in the network self-organise their response properties and connectivity during the random exploration of a simulated 2D grid world environment.

The fifth simulation chapter (Ch. 7) describes how the proposed model can operate when it encounters an unexpected obstacle during planning, and how the proposed model can plan in a environment with nondeterministic state transitions. This chapter then demonstrates that the model can reproduce behavioural results from the Tolman Detour task [2, 5, 21].

The final two simulation chapters (8 & 9) describe an extension to the network that allows it to plan in a hierarchical fashion, similarly to the option framework, and allows it to learn options using a policy-based approach (sec. 1.7.3). Specifically, the hierarchical mech-

anism learns frequently occurring sequences of state-action combinations—essentially specialised sub-policies—and so has more in common with policy-based hierarchical methods than with subgoal-based methods. We show how such a mechanism could self-organise and operate in a biologically plausible neural network model.

The final chapter (10) discusses the strengths and weaknesses of the work described in this thesis, and how it may be of use to modellers and experimental researchers. It gives a programme for continuing to develop the models that we have proposed, and speculates on the future of the field.

2

CONTEMPORARY ALTERNATIVE NEURAL NETWORK MODELS

Section 2.1 below defines the aim of the theoretical work presented in this thesis. Section 2.2 then gives a methodology for carrying out a review of relevant background literature, and the rest of Section 2 reviews the most appropriate papers in turn.

2.1 THE AIMS OF THE PROPOSED MODEL

The core aim of this project has been to investigate the brain mechanisms underpinning cognitive-map based behaviour. To do so, neural networks were developed with the following properties:

1. The proposed network should learn a causal model of the simulated agent's environment purely through unsupervised sensory exploration. The network should then be able to use this causal model to guide intelligent sequential behaviour such as navigation towards a specified state or location. Once this environmental model is learned, the proposed network should be able to produce a novel route from any randomly chosen state to any other randomly chosen state.

2. The architecture and operation of the proposed network should remain broadly consistent with known neurobiology. In particular, it should rely on local Hebbian learning to model synaptic plasticity between neurons.[1] It should also self-organise all of its connectivity from sensory and motor inputs as the simulated agent explores its environment, rather than rely on pre-existing hardwired connectivity or the manual alteration of

1 The proposed model is rate-coded, and so Hebbian learning was used as a time-averaged approximation of Spike-timing-dependent plasticity (STDP).

specified synapses using meta-knowledge of decoded cell representations.[2]

3. The proposed network should achieve aims 1 and 2 while addressing outstanding problems raised by earlier neurobiological models of these functions.[3]

4. Extensions of the network should learn a hierarchy of motor/behaviour primitives, similar to that described by work on hierarchical cognitive maps in humans [49–51], which would allow the model to reproduce important elements of hierarchical map-based planning behaviour [52, 54, 55]. All of the neural mechanisms for implementing this behaviour should self-organise in a biologically plausible manner according to aim 2.

There are several reasons for producing a model that fits the conditions described above:

1. If a model uses a neural network to simulate learning and planning in the brain, it must produce predictions of processes at the neural level. A high-level non-neural mathematical formulation such as Dijkstra's algorithm [78], does not operate on a neural substrate and so cannot directly produce predictions about neural systems.

2. Such a model, if it self-organises using local Hebbian plasticity, would produce predictions about the self-organisation process that could not be obtained from a model which relies on predetermined hard-coded synaptic connectivity. Previous learning and planning networks have proposed the existence of cell types—such as transition cells (Sec. 2.3) and state-action cells (Sec. 2.9)—which are consistent with data from experimental work but whose existence has not been conclusively demonstrated. A self-organising model of these cells therefore demonstrates that these cells can plausibly develop given known cellular learning mechanisms.

2 For example, Friedrich 2016 calculate target synaptic weights from observed state probabilities and update weights to minimise the difference between the actual and desired weights.

3 Discussed further in the review of each model.

3. If such a model were able to address outstanding problems raised by earlier models of its type—those that have already attempted to address some or all of the aims above—it would not be subject to the distortions produced by biologically implausible architectures implementing stopgap solutions to these problems. The behavioural and neural predictions of such a model should therefore be more accurate than the predictions of previously published models.[4]

4. Such a model would show how a Hebbian neural network can learn commonly used sequences of actions to form a hierarchy of temporally extended actions and use them to improve the generation of novel trajectories. Section 1.6 of the previous chapter explains why we believe that this is important. Section 10.4 of the Discussion further explores the issue of hierarchical planning, and suggests further modelling and experimental work for investigating this issue in the light of our results.

2.2 PRODUCING A LITERATURE REVIEW

A review process was carried out to find models that addressed aims 1 and 2. In other words, to review models that addressed map-based planning by modelling neurobiological mechanisms. We did not include the presence of a neural hierarchical mechanism (aim 4) as part of our search criteria, as such work is rare and would be picked up by a search for aims 1 and 2. This process took place in three stages:

1. Identifying a suitable database and performing a search.

2. Filtering the search results.

3. Reviewing the identified papers.

4 See Section 2.2 for the process used to identify these models.

2.2.1 *Producing the Search*

Producing a sensitive and precise search for models that were comparable to that proposed in this thesis is difficult. The obvious search terms are extremely vague: "action planning", "sequential action selection", "goal-directed behaviour", "navigation" et cetera. Text mining methods (reviewed by Stansfield 2017 [79]) were used to find patterns of terms that characterised appropriate papers.

The initial sample contained 149 papers that had been previously identified[5] as relevant to the topic. Termine[6] was used to analyse the titles and abstracts of these papers and identify key terms. Combinations of these search terms were tested to obtain a search that was sensitive enough to capture almost all previously identified papers whilst precise enough to produce a tractable number of total results.

2.2.2 *Filtering Search Results*

The initial search results returned 2405 papers, of which 693 remained after initial review. Specific inclusion and exclusion criteria were then identified and applied based on the aims described in Section 2.1. Below we list the four criteria we used, in the order in which they were applied. For each criterion, the number of papers that remained after applying this criterion is given in italics.

1. The paper should describe a model whose purpose is to explore an environment and then use what it has learned to move an agent between two randomly chosen states without extensive trial and error: *157*.

2. The model should operate using a neural network of some kind: *113*.

5 These papers had been identified by reading around the area in the usual manner. The method described in this section was used to ensure more comprehensive retrieval of all related material.

6 Termine is an algorithm that extracts a ranked list of terms and phrases from a piece of text based on frequency and significance. The version that was used is available online at: http://www.nactem.ac.uk/software/termine/

3. The model should be tested in a topological environment, demonstrating that it can plan in the presence of obstacles[7]: *56.*

4. The model should learn its cognitive map from experience of the environment[8]: *31.*

2.2.3 *Reviewing the Relevant Papers*

Having read the papers resulting from this search and filtered them according to the criteria described above,[9] 31 papers remained that were comparable to the proposed work. These 31 papers included Cuperlier 2007 [80], Hasselmo 2005 [81], Martinet 2011 [2], Erdem & Hasselmo 2012 [82] and Friedrich 2016 [83], the papers that had previously been identified as being the most directly relevant to the proposed model, and so the search process was deemed to be suitably sensitive.

The quality, aims and relevance of these 31 papers varied. The rest of this section will review the most relevant of these models, listed above.

For the sake of simplicity, the models reviewed in this chapter will be referred to simply as "Cuperlier 2007", "Hasselmo 2005" etc. for the rest of the chapter. Full citations are given in this section.

7 Certain planning mechanisms are inherently unable to cope with obstacles (for example, those that operate by calculating a Cartesian vector between the agent and the goal). Furthermore, topologically complex environments frequently prove to be difficult for un-robust models due to the extra complexity of the cognitive map. For both of these reasons, it cannot be assumed that a model which is able to plan in an open field is able to plan in an environment with obstacles.

8 Papers were only reviewed if the model both formed *and* used cognitive maps. Although models of mapping without planning exist, as do models of planning without mapping, both processes substantially constrain the design of the model. It cannot be assumed that a model which learns a map but does not plan would be able to use that map; similarly it cannot be guaranteed that a planning model which assumes the existence of a certain type of map would be able to learn that map. This is particularly true given the complexity of the mapping substrate in most of the reviewed models.

9 The speed and depth of this reading was sufficient to ascertain whether or not the paper fit the criteria, but only the papers which fit all four criteria were read in depth.

2.3 CUPERLIER 2007

Learning

Cuperlier 2007 produces a cognitive map during random exploration of the environment. The cognitive map is encoded in the synaptic connectivity between *transition cells,* which each encode a deterministic transition between two states (Fig. 2.1). These transition cells themselves belong to 'neighbourhoods'. A 'neighbourhood' consists of a state cell hardwired to a number of transition cells. The hardwired connectivity in each neighbourhood ensures that these cells receive input from one and only one state cell.

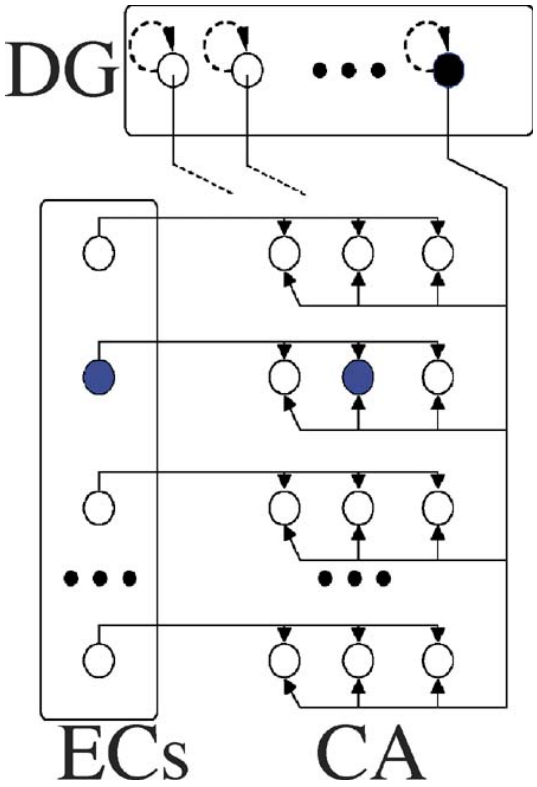


Figure 2.1: "Transition cells (CA) inputs from the neurons coding for current location at time t (ECs) and at time t-1 (DG). In order to have a clear figure, the transition cells layer has only three possible transitions from a given starting place. For the same reason, connections from only one neuron of DG are drawn." Figure and legend reprinted with permission from Cuperlier et al. 2007 [80]

A mechanism exists that allows these transition cells to partially self-organise their characteristic responses. As previously stated, each

transition cell will receive input from a preset state cell according to its neighbourhood. Transition cells also receive input from a layer of 'previous-state' cells, which encode the agent's previous state. The connectivity between the transition cells and the previous-state cells is plastic. In effect, each transition cell will receive simultaneous input from a single (pre-determined) state cell and from a (layer of) previous-state cells. By applying competitive learning at this point, the transition cells (which are already connected to a place cell) will compete to represent the currently firing previous-state cell. This essentially allows the self-organisation of transition cells' response properties.

The network does not model the processes that produce recurrent connectivity between transition cells and so does not model how the environment transitions actually end up being encoded. Instead, the authors find and alter the appropriate synapses algorithmically, setting the synapses between the two relevant transition cells to an arbitrary value after a state transition. The exact nature of this process is described only as "linking transition cells successively reached", and the details of transition cell connectivity are not clearly described—it is unclear whether these synapses are bilateral or unidirectional.

The details of the map learning process is therefore an open question that Cuperlier 2007 fails to address; see Ch. 5 for details of how the model proposed in this thesis addresses this question and Section 10.3.2 for a comparison to other models. The method of self-organising transition cell responses is essentially similar to the state-action cell self-organisation procedure described in Ch. 4, save for the fact that this thesis proposes a two-step self-organisation process that allows state-action cells to self-organise their connectivity to both inputs whereas Cuperlier 2007 hardwires one of these inputs and so allows a simpler learning process.

Planning

Cuperlier 2007 plans by *propagating activity from a goal* backwards through the cognitive map encoded by the *transition cells* (Fig. 2.2).

At each time step, the firing rate of each cell is set equal to the magnitude of the largest input it is receiving. The use of this 'max propagation' mechanism prevents the complex dynamics that would otherwise arise once activity spreads through the layer from interfering with the planning process. The max mechanism does not seem to be consistent with known neurobiology.

Once the activity in the layer is stable, a layer of *gating cells* reads the relevant part of the cognitive map and then activates the motor output. Although this gating mechanism is described in neural terms, gating cells are activated algorithmically and their synaptic connectivity is not modelled. This is a weakness of the model, and it is an open question if and how such cells could develop in the brain. The model proposed in this thesis addresses that question in Ch 6.

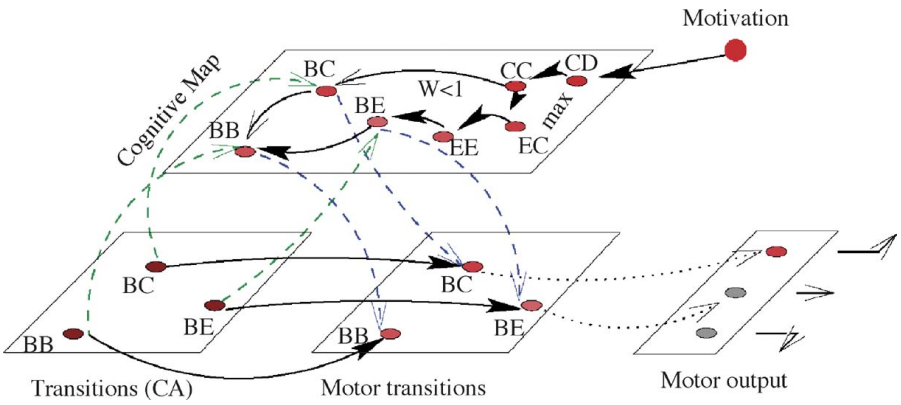


Figure 2.2: "Interaction between the cognitive map and the transitions. Robot is in "B" (transition "BB"). Goal is in "D." The motivation activates all possible transitions arriving in "D," here "CD." This activity diffuses on the cognitive map according to described algorithm. Activity in "BC" is higher than in "BE" or "BB." In the same time, in CA different transitions from "B" are predicted ("BB," "BC," and "BE"). They activate the corresponding motor transitions. The bias coming from the cognitive map enhances "BC" leading to the corresponding motor command." Figure and legend reprinted with permission from Cuperlier et al. 2007. [80]

Cuperlier 2007 only outputs an action once activity in the network is stable; this is likely to be once activity has spread through the entire network. This would make planning time effectively constant, regardless of path length, something that does not fit with known data (see Sec. 9.1.6).

The model is Markovian and seems likely to have the inherent limitations associated with that: an inability to distinguish hidden states[10] and an inability to perform tasks that have looping solutions. The model also encodes the state space using discrete representations of unique state-state transitions, requiring ever larger numbers of cells to encode larger state spaces.

Cuperlier 2007's use of transition cells to encode the cognitive map means that it likely to require an extremely large number of cells to represent the map if state transitions are probabilistic rather than deterministically leading to one successor state. The model proposed in this thesis encodes state transitions in the synapses between state-action cells[11] and so any set of transitions between those states can be represented provided that all state-action cells are present. The number of cells that are required to encode any set of state transitions in an environment is therefore equal to the number of state-action combinations in that environment:

$$N^M = N^S N^A \tag{2.1}$$

where N^M is the number of cells required to encode a map, N^S is the number of possible states, and N^A is the number of possible actions.

By contrast, in Cuperlier 2007, each state-state transition must be encoded by a dedicated transition cell and so the number of transition cells required to map an environment will increase in proportion to the number of possible transitions:

10 Hidden states occur when the agent has incomplete access to information about its environment, so that they have to deduce their true state from limited observations. Tasks involving hidden states are an important generalisation of Markov Decision Processes, usually known as partially observable Markov Decision Processes (POMDPs), which relax the assumption that states are fully observable [84]. A POMDP is usually stated in terms of observations (sensory information received from the environment) and belief states (a probability distribution estimating the agent's true state given the observations received). POMDPs are frequently considered as a Bayesian estimation problem [85].

11 See Sec. 7.2 for planning with probabilistic transitions.

$$N^M = N^T \tag{2.2}$$

$$N^T = N^S N^A N^{S'} \tag{2.3}$$

where N^T is the number of potential transitions in the environment and $N^{S'}$ is the number of potential successor states (i.e., the number of states that may be reached from a given state-action combination). Since the number of transitions increases as the number of potential successor states increase, and the number of successor states will always be greater than or equal to 1, we see that for transition cells:

$$N^{S'} \geq 1 \tag{2.4}$$

$$N^M \geq N^S N^A \tag{2.5}$$

and so the number of transition cells (eq. 2.5) will always be greater than or equal to the number of state-action cells (eq. 2.1). Even if a state-action combination only has two possible results, it will need two transition cells to encode it; in more complex problems the number of transition cells would become unmanageable. This is a weakness in the model, as it is even less able to encode realistic state spaces than other models with discrete representations of the environment. The model proposed in this thesis partially addresses this problem (see Chapters 4, 5 and 10) by encoding state transitions in the synaptic connectivity between state-action cells.

2.4 HASSELMO 2005

Learning

Hasselmo 2005 forms a cognitive map by alternating the network between learning and planning modes of operation. At every timestep, the model will run a planning phase and attempt to find the optimal action for the current state. In practice, this initially produces a ran-

dom action. The model then runs a learning phase to encode the state transition that has just occurred. As the cognitive map is formed, the model's behaviour during the planning phase becomes less random and more directed towards the goal.

The cognitive map is encoded in the synaptic connectivity within and between "neocortical mini-columns". Each column responds to a predetermined state or action, and all of these columns share the same structure: stereotyped populations of cells that mediate interactions between columns in different ways. Each column (whether a state column or an action column) has a population of *column-input cells* that receive topological input from other columns and a population of *column-output cells* that transmits topological output to other columns. In this way columns are considered to be repeating functional units. Figures 2.3 and 2.4, reprinted with permission, show the intra-column and inter-column architecture described by the paper.

Because the network contains both action and state columns, the topological synapses between columns encode not only the connection between states but also the action that causes movement from one state to the other. Specifically, the column-input cells in a state column collectively encode all the actions that could move an agent into that state, while the column-input cells in an action column collectively encode all of the states in which that action could take place.[12] Similarly state-column-outputs encode all possible actions that could be performed in that state and action-column-outputs encode all possible states that could be reached by performing that action. Column-input and column-output cells can therefore be thought of as a form of state-action cell. Note that this is a somewhat wasteful representation, as the model requires 4 copies of each state-action cell, in contrast to other models including that proposed in this thesis.

However, the connectivity between columns only partially encodes the topological structure of the environment. Inter-column connectivity encodes that certain state-action combinations are possible but does not encode the transitions caused by these combinations. The role of representing state-action consequences appears to be filled

12 Note that there is only one action column for each action.

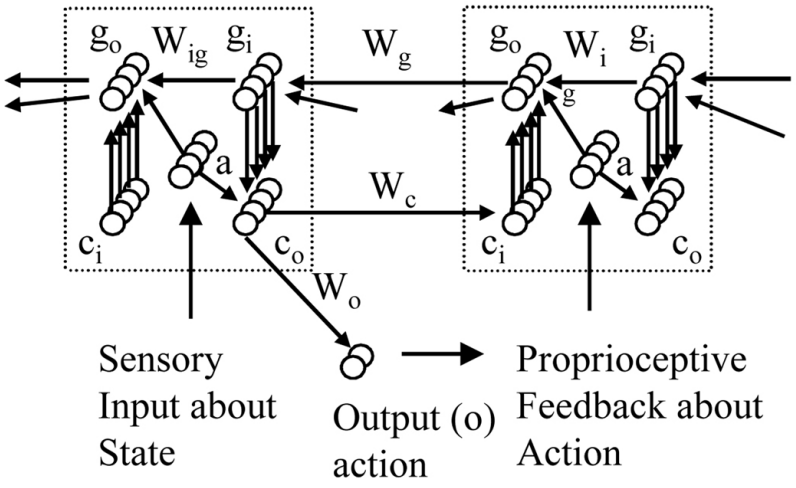


Figure 2.3: "Components of each model minicolumn. Each minicolumn has multiple units in vector a receiving input from thalamus about current sensory state or proprioceptive feedback about motor output (actions). The population g_i receives input spreading back from the goal via connections W_g from other minicolumns. The population g_o receives activity spreading from the goal via internal connections W_{ig} and sends output to other minicolumns. The population c_i receives forward spread from the current state or action. Population c_o is activated by current input from population a in the same minicolumn, which converges with input from population g_i. Population c_o projects via connections W_o to output (action) units o to generate the next action on the basis of activity in population c_o." Figure and caption text reprinted with permission from Michael E. Hasselmo, 'A Model of Prefrontal Cortical Mechanisms for Goal-Directed Behavior', Journal of Cognitive Neuroscience, 17:7 (July , 2005), pp. 1115-1129. © 2005 by the Massachusetts Institute of Technology, published by the MIT Press.

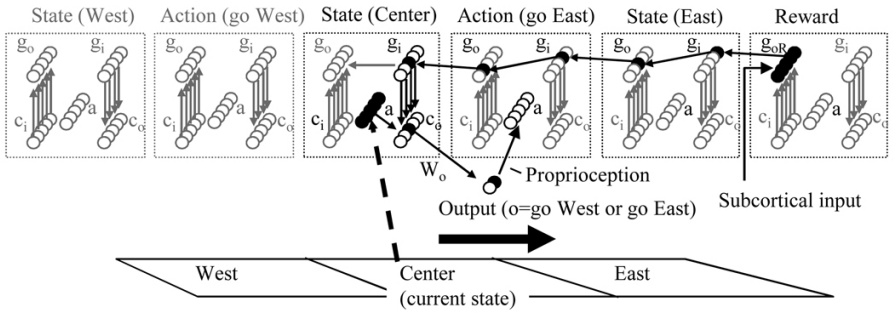


Figure 2.4: ”Retrieval activity in an example linear track task. Separate minicolumns represent the two actions "go West" and "go East", and the three states "West", "Center" and "East". One minicolumn represents goal (reward) in "East". The selection of output action depends on convergence of reverse spread from goal with sensory input to the minicolumn representing the current state (Center). The desire for goal is represented by subcortical activation of the population g_oR in the reward minicolumn, which causes activity to spread back through the network (active units are black). Activity spreads to population g_i and then g_o in the East state minicolumn, then spreads through g_i and g_o in the "go East" action minicolumn before reaching the g_i population of the "center" state minicolumn. In the center minicolumn, the subthreshold input from population g_i converges in population c_o with subthreshold input from population a. The activation of one unit in population c_o spreads over weights W_o to generate the appropriate action via the output vector o (which has two elements representing "go East" and "go West" actions.) Here, the activity of the output vector guides the rat to go East, and also returns as proprioceptive feedback to activate components of the a vector in the minicolumn for "go East. "” Figure and caption text reprinted with permission from Michael E. Hasselmo, ’A Model of Prefrontal Cortical Mechanisms for Goal-Directed Behavior’, Journal of Cognitive Neuroscience, 17:7 (July , 2005), pp. 1115-1129. © 2005 by the Massachusetts Institute of Technology, published by the MIT Press.

by intra-column connectivity connecting the column-input cells of a column to the column-output cells of the same column. To be more specific, imagine an action column that encodes action a. Performing action a produces a transition from state s to a successor state s'. Because a is possible in s, and because s' can be reached by performing a, the action column for a will receives inter-column input from state column s', and send inter-column input to column s.

However, action column A is also receiving inter-column input from lots of other states and sending inter-column input to lots of other states. To encode the specific transition $T(s, a, s')$ requires intra-column connectivity to be learned correctly. Specifically, the column-input cell from state s' has to be connected to the column-output cell to s. This will route activity from s' (and *only* activity from s') to s. This intra-connectivity allows many transitions to route through the same action and state columns.

An analogy is a telephone switchboard. Inter-column connectivity learns what actions and states are theoretically able to interact with each other, and intra-column connectivity shows how actions and states actually *do* interact with each other.

The learning process that shapes the synapses *between minicolumns* seems to do so using a Hebbian learning rule. A memory 'buffer', essentially a clamped input, is used to maintain activity in the previously active minicolumn. Two minicolumns are therefore active, one because it is being stimulated by current sensory/motor output and the other because it is being stimulated by the memory buffer. The column-input cells in the previous minicolumn are stimulated by the buffer and also (because the connections between columns are all-to-all with low weights) receive input from the new minicolumn. The upshot of this is that the agent learns what actions are available in each state, and what state can be reached using each action. Fig. 2.5, reprinted with permission, describes this process.

The learning process that shapes connectivity *within minicolumns* also uses the memory 'buffer' described above. This clamped input makes the column-output cells of the previous minicolumn active in some manner; meanwhile the column-output cells of the current

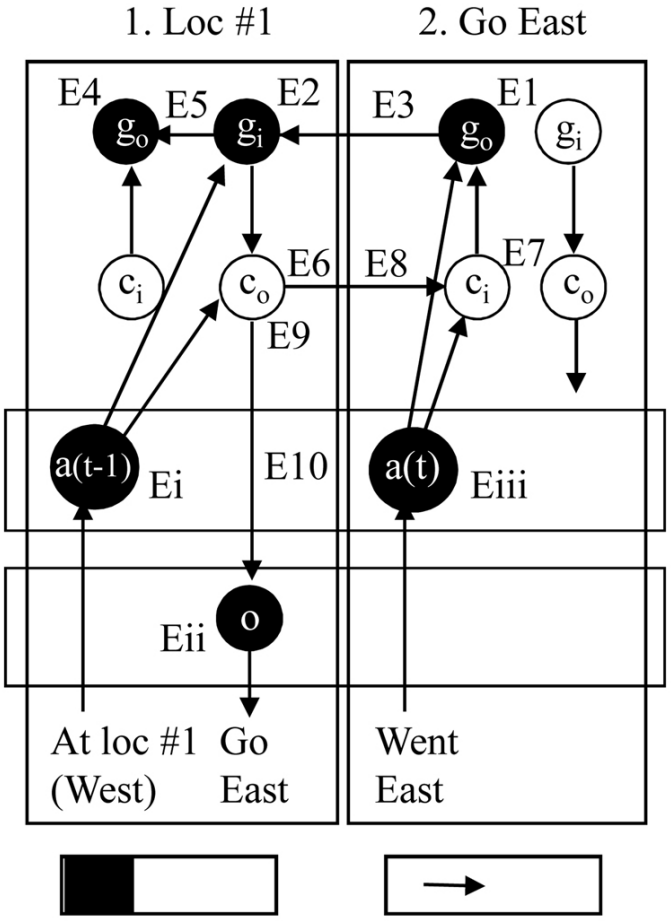


Figure 2.5: "Encoding of necessary connections between a state and the action randomly generated at this state. Each step of encoding is listed numerically. Ei. A buffer in the state minicolumn holds activity from previous input $a(t_e - 1)$. Eii. A randomly generated output o generates a movement. Eiii. The proprioceptive feedback of this output causes activity $a(t_e)$ in the action minicolumn. E1. This activates the g_o population (equation E1). E2. Activity spreads to the gi population in the state minicolumn. E3. Connections W_g are modified between g_o and g_i. E4. Buffering of previous activity $c_i(t_e - 1)$ activates g_o in the state minicolumn. E5. Connections W_{ig} are modified. E6. The buffer of previous input $a(t_e - 1)$ causes activity in c_o. E7. Activity in c_o causes activity in c_i. E8. Connections W_c between c_o and c_i are modified. E9. Activity in g_i causes activity in c_o. E10. Connections between c_o and output vector o are modified." Figure and caption text reprinted with permission from Michael E. Hasselmo, 'A Model of Prefrontal Cortical Mechanisms for Goal-Directed Behavior', Journal of Cognitive Neuroscience, 17:7 (July , 2005), pp. 1115-1129. © 2005 by the Massachusetts Institute of Technology, published by the MIT Press.

column are also active. The previous minicolumn is therefore receiving column-input activity from the current minicolumn and column-output activity (in some manner) from the pre-previous (current time - 2) minicolumn, which allows Hebbian learning to connect the firing column-input cell and column-output cell in the previous minicolumn correctly. Fig. 2.6, reprinted with permission, describes this process.

The structure and connectivity of the network in Hasselmo 2005 is extremely complex, as is the sequence of events that must occur for correct learning of and planning within an environment. It is also explained briefly and unclearly in the paper. We have therefore described it as far as we are able. Without a perfect understanding of the model's structure and operation, it is difficult to study its mechanisms or its resemblance to other models described in this literature review. The structural complexity of this model has the consequent disadvantage that it requires large numbers of cells to encode even a small state space.

Planning

Hasselmo 2005 plans by manually connecting a 'reward' minicolumn to the state minicolumn encoding the reward location. Activity propagates outwards from that state column through the synapses between columns. This process is mediated by an inhibitory rule that subtracts a certain amount of activation from each minicolumn according to the summed activity in that column. Propagation continues for a set number of timesteps. In effect, this is an implementation of decaying activation planning, as seen in Cuperlier 2007 and Martinet 2011. The activity propagating through the network decays as it propagates outwards from the goal, so that columns close to the goal have high firing rates and state columns far from the goal have low firing rates. The difference in firing rates between state columns therefore encodes the relative value of the states that they represent. It is therefore vital that the correct firing rates are established and stabilised.

This need for stable, correct relative activities requires Hasselmo 2005's model to wait for a set period of time before producing an out-

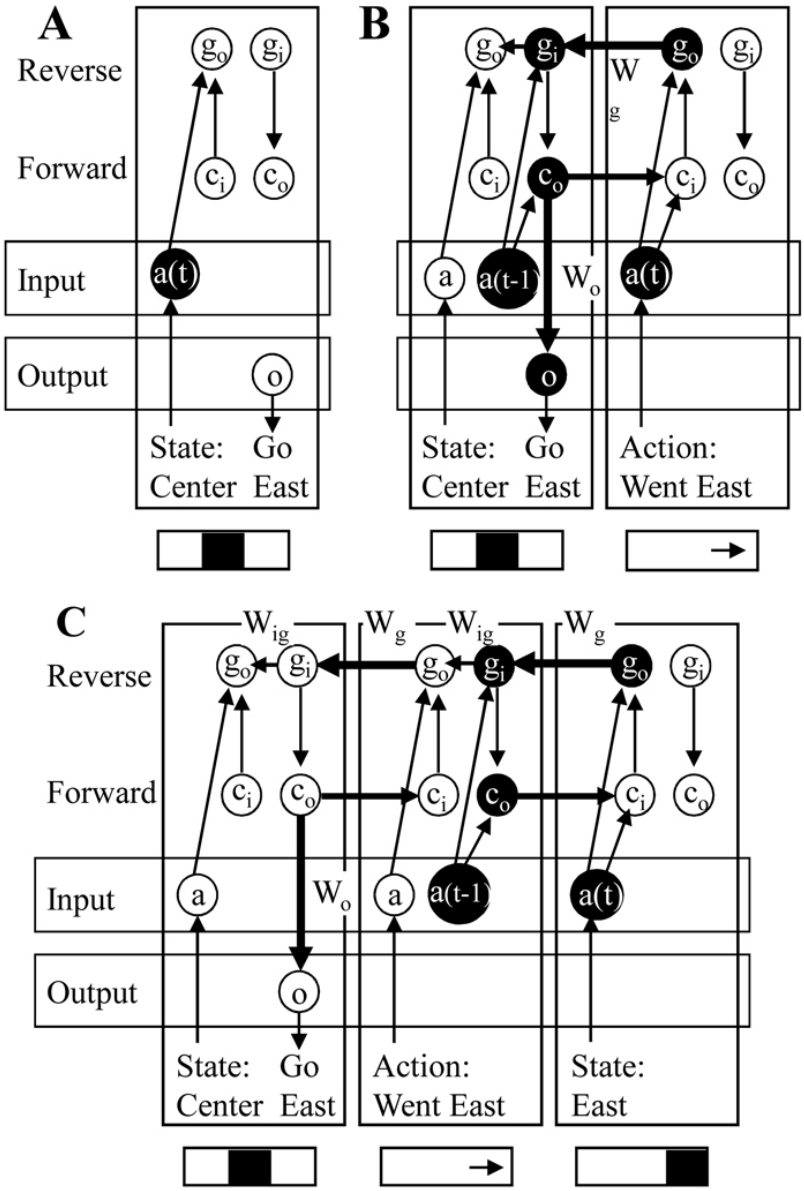


Figure 2.6: "Simple schematic illustration of the steps of synaptic modification during encoding. A. The network receives state input which activates $a(t_e)$. B. An output action is generated randomly. The network receives proprioceptive feedback about the new action $a(t_e)$ which activates g_o in a separate action minicolumn. Connections W_g are strengthened between g_o and population g_i for the previous state, which buffers activity from $a(t_e - 1)$, followed by strengthening of W_{ig} between g_i and g_o. Population g_i activates population c_o, which strengthens connections W_o with the output o. C. The network receives sensory input $a(t_e)$ about the next state which activates a new minicolumn. Connections are strengthened between this new state minicolumn population g_o and the preceding g_i and g_o of the action minicolumn." Figure and caption text reprinted with permission from Michael E. Hasselmo, 'A Model of Prefrontal Cortical Mechanisms for Goal-Directed Behavior', Journal of Cognitive Neuroscience, 17:7 (July , 2005), pp. 1115-1129. © 2005 by the Massachusetts Institute of Technology, published by the MIT Press.

put, presumably to allow firing in the model to stabilise. Cuperlier 2007 uses a similar mechanism. However, this waiting period makes the planning process inherently inflexible. If the waiting period is too short, then the correct firing rates will not have been achieved and stabilised and so the planning mechanism will fail. If the waiting period is too long, there will be a period of wasted time when the correct firing rates have been established but no output has been produced.

In contrast, the planning mechanism used in our model (see Ch. 3) relies on activity propagation time rather than relative firing rates to encode the utility of a state-action combination and therefore moves as soon as a solution is found.

A neural gating mechanism reads off actions from the map. Specifically, *gating cells* exist in each minicolumn which are activated by column-input cells (i.e., activity propagating through the map) and by *state cells* which are activated by sensory input from the current state. These gating cells become active if-and-only-if they receive column-input activity and sensory-input activity. They then transfer this activity to *output cells*, which produce the action. In effect, propagating activity in the map can only produce an output if that part of the map is relevant to the agent's current state. This is similar to the gating mechanism used in our model, with the important difference that our model does not have to wait for the spreading activity to stabilise before outputting it to produce actions. Our model is therefore more responsive, and matches more closely to existing data on planning times (see Sec. 9.1.6).

Hasselmo 2005 is Markovian and uses a discrete state space representation; it is therefore likely to have the inherent limitations associated with these properties. These limitations include an inability to distinguish hidden states and an inability to perform tasks that have looping solutions. The model also encodes the state space using discrete representations of unique states and actions, and so it is unable to perform tasks with continuous state or action spaces, and the number of cells must increase in proportion with the size of the state space.

Possibly for this reason, Hasselmo 2005's model is tested only in very small environments (with up to 15 states and 60 state-action combinations). Mapping larger areas would require larger cell layers and so it is probable that the internal dynamics of the neural activity propagating through the layer might become less controllable in larger environments. The lack of larger-scale experiments represents a gap in this model's explanation of behaviour. In this thesis our model is tested on much larger environments with up to 324 states and 2592 state-action combinations. During early experimentation, we discovered that unstable planning models would start to fail planning tasks as the number of states in the environment reached about 64 states. In particular, self-sustaining groups of active cells (continuous attractors) began to form in open spaces. For this reason it would be important to see Hasselmo's model tested on larger state spaces.

2.5 ERDEM & HASSELMO 2012

Learning

The model developed by Erdem & Hasselmo 2012 learns a cognitive map through an initial period of random exploration.

This cognitive map is encoded using *place cells* and *neocortical minicolumns.* Place cells are assigned to a particular location using either a deterministic triggering mechanism (to ensure complete map coverage) or a pseudo-random triggering mechanism that recruits a new place cell with a probability of 0.001 for each simulation iteration.

Each place cell has a hardwired connection to a minicolumn, which contains a *recency cell,* a *topology cell,* and *a reward cell* (Fig. 2.7). The connectivity within the column is hardwired, and the unique dynamics of each cell type are imposed. The artificial mechanisms that form this model's map substrate stand in contrast to the model proposed in this thesis, which models the complete self-organisation of the map substrate explicitly and as far as possible consistently with known neurobiological mechanisms (see Ch 4).

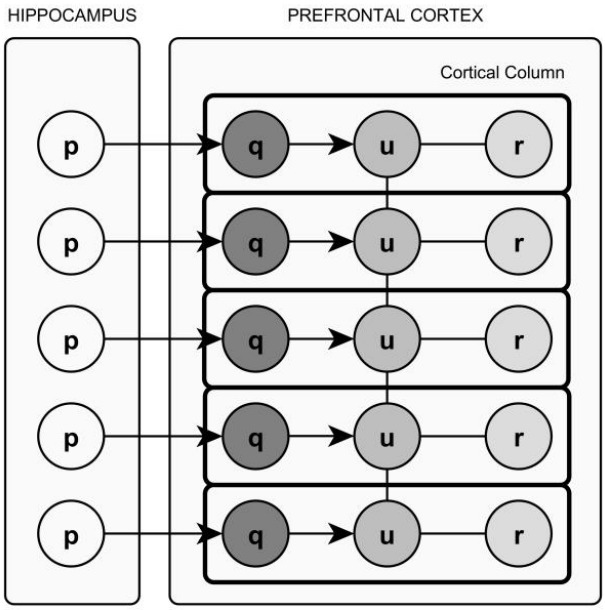


Figure 2.7: "Place cell to PFC cortical column network. The letters p, q, u and r represent, respectively, place cell, recency cell, topology cell and reward cell. The cortical columns are also connected laterally via topology cell layer enabling emergence of a temporal neighborhood of the experienced environment. In this illustration, only a subset of possible lateral connections are shown. All connections without arrowheads are bidirectional." Reprinted with permission from Erdem and Hasselmo 2012 [82].

Bilateral connectivity between topology cells in different mini-columns encodes the cognitive map. The network forms this connectivity using a learning rule that might be best described as semi-Hebbian. The learning rule itself (as given in the paper) does not incorporate any firing information related to the topology cells that are being connected together. Instead, the equation used implicitly combines activation, competition and recency cell activity to connect topology cells together as a function of a "recency signal".

This implicit modelling seems to presuppose that recency cell activity would activate the appropriate topology cells in certain ways, that competition could be applied to the topology cells such that their firing came to encode the desired function of their inputs, and that Hebbian learning could then connect the still-firing topology cells together in such a way as to obtain the desired connectivity. Essentially, the learning rule implicitly describes an extended sequence of mechanisms that are not explicitly modelled. Since these mechanisms are not explicitly simulated it cannot be assumed without reservation that they would operate as hypothesised and without unforeseen side-effects. The possibility that—if these mechanisms were implemented explicitly—Erdem & Hasselmo 2012 might be unable to learn a cognitive map is a limitation in the published work and stands in contrast to the network proposed in this thesis, where the self-organisation of the cognitive map is modelled and tested in detail over the course of Ch. 5.

Planning

Erdem & Hasselmo 2012 combines two mechanisms for planning: a *backwards propagation of activity* through the cognitive map combined with a system of *linear look-ahead probes* based on grid cells. Essentially, the linear look-ahead system plans by shifting a bump attractor from the agent's current position along straight lines in different directions (imagine spokes coming out from a central point, like a bicycle wheel). If the bump attractor interacts with a reward while it is shifting along one of these spokes, then the agent will move along that spoke until

it reaches the goal. This movement is produced algorithmically; the process that checks whether the bump attractor has indicated the reward and then generates a movement of the simulated agent in the correct direction is not explicitly modelled using a neural network architecture.

There are many reasons why the look-ahead mechanism might not be able to successfully reach the reward: there may be obstacles in the way, or the reward might be out of range. In this case the model will propagate activity backwards through the cognitive map to create a gradient of reward that is high for states close to the goal and low for states further away. The model then uses the look-ahead mechanism to find the direction that produces the maximum reward. More specifically, the bump attractor will move along several spokes in different directions, as described in the previous paragraph. A non-neural algorithmic mechanism records the firing rate of each place cell that the bump attractor interacts with during this process and identifies the place cell with the highest firing rate; the same mechanism then ascertains the direction of that place cell's receptive field in relation to the agent's current position and moves the agent towards that location, effectively performing iterative hill-climbing.

Erdem and Hasselmo 2012 do not implicitly or explicitly model any neural mechanism for outputting an action using the look-ahead mechanism in either the simple or the hill-climbing case. This leaves open the question of how this model could output action in a biologically plausible manner. In contrast, the model proposed in this thesis explicitly describes such a mechanism, and demonstrates that its action read-out mechanism can self-organise using Hebbian plasticity rules (see Ch. 6).

The model's planning time is dependent on several factors: the range of the look-ahead 'spokes', the size of the angle between spokes, and to some extent the current distance to the goal. In general, the model should take longer to make decisions if the environment is complex and the path distance is long.

Unlike any other model discussed here, the linear look-ahead mechanism allows the network to produce shortcuts through unexplored

space, a result which has been experimentally demonstrated by Tolman and numerous other researchers. However, the utility of this mechanism is limited by its implementation. The look-ahead mechanism only looks ahead in straight lines. Each look ahead is a straight line laid over the top of the maze, ignoring obstacles. Because the look-ahead mechanism pursues reward whilst ignoring obstacles, it is likely to direct the model straight into a wall.

This problem is depicted in Fig. 2.8. If an agent in the marked location extends two look-ahead probes of the same length, the probe with highest reward would direct the agent into the nearby square obstacle. The agent described in the paper has no means to detect when a probe passes through an obstacle like this and, therefore, would select this movement. A two-step straight-line solution is possible by moving straight south and then straight east, but the agent would not select this sequence of movements unless the higher-reward probes that pass through obstacles were cancelled by external oversight, as they are in the paper. The problem could be solved in this case by reducing the length of the probes, but this would also reduce the model's ability to detect shortcuts through unexplored space, which is the main property that Erdem and Hasselmo 2012 are investigating. Furthermore, there is no way of determining a 'good' probe length for different environments. Probes that are too long for an environment will produce bad plans, as demonstrated in Fig. 2.8, and probes that are too short will be highly inefficient. The required length varies moment by moment according to agent position and obstacle placement.

In other words, the model as specified will struggle to plan reliably in environments that contain obstacles, unless its planning abilities are significantly restricted.[13] While the look-ahead mechanism is interesting, and may produce more capable models in the future, this is a clear flaw in the presented work. Using very short probes, this model would produce similar results to non-hierarchical versions of the model proposed in this thesis, but the model proposed in this thesis is mechanically simpler and explicitly models the entire plan-

13 The authors recognise this problem and manually prevent the network from performing probes in directions blocked by obstacles.

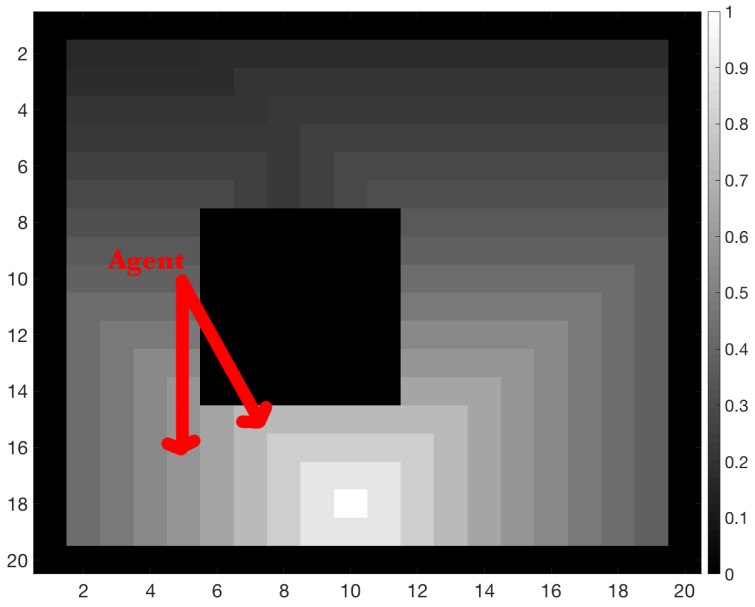


Figure 2.8: The problem of planning with linear probes. In this figure, the reward gradient has been calculated and is displayed in grayscale: white indicates the rewarded location (max value), black indicates obstacles (no value) and the grayscale indicates the gradient.

ning process at the neural level from input to output in Ch. 3. We also explicitly model the process of map learning in Ch's. 4, 5 and 6.

2.6 SCHMAJUK 2006

Learning

Schmajuk 2006 uses two maps: a full map and a regional map. The low-level map is defined as a network of place cells. These cells represent discrete states and each place cell is initially connected (via a hardwiring mechanism) to all cells representing neighbouring states. The authors refer to this as a 'canvas', assuming an empty 2d space that can be adjusted according to the agent's experience during exploration.

The recurrent synaptic connections between these place cells that form this canvas represent a backward cognitive map of the environment: a positive weight from cell i to cell j signifies that cell j's

assigned state can be reached from cell i's state. This weight is initialised at 1 if these states are adjacent, signifying a hypothesis that these two states are connected. Alongside the full map, the network contains a regional map which represents the environment as a collection of regions, and stores the connectivity between these regions. Two more networks store 1) the associated region for each state and 2) the states contained in each region.

The agent's exploration is curiosity-based. Before exploration, a 'curiosity' goal cell sends synapses to every place cell and guides the agent towards them (see Planning). When in a given state, the agent can observe transitions to adjacent states without actually performing them. If an adjacent state is accessible, the initial weight is raised towards an asymptotic limit; if the adjacent state cannot be reached due to an obstacle, the transition weight is decreased to 0.

After the agent has explored a state, the associated place cell loses its connection to the curiosity goal cell. The agent then moves to an adjacent unexamined state, or, if none are available, to an adjacent space that will move the agent closer to the goal. Furthermore, as states are examined, if a state is found to contain a goal h, the weights $w_{j,h}$ from the place cell j to the relevant goal cell are updated to signify that the goal can be found in this state, while weights $w_{h,j}$ are updated to signify that the agent should move to this state to find the goal.

The updating of all weights appears to be hard-wired, rather than the result of a specific learning rule. The authors refer to a Hebbian learning rule that updates the connectivity between place cells in the low-level map and between regions in the high-level map, and refer to a particular appendix where this rule is to be shown. However, in this appendix the synaptic weights are presented as being simply set to a given value. Voicu & Schmajuk 2002 [86] add a stepwise method for increasing and decreasing the weights between place cells but this method does not use cell firing information. It remains an open question how such a structure could be learned purely from the information available to the relevant cells.

Planning

Schamjuk & Voicu define two planning methods. Both methods rely on spreading a decaying activation gradient through the map in order to measure the distance and direction of the goal. The first methods spreads activity through the map forward from the agent's position until it reaches one or more goal representations; the second spreads activity back from a goal position until it reaches the agent's position (this method is similar to the other models reviewed in this chapter).

The nature of these two methods means that they require different planning times proportional to path length. If the plan is produced by propagating a gradient back from the goal, this gradient will not change as the agent moves. The agent can therefore perform the planning process once, at the beginning of the task, and then complete the task using the stored gradient. The agent can simply consult the stored gradient at each step. Planning time for the goal-backward planning method is therefore proportional to the length of the path.

By contrast, the gradient produced by the agent-forward mechanism becomes meaningless as the agent moves. The planning process must be rerun, and a new gradient generated, every step. The agent-forward mechanism therefore has a decision time that is proportional to the *squared* path-length,[14] whereas the goal-backward mechanism is only proportional to the length of the path.

The authors note several issues with these planning methods. Firstly, unless planning is repeated at every step, the require a large working memory to store the gradient from the agent towards the goal, which contradicts the current understanding that working memory is limited to a small number of pieces of information. Secondly, as noted below by Martinet 2011 and by this thesis, a decaying activation mechanism can only operate over a small number of states before the decaying activation decays to nothing.

The authors' solution to these problems is the regional structure described in the Learning section: when the agent's goal and current

14 Equivalent to running the goal-backward mechanism for every step on the way to the goal.

location lie in the same region, the agent plans using the low-level map exclusively. However, when the agent's current state and the goal are in different regions, the agent plans a high-level route to the goal in regional space and a low-level route to the next region in the full-detail map. The use of a regional map structure allows the maximum length of a path to be reduced by decreasing the number of steps required to represent that path, and the regional path planning mechanism achieves this. For example, a 36-state environment can be broken up into 6 regions contained 6 states each. An agent with a maximum WM capacity of 6 will then be able to represent any path through this environment.

Schmajuk et al. found that, when they tested the hierarchical planning mechanism in a simple one-dimensional environment, found that its decision-making time was proportional to the number of regional steps required for goal-based activity to reach the agent and was therefore proportional to the depth of the hierarchy. They also found that, since place-cells in the low-level map to not connect to cells from other regions, introducing a hierarchical map representation greatly reduced the number of connections required to encode the environment. In a 36-state environment, use of hierarchical representations reduced the number of synaptic connections to a third of those required by a non-hierarchical map representation.

The use of a hierarchical regional representation allows predictions to be made about the regional structure of human cognitive maps. The authors speculate, for example, that associations between place cells within a region are stronger than those between places in different regions. The model uses decaying activity to judge distances between places so as to find the shortest path; therefore if activity decays less within one region than between several regions, the model would consider places within a region to be disproportionately close together. This would replicate the results of Hirtle and Jonides 1985[49] although the experiment is not performed in the paper. Such a regional representation is in contrast to the sequence-based hierarchical mechanisms proposed in this thesis, which produces a more

instrumental hierarchical represention based on an agent's performance during planning.

2.7 MARTINET 2011

Learning

Martinet 2011 models learning and planning in the context of a particular maze (see Figure 7.12). They therefore use a specific training regimen that incorporates forced runs through all possible paths in the maze that they are modelling, followed by several random learning phases and supplemented by stochastic learning phases during planning.

Martinet 2011 encode a backward cognitive map of the environment; such a model encodes how each state can be reached as the result of performing particular state-action combinations. This is in contrast to a *forward causal model* that encodes the reverse information: the predicted results of performing certain state-action combinations. The cognitive map is encoded in synapses connecting *neocortical mini-columns* connected to *place cells.* These mini-columns are a repeating unit. The connectivity within these columns is complex, stereotyped and hardwired, and contains a set of *state-action cells* (Fig. 2.9). Both place cells and mini-columns self-organise, but this self-organisation is produced by an externally-controlled two-phase process of dubious biological plausibility. In the first phase, the model automatically assigns place cells to map locations. The place cell assignment process continues until at least six place cells are active for every state that the model has visited, as verified by an external algorithm. When this happens, the model switches to the second phase, where it assigns mini-columns to states. This is done by algorithmically checking whether any column currently responds strongly to that state; if none are responding, then the authors connect a random column to that state. That column is then connected to all place cells that are firing above a certain threshold.

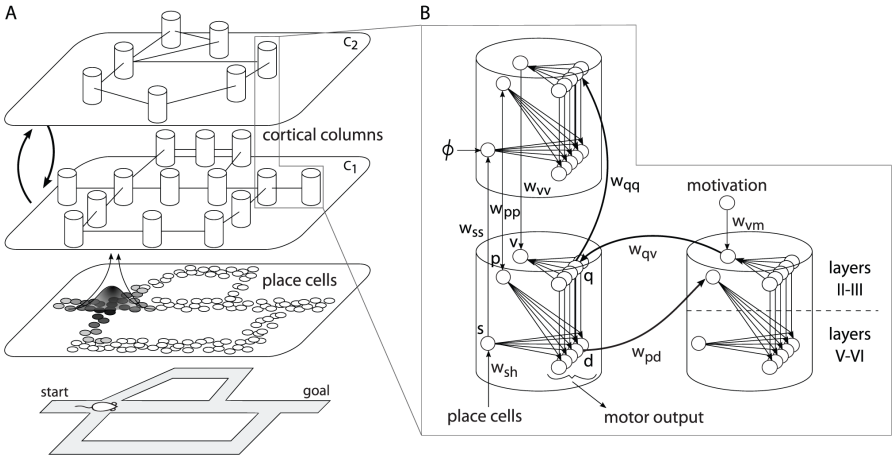


Figure 2.9: "Overview of the model architecture and connectivity. (A) Model hippocampal place (HP) cells are selective to allocentrically-encoded positions. The prefrontal cortex (PFC) columnar network takes HP cell activities as input to learn a sparse state-action code $S \times A$ reflecting the topological organization of the environment. The model employs recurrent excitatory collaterals between minicolumns of two subpopulations (C1 and C2) to implement multilevel spatial processing capturing morphological regularities of the environment. (B) Each model column uses three units s, p, v and a population of minicolumns, each of which is composed of two units q and d. Neurons s receive inputs from HP cells through w_{sh} synapses to encode spatial locations. Forward and backward associations between locations are encoded by w_{pd} and w_{qv} connections, respectively, so that the minicolumn corresponding to the execution of an action in a given place is linked to the place visited after movement. The model uses a motivational signal conveyed by w_{vm} synapses to encode goal information. The population of neurons d projects to motor output, where a winner-take-all competition takes place to select actions locally. Collateral projections between columns (w_{ss}, w_{pp}, w_{vv} and w_{qq}) together with a proprioceptive signal ϕ allow the model to implement multilevel spatial processing." This figure and legend are reproduced from Martinet et al. 2011 [2] in accordance with the Creative Commons Attribution (CC BY) licence.

Once a state transition has been made, the authors add it to the cognitive map by algorithmically altering the connectivity between columns according to whether or not the transition was successful. If the transition was successful, the relevant synapse is set to 0.9. If not, the relevant synapse is halved.

Both the self-organisation of the map substrate and the learning of the cognitive map are heavily managed by non-neural algorithms rather than self-organising biological neural network architectures. This stands in contrast to the model proposed in this thesis (see Ch. 5).

Planning

Martinet 2011 plans by propagating activity from a goal backwards through the topological connections between columns. The activity decays at each synapse, such that an activity gradient forms: the activity of the appropriate cells is proportional to the value of the state-action combination that those cells encode because a cell's activity depends on how close the cell is to the reward location.

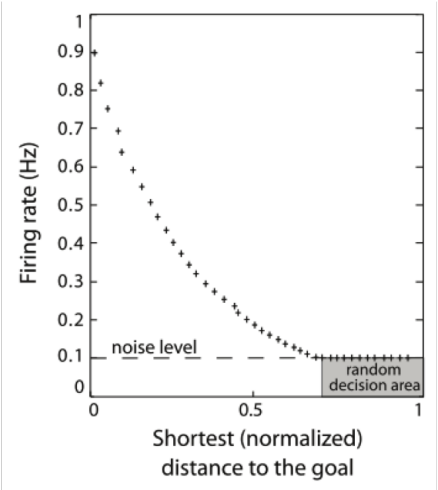


Figure 2.10: Decaying activity limits viable path lengths in Martinet 2011. This figure is reproduced from Martinet et al. 2011 [2] in accordance with the Creative Commons Attribution (CC BY) licence.

As with other models of this type, the fact that the activity decays in proportion to the agent's distance from the goal means that

the activity quickly becomes too small to be useful for planning; in other words, the decaying activity mechanism has a limited range (Fig. 2.10). Martinet 2011 deals with this problem by introducing an extra map at a lower resolution. This map works on the same principles as the base map but essentially merges states that are considered to be functional aliases of each other. This allows activity to propagate over a larger section of the state space before it decays. In practice, two states are considered to be functional aliases (and are therefore merged in the low-resolution map) if the two states lie on a straight line and do not connect to any other states. This works in the maze that the model is tested in because it is formed exclusively from straight line corridors (Fig. 7.12) without wider sections. The mechanism is not demonstrated in any other environment and it is unclear how it would operate in such conditions. This is in contrast to the hierarchical mechanism proposed in this thesis (see Chs. 8 and 9).

Martinet 2011 outputs actions using a neural read-out mechanism. Once the spreading goal activity reaches the agent's current state, it activates a forward propagation of activity from the current state towards the goal. This *forward propagation* appears to be guided by the activity gradient set up by the initial *backward propagation*, and produces action output for every state on the path towards the goal. This mechanism is novel, and seems likely to produce output reminiscent of the forward sweeps described by Johnson and Redish [26]. However, like the hierarchical mechanism described above, this read-out mechanism can only work in a maze that forces the agent to travel forwards or backwards along straight lines. In an open field this activity would propagate forwards in a wave rather than a line and probably produce maladaptive actions[15].

Martinet 2011 uses a spreading activation mechanism and a hierarchy of variable resolution maps. Planning time is therefore likely to be proportional to the length of the path, but this would depend heavily on how the path is discretised in the low-resolution maps.

15 This is a problem discussed in other papers that have tried to replicate forward sweeps [27]).

As described previously, Martinet 2011 would not easily generalise to environments other than the maze that it is tested in, particularly those containing large open spaces. This is because the mechanism for forming and using the low-resolution map that it relies on is dependent on specific properties of its current environment, as is the forward propagation mechanism on which it relies to output actions. In contrast, the model described in this thesis is tested in an open field environment and a 4-room environment (see Fig. 3.6) in addition to the corridor-based detour environment used in Martinet 2011.

2.8 MATSUMOTO 2011

Learning

Matsumoto 2011 [87] learns the cognitive map during an initial period of exploration. This exploration is semi-random, as the agent's actions are partially random but biased in various ways by properties of the environment.

The network encodes a backward cognitive map (see Sec. 2.7) in synapses connecting *place cells* (Fig. 2.12).[16] These place cells have overlapping Gaussian receptive fields (Fig. 2.11), so that each location in the state space is signified by the combined activation of multiple state cells. As the agent moves through the environment, it will move from one receptive field to another, and as it moves between the two receptive fields both place cells will be firing simultaneously. A simple Hebbian learning rule is therefore able to connect neighbouring place cells.

Planning

Having learned a topological map, Matsumoto 2011 uses it to form broad synapses between place cells and a *goal-place cell*. The goal-place cell is based on cells in the rat hippocampus that fire when the

16 The authors do not model how these cells self-organise, but do cite models of place cell formation from the literature.

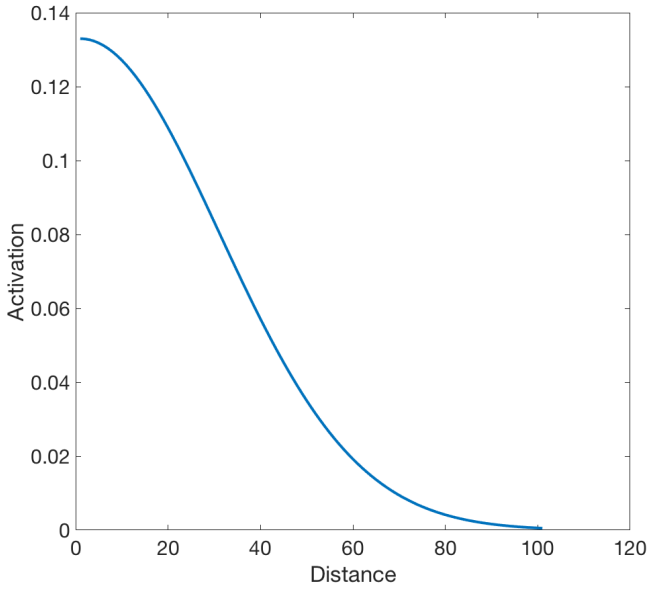


Figure 2.11: *An example of a Gaussian Place Field.* The receptive field of this cell is centred on a particular point in continuous space, and the cell receives activation in proportion to the distance of the agent from this preferred location. Each sensory cell has a Gaussian receptive field so that the overall firing in the sensory layer takes the form of a gaussian 'bump' centred on the location of the agent.

agent is at a certain location and receiving reward; if the agent no longer receives reward at this location then the goal-place cells cease to fire. In Matsumoto 2011, these cells serve primarily to encode the presence of a reward - the location does not appear to matter.

When a task begins, a goal-place cell is stimulated using an external signal that decays over time. At the same time, the place cell encoding the location of the reward is stimulated using another external signal, and activity propagates from that place cell through the place cell map.

After the initial stimulation, the activity of the goal-place cell decays over time. When goal-place cell activity is still high, the propagating activity will not have propagated very far, and so place cells close to the goal will be active. Strong synapses therefore form between the goal-place cell and place cells close to the goal. After a significant period of time has passed, the propagating activity will have propagated further, and the goal-place cell activity will have decayed further.

Weak synapses therefore form between the goal-place cell and place cells far from the goal. At the end of this process, each place cell is connected to the goal cell by a synapse whose weight reflects the place cell's value, i.e., how far that place cell is from the goal (Fig. 2.12).

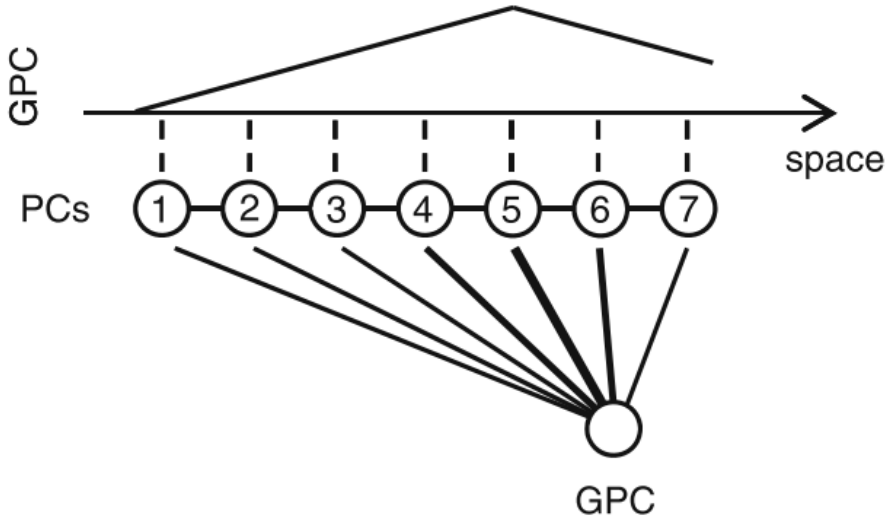


Figure 2.12: "Mechanism to create the distance-dependent GPC activity ... through pre- and post-synaptic-activity-dependent Hebb's rule, the PCs with their place field closer to the goal are more strongly connected to the GPCs. Through these connections, the GPC is activated by input from the PCs corresponding to an animal's current position. As a result, the GPC is more strongly activated when the animal is closer to the goal." Figure and legend reprinted with permission.

Activity in any place cells (representing the presence of an agent) will therefore cause the goal-place cell to fire at a rate that reflects the value of that place cell. The activity of the goal cell therefore encodes the value of the current state, and can be used for (algorithmic) hill-climbing. The process of action-output is not modelled. The process of action read-out is therefore an open question in this model, unlike the model proposed in this thesis (Ch. 6 covers this topic in depth).

The mechanism used to plan in Matsumoto 2011 is notable primarily for its simplicity: the model is able to plan using only known cell types with simple properties. For the same reason it is able to learn a map using simple Hebbian connectivity. Unfortunately, this planning mechanism has behavioural limitations. Every time the model plans any task, it must use the mechanism described above to pro-

duce a cached version of the value function in the synapses between place cells and goal place cells. This planning mechanism relies on Hebbian learning and is therefore likely to take significantly longer than a planning mechanism which relies purely on propagating activity.

We can produce two behavioural predictions from Matsumoto 2011: firstly, the model can only solve one problem at a time.[17] Secondly, it should produce a distinctively long planning time before any action is taken, with all further decision times being constant and unrelated to path length. Neither of these predictions (see Sec. 9.1.6) match experimental evidence [57]. The model proposed in this thesis better accords with the available data.

2.9 FRIEDRICH 2016

Learning

Friedrich 2016 uses a training technique called *parallel sampling* to form a cognitive map. In this technique, the model does not learn during organic exploration of the environment. Instead, during an iteration of the parallel sampling algorithm, a new state is generated for *every possible state-action combination simultaneously* according to the known transition probabilities of those state-action combinations, and their synaptic learning rule is then run on each state/action/new-state combination. If the state space is continuous, parallel sampling is run on 500 randomly sampled states at every iteration.

The architecture of this model is not described explicitly but seems to consist of a single layer of mutually inhibiting state-action cells which encode a backward cognitive map. These cells can encode both

17 Technically, it might be possible to have a variety of goal-place cells, each with their own connectivity and each representing a different goal, so that the model would be able to cache many different plans. However, this is not demonstrated in the paper. It would also be inflexible, requiring the connectivity to be relearned every time a relevant environment variable changed, and it would not allow the model to reach an arbitrary new goal without relearning.

a discrete (or discretised) state space[18], or a continuous state space[19]. The connectivity between these state-action cells encodes the cognitive map.

Friedrich 2016 uses two separate learning rules to learn the cognitive map. These rules correspond to the discrete and continuous cases respectively. Both rules effectively alter the synapses between cells algorithmically to reflect the training data, until they accurately encode the transition probabilities between states.

Friedrich 2016's environment mechanism is unique, relying on the simultaneous generation of a transition for every possible state-action combination simultaneously, and since an agent cannot occupy more than one state at a time (let alone *all* of them) this mechanism is physically impossible. The model also requires a surprising amount of training to solve even simple tasks: >6600 state transitions to navigate a 7x7 state maze and >35,000 transitions to perform well at a pendulum swing-up task. This requirement for large amounts of tightly-controlled training suggests that the planning mechanism may be excessively reliant on the quality of the cognitive map. If this is so, it represents an important limitation in the model's explanation of map learning compared to models which learn using relatively small amounts of random exploration. The model proposed in this thesis achieved ~60% success rates in an 8x8 environment after only 150 transitions, and >90% after 300 transitions, less than one twentieth of the number required by Friedrich et. al 2016 (see Fig. 6.8).

Friedrich 2016 also do not model the formation of the map substrate, and their explanation of map learning is highly artificial. Both of these are open questions that are addressed in the model proposed by this thesis (see Chapters 4 and 5 for details).

18 A space where states and actions are defined by a unique combination of discrete properties (e.g., agent at path intersection 1, turn left).

19 A space where states and actions are defined as a combination of continuous properties such as real-valued coordinates or angles.

Planning

Friedrich 2016 demonstrate mathematically that once the network becomes stable, the summed activation of all state-action cells encoding a particular state will accurately approximate the value of that state. This is true provided that certain conditions are met:

- The weight of the synapses between state-action cells are close to the true transition probabilities.
- A particular level of lateral inhibition operates between state-action cells that encode the same state.

Once the network has reached a steady state, the only state-action cells left active are those encoding the optimal action to take for each state. An algorithmic mechanism reads off the optimal action for the current state. As in other models which use algorithmic mechanisms to perform action read-out, this gap leaves it an open problem whether—and how—a neural mechanism could produce output based on this model. The model proposed by this thesis does not suffer from this limitation (see Ch. 6).

Friedrich 2016 demonstrates high performance in a wide variety of tasks, including several benchmark reinforcement learning tasks. Uniquely amongst the models reviewed in this thesis, the model demonstrates strong performance in tasks which have multiple goals of varying value.

However, several pieces of evidence indicate that the model may lack robustness, and is sensitive to imperfections in the map. The model requires extensive, tightly-controlled training, as discussed above. The modelled tasks have small state spaces, requiring relatively small state-action layers, and it is possible that tasks with larger state spaces might produce more complex dynamics or require unreasonable learning/planning times. The model also seems likely to be vulnerable to noise, a factor which is not described or tested.

Although Friedrich 2016 appears to perform extremely well, it remains to be seen whether the open gaps in the model (self-organisation, map learning and action read-out) will be explored by further work.

It also remains to be seen how well the model would perform in larger tasks or with more limited and ecological training.

3

NAVIGATION TO A GOAL LOCATION USING STATE-ACTION CELLS AND A WAVEFRONT PLANNING MECHANISM

This chapter will begin with a description of the model and its mechanisms. Section 3.1 will show how the model can use a cognitive map to solve grid world navigation tasks. Section 3.2 will show that this model can continue to solve such tasks despite the inclusion of noise in the synaptic transmission of activity between cells. Finally, Section 3.3 will show that this model can solve grid world navigation tasks if the model's position in the grid is represented as a combination of separate pieces of information (x- and y-coordinates). In this initial chapter, we use a model whose synaptic connectivity has been entirely hardwired. That is to say, we show that as long as a certain synaptic connectivity exists within and between the cell layers in this model, then the model is able to plan using a cognitive map. Chapters 4, 5 and 6 will then show how such connectivity might be learned.

3.1 PLANNING IN SIMPLE GRID WORLDS

3.1.1 *Statement of Problem*

In order to solve a navigation task, an agent must use an internal map of its environment to produce actions that move it towards the goal location at the end of the maze. Animal experiments such as the detour task show that even if the layout of a maze is manipulated animals still produce a correct path despite this manipulation, and do so without extensive trial and error. It therefore seems likely that these actions must be generated almost exclusively by use of the internal map. The proposed model must therefore be able to do the same quickly, accurately, and robustly.

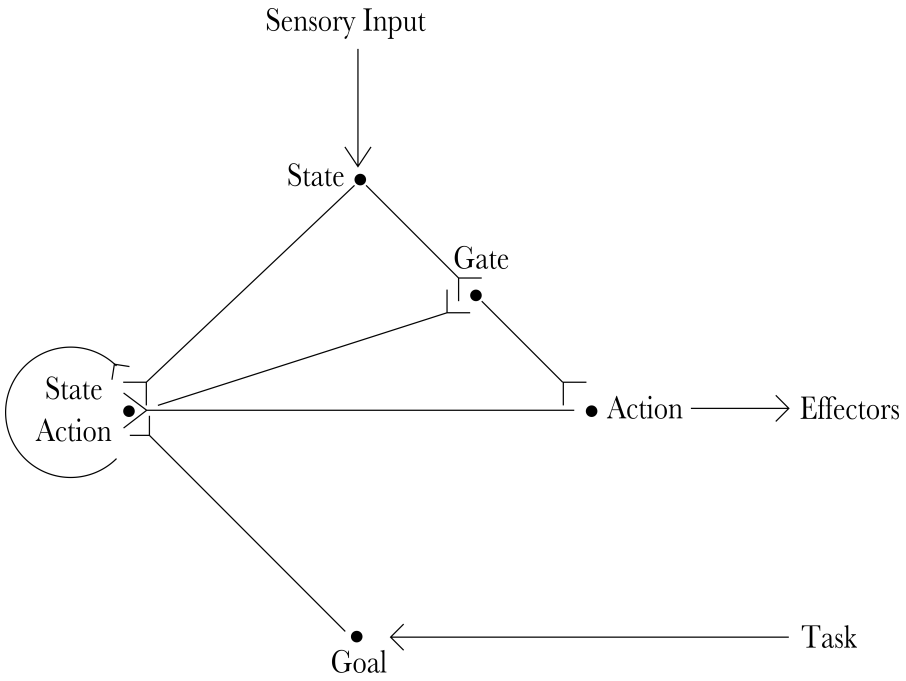


Figure 3.1: Architecture of the Proposed Network Model

3.1.2 *Hypothesis*

MAP REPRESENTATION - STATE-ACTION (SA) CELLS We hypothesise that, for a cognitive map to be useful for the production of actions, it should record how the actions of an agent in different states cause the agent to move between states. In Markovian terms, a cognitive map should encode the state transition probabilities within the environment.

For the sake of simplicity, we initially assume that transition probabilities are always 100%. Ch. 7 demonstrates planning when this is not true.

We therefore hypothesise that the proposed model encodes this information in terms of *state-action combinations* and it follows that the network should encode information using a neural substrate of SA cells, each of which encodes a unique combination of one state and one action.[1]

If SA cells respond to a combination of state and action information, they are therefore most likely to occur in an area which receives high-level sensory and motor feedback. The prefrontal cortex is located in the right part of the sensorimotor pathway and prefrontal cells

1 State-action cells receive state input from *state cells* and action input from *action cells*. See Figure 3.1. At this stage it is assumed that the state space is discrete; that there are a finite number of discrete states and so that each state is uniquely identifiable and separable from any other state. The same is true of actions.

are activated by stimuli from all sensory modalities, before and during a variety of actions, and in anticipation of expected events and behavioural consequences [88]. They are also modulated by motivational state [88]. Prefrontal neurons also output to the premotor cortex, which is known to represent and perform high-level actions [89].

In general, the prefrontal cortex is considered to be responsible for executive function at the top of the motor hierarchy [90]. Lesions of lateral prefrontal cortex produce an inability to generate novel or complex sequences of behaviour, and patients find it difficult even to consciously represent such sequences, indicating that the lateral prefrontal cortex is at least necessary for generating such sequences and may contain the substrate which represents them [90].

More specifically, single-neuron recordings have shown that over the course of a task, cells in the primate prefrontal cortex come to represent and respond to combinations of sensory cues and motor actions by a process of associative learning [91]. Later work has strongly suggested that abstract and hidden states are represented in the orbitofrontal cortex [92, 93] and recent work has found that cells in the rat orbitofrontal cortex encode a mixture of stimulus and choice information about the rat's previous decision [94].

MAP REPRESENTATION - MINICOLUMNS We further hypothesise that these state-action cells are likely to be organised into minicolumns. These are stereotyped structures commonly found throughout the neocortex, usually with approximately 80-100 neurons, and are subject to an array of cortical inhibitory interneurons [95]. This allows a variety of inhibition to occur within and between minicolumns, with very different levels of specificity and suppressive effect. We propose that such inhibition plays an important part in the formation of SA cells (see Ch. 4) and allows activity to be controlled and sustained during planning (see below). *In this thesis, the term 'column' refers to a minicolumn unless otherwise stated.*

MAP REPRESENTATION - ENCODING BACKWARD CAUSAL MODELS USING RECURRENT CONNECTIVITY One way that a layer of state-

action cells can encode the topology of an environment is in the recurrent connectivity between them. If each state-action cell encodes a certain state-action combination possible in the environment, transitions can therefore be encoded in the recurrent weights between cells encoding a state-action and cells encoding the resultant successor state. If this recurrent connectivity is extensive enough, the network as a whole would encode the topology of the state space. Furthermore, activity propagating through the layer of state-action cells would do so according to the pattern of recurrent connectivity and so would be influenced by its topology, allowing planning.

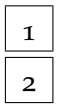
In the model described below, the recurrent connectivity between these state-action cells encodes a *backwards causal model* of the environment; such a model encodes how each state can be reached as the result of performing particular state-action combinations. This is in contrast to a *forward causal model* that encodes the reverse information: the predicted results of performing certain state-action combinations.

The distinction between forward and backward causal models is clearer in the context of an example. Figure 3.2 shows a very simple grid world with only two states, two possible actions, and two (useful) state-action combinations. In this case a forward model records the result of performing the state-action combination *North from State 2* (eqn 3.1), whereas the backward model records a list of state-action combinations that produce the result of *being in State 1* (eqn. 3.2). Another way of thinking about this is that a forward model records how causes produce effects, and therefore models the causal relationship forward in time, while a backward model records how effects are produced by causes, and therefore models the causal relationship backwards in time.

This distinction is significant because it determines what kind of information the model stores and is able to output. A backward causal model is able to give a set of state-action combinations that will lead to a desired state, but cannot output predictions about the result of performing any given state-action combinations. A forward model is the opposite.

North from 2 $\rightarrow$ State 1 Forward Model (3.1)

State 1 $\leftarrow$ North from 2 Backward Model (3.2)



States	Actions	State-Action Combinations
1	North	South from State 1
2	South	North from State 2

Figure 3.2: A Two-State Grid World

A navigating model receives a desired end goal at the beginning of a task and is then required to output actions in order to move an agent from its current state to the desired state. The format of a backwards causal model is much more suited to this task (see eqn. 3.2) and so we hypothesise that the planning mechanism in the brain uses a backward causal model.

As stated previously, we have hypothesised that the model encodes causal relations between states by means of synaptic connectivity between state-action cells. Because we have also hypothesised that these causal relations are encoded using a backwards causal model, the hypothesised connectivity must reflect that. A state-action cell (s, a) that moves an agent into a new state s' therefore receives an excitatory synapse from any state-action cell that encodes that new state, so that activity can propagate *backwards* from the desired state to identify suitable state-action combinations to bring about that state.

MAP-BASED PLANNING A cognitive map is encoded using the strengthened recurrent connections within the layer of state-action cells. The proposed network feeds goal signals into the state-action layer at the goal location and allows these signals to propagate outwards through the topological connectivity of the cognitive map.

This propagation through the cognitive map is the primary element of the planning process. The quickest path to take from the agent's current state to the goal location is that requiring fewest state

transitions. Since each state-action cell represents, in effect, a state transition, the path requiring fewest state transitions will also be the path that activates fewest state-action cells between the agent's current state and the goal location.

A common mechanism in the literature is to use decaying propagation, so that some activity is lost every time activity propagates between state-action cells. Consider a goal signal propagating through the recurrent connections towards the state-action cells associated with the current state of the agent. In this decaying-activation paradigm, the optimal state-action cell for each current state is connected to the goal location via the fewest state transitions and so receives the least-decayed activation; in other words the most active state-action cell for each state will move the agent towards the goal fastest. However, this mechanism has the important limitation that the activity will eventually decay to a negligible level and so this mechanism (in a naïve form) is unable to plan over a sufficiently long distance. This problem is well illustrated in a figure from Martinet 2011 [2], Fig. 2.10, which shows the decline in state action firing rates depending on the shortest distance between that state and the goal. (This figure is reproduced from Martinet et al. 2011 [2] in accordance with the Creative Commons Attribution (CC BY) licence.).

The proposed model uses an alternative mechanism to identify the fastest path to the goal. This mechanism plans using the *timing* of goal-based activity propagation through cells encoding the cognitive map, rather than how much activation those cells receive. We do not hypothesise that activity decays as it propagates through the state-action layer. Instead, we hypothesise that activity will propagate along the shortest path to the goal fastest, because that path involves the fewest state transitions and so the activity does not have to transfer through very many state-action cells. The relative values of different actions in each state—the speed with which those actions move the agent towards the goal—are therefore determined by the temporal order in which those SA cells receive activation from the goal and become active.

The *first* SA cell to receive activation in each state now represents the optimal action to take in each state, and the level of activation is not relevant. This allows the level of activation to be kept at a constant level as the wave propagates through the map, allowing it to plan over longer distances. This mechanism also does not require time for activity to 'settle' in the layer, or for cells to integrate activity. Figure 3.7 displays recordings taken of state-action cells during a simple wayfinding task.

This mechanism owes inspiration to work by Ponulak & Hopfield 2013 [3]. Their paper described a model in which a wave of activity propagates through a 2D layer of topologically organised pure state cells (neighbouring state cells were recurrently connected) and showed that this wave carried information about the direction of the goal. Specifically, if the goal is east of a state cell, the wavefront will 'hit' the state cell from that direction. Ponulak showed that an anti-STDP mechanism could record this information in the recurrent synaptic connectivity of the layer by strengthening the synapses between cells to indicate the direction from which they were 'hit' by the wave.

The mechanism that we have proposed retains the insight that the propagation of a wavefront can carry information about the direction of a goal, but uses this information in a rather different way. Rather than use a propagating wavefront to adjust the synapses between pure state cells and thus produce a synaptic vector field, we have proposed that a propagating wavefront can identify and output the most valuable state-action combination for a given state.

The proposed mechanism is therefore able to improve on the mechanism described by Ponulak & Hopfield in several key ways. Firstly, because the proposed planning mechanism is able to work in a state-action map, it is able to output explicit actions that move the agent towards the goal (see Output below). Secondly, because the proposed planning mechanism is able to perform planning without requiring a period of synaptic plasticity, it seems likely that an agent using this mechanism would be able to plan more quickly, and would not have to 'undo' the new synaptic weights if there is a change in the goal state or the transition structure of the environment. Thirdly, because

the proposed planning mechanism is able to plan without altering the synaptic weights that encode the cognitive map, we can store information in these weights. We show in Chapter 7 that this ability allows the model to store nondeterministic transition probabilities, and to use this information during planning. Finally, we show in Chapter 8 that the proposed planning mechanism is able to interface with a mechanism for producing hierarchical behaviour.

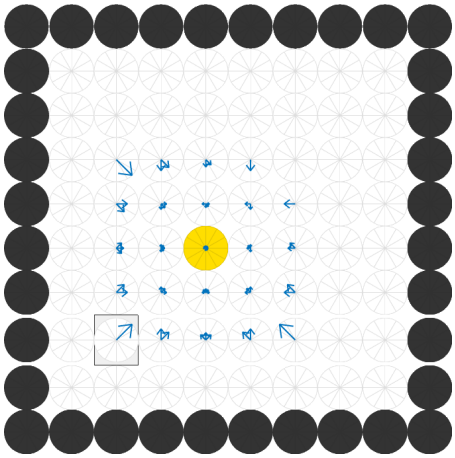
OUTPUT The proposed network is able to read off the optimal action at the current state from the planning process using biologically plausible neural mechanisms. An action read-out mechanism must perform two important roles. It must read out the recommended action for the current state whilst *not* reading out anything associated with any other states, a problem illustrated in figs. 3.3 and 3.4. In the case of the proposed network, this is effected by a set of *gating cells* that propagate activity from the appropriate section of the cognitive map (that which represents the current location of the agent) to the action cells, where it produces motor output. Fig. 3.5 displays the hypothesised connectivity of these gating cells. The gating cells are under heavy inhibition such that they can only produce firing if they are receiving both SA and state input (see eqn. 3.11), meaning that they only pass on activity from an SA cell when that SA cell matches the current state.

3.1.3 *Task*

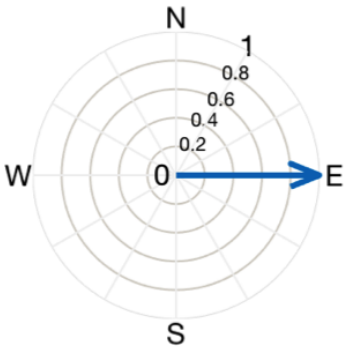
The task is based on a 2D grid world formulation shown in Fig. 3.6. The environment is a 10 by 10 grid, bounded by walls to make it 8 by 8. In each trial a random location within this grid is designated as the agent's goal and another random location is designated as the agent's starting point. The agent can move one space at a time in eight compass directions[2] or stay still,[3] giving nine possible actions.

2 N, NE, E, SE, S, SW, W, NW

3 The action of staying still would not normally occur during goal-seeking behaviour, as by definition it does not move the agent towards the goal. However, the suppression of movement can in itself be a task-relevant action in certain tasks (notably the

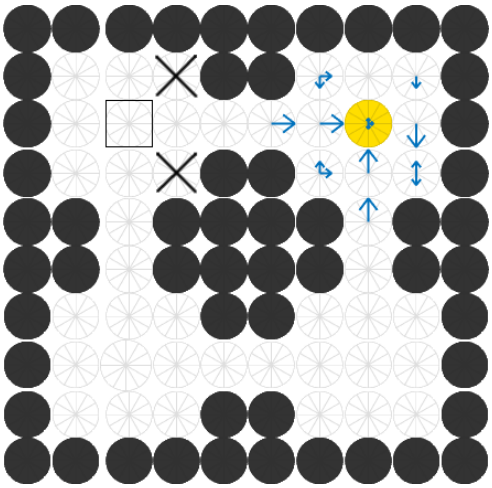


(a) SA Firing Rates During a Navigation Task

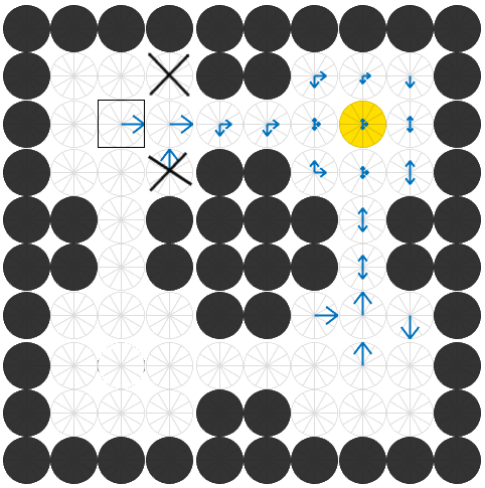


(b) Compass Plot Legend. Each state is represented by a compass plot; each arrow within a compass plot represents the firing rate of a state-action cell that represents moving in the direction of the arrow from that state.

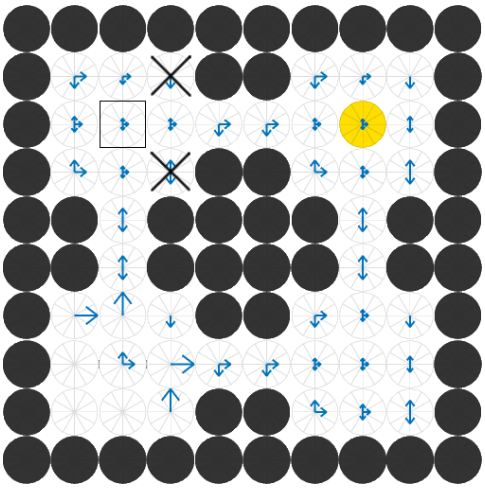
Figure 3.3: Illustration of the problem of reading out the correct action for the current state. (a) shows the firing rates of a layer of SA cells partway through a planning task. Each compass plot in (a) represents the firing rate of all SA cells which encode a particular state as illustrated in (b). The golden state is the goal location and the boxed state is the agent's current state. The planning wavefront (ch. 3) has activated a large number of SA cells, most of which encode an action (NW, SW, W) that will not move the agent towards the goal. The model cannot therefore simply sum or average the activity in the SA layer but has to gate this activity by the current state of the agent before it is passed to the action output layer. Furthermore, this read-out must happen at a specific time (see Fig. 6.2).



(a) T = 3



(b) T = 6



(c) T = 11

Figure 3.4: The Gating Problem. This figure shows compass plots of state-action activity during planning (taken from Fig. 3.7). If the agent is occupying the state marked by the square, then the model should only read out activity from the SA cells marked by the square, and not (for example) those marked by crosses. Also, the agent should not read out activity from the marked SA cells at t = 3 (when the wave of activity has not yet reached the agent's state) or at t = 11 (when the wave of activity has passed the agent's state) but only at t = 6.

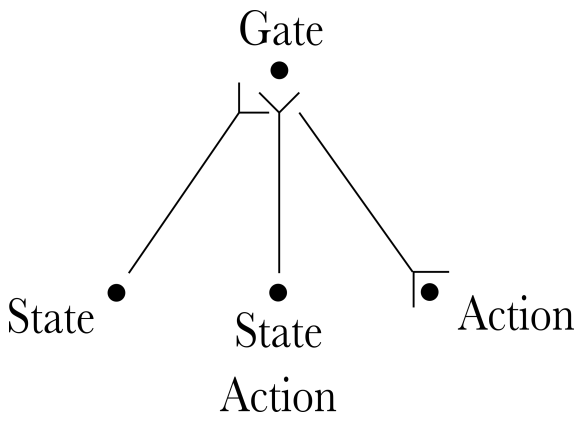
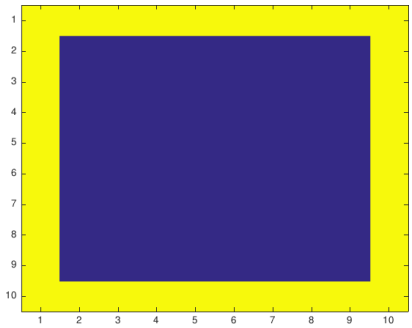
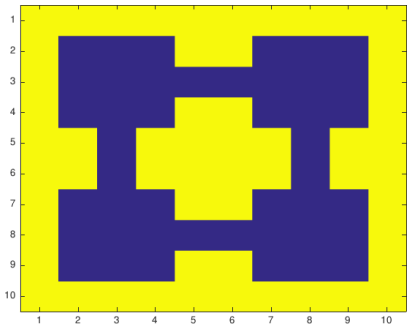


Figure 3.5: Connectivity of a Gating Cell. The gating cell receives afferent connections from state cells and state-action cells, and sends efferent connections to action cells.

The task is for the simulated agent to navigate to the goal location. If the agent reaches the goal location then the task has been been completed successfully. If the agent fails to reach the goal location within 1000 timesteps then the agent has failed the task.



(a) Open Environment



(b) 4-Room Environment

Figure 3.6: The 2-dimensional grid-world state space used to test the simulated network agents. The agent can move between blue squares, which are free states, and cannot move into yellow squares, which are walls. Each state has a unique index value which is used to fire a specific state cell.

3.1.4 *Network Model*

CELL TYPES AND CONNECTIVITY The network is modelled in a *rate-coded* fashion: the model does not simulate individual action po-

go/no-go task and the stop-signal task) and has been included here out of completeness.

tentials or 'spikes' as emitted by real neurons in the brain but simulates how the average firing rate of each cell changes over time. Each cell has an *activation*, which is the amount of input that it receives from other cells averaged over a certain period of time, and a *firing rate*, which is calculated from the cell's activation and represents the time-averaged spiking rate of that neuron once other factors (mainly inhibitory) have been taken into account.

The proposed model's architecture is depicted in Figure 3.1 and contains:

- State cells
- Goal cells
- State-action (SA) cells
- Gating cells
- Action cells

Each state cell r_i^S encodes a specific state that the agent can occupy, each goal cell r_i^{GL} encodes a specific state that the goal can occupy, and each action cell r_i^A encodes a specific action that the agent can take.

Each cell r_i^{SA} in the SA layer receives hardwired afferent synapses from a unique combination of one state cell $\left(w_{ij}^{S-SA}\right)$ and one action cell $\left(w_{ij}^{A-SA}\right)$, such that each SA cell responds to a unique combination of these two inputs. These inputs are only active during learning (see later chapters). Each SA cell also receives a synapse from one goal cell $\left(w_{ij}^{GL-SA}\right)$, which indicates if there is a goal at the SA cell's preferred state. The SA cells send and receive hardwired recurrent connectivity $\left(w_{ij}^{SA-SA}\right)$, *and the pattern of this connectivity encodes the cognitive map.*

The gating cells r_i^G receive hardwired afferent synapses from a unique combination of one state cell $\left(w_{ij}^{S-G}\right)$ and one SA cell $\left(w_{ij}^{SA-G}\right)$. These gating cells each send hardwired efferent synapses to one action cell $\left(w_{ij}^{G-A}\right)$. For each gating cell, the afferent state cell and efferent action cell correspond to the state and action components of the afferent SA cell.

Chapters 4, 5 and 6 provide detailed explorations of biologically plausible learning mechanisms that produce the synaptic connectivity proposed here.

Algorithm 3.1 *Planning in the Hardwired Model.* This describes the network's operations during one timestep. Steps in (brackets) only occur if there is activity in the action cell layer signifying that an action has been selected for the agent's current state.

CELL FIRING State Cells & Goal Cells Fire (eqs. 3.3 & 3.4)

ACTIVITY PROPAGATION SA Cells & Goal Cells to SA Cells (3.7)

INHIBITION SA Cells: Rescale SA Activity in All Active Columns (3.9)

ACTIVITY PROPAGATION State Cells & SA Cells to Gating Cells (3.10)

INHIBITION Gating Cells: Highpass Filter (3.11)

ACTIVITY PROPAGATION Gating Cells to Action Cells (3.5)

INHIBITION Action Cells: Winner-Take-All (3.6)

(AGENT) Agent Moves to Successor State in Grid World

(RESET) All Cells Reset to Zero Activation

CELL INTERACTIONS Algorithm 3.1 shows the network's behaviour over the course of a single timestep, and Table 3.1 shows relevant parameter values. To begin, a layer of state cells represents the state of the agent, imitating sensory input (eq. 3.3). These cells are one-hot: each state cell uniquely represents a single state and only one state cell fires at a time, denoting the current state. These cells are stimulated automatically, and are considered to be the output of unmodelled sensory processes.

$$r_i^S = \begin{cases} 1 & i = \text{current state} \\ 0 & i \neq \text{current state} \end{cases} \tag{3.3}$$

where r_i^S is the firing rate of state cell i.

A layer of goal cells, similar to the state cells, encodes the desired state to be navigated to (i.e., the goal):

$$r_i^{GL} = \begin{cases} 1 & i = \text{goal state} \\ 0 & i \neq \text{goal state} \end{cases} \tag{3.4}$$

A layer of action cells represent the activity of motor effectors, and also function as motor feedback. These cells are stimulated by the gating cells (see below) as follows:

$$h_i^A = \sum_j w_{ij}^{G-A} r_j^G \tag{3.5}$$

$$r_i^A = \begin{cases} h_i^A & h_i^A > t^A \\ 0 & h_i^A \leq t^A \end{cases} \tag{3.6}$$

where h_i^A is the activation of an action cell i, w_{ij}^{G-A} is the weight of a synapse from a gating cell j to that action cell, r_j^G is the firing rate of gating cell j and r_i^A is the final firing rate of the motor cell i after thresholding inhibition using the threshold t^A. Winner-take-all competition is used to ensure that only one action cell fires at a time.

A layer of state-action (SA) cells has been hardwired such that each state-action cell receives a synapse from one state cell and one action cell [4]. Together, the cells of this state-action layer encode all possible state-action combinations available in the environment. At this point in the timestep SA cells are stimulated by a combination of goal input and recurrent SA activity as follows:

$$h_i^{SA} = \sum_j w_{ij}^{GL-SA} r_j^{GL} + \sum_j w_{ij}^{SA-SA} r_j^{SA} \tag{3.7}$$

where $\sum_j w_{ij}^{GL-SA} r_j^{GL}$ is the summed input received from the goal cells and $\sum_j w_{ij}^{SA-SA} r_j^{SA}$ is the summed input from recurrent SA input. SA cells are mutually inhibited so that the firing rates within each currently active column are scaled to sum to 1, and each inactive column

4 This connectivity is not directly used for planning (although Ch. 5 shows that it is necessary for learning to encode the cognitive map) although from a planning perspective the connectivity of each SA cell marks the state-action combination which that SA cell encodes.

remains inactive (with a sum of 0). We produce this by setting the firing rate of each SA cell is equal to that cell's activation as follows:

$$r_i^{SA} = h_i^{SA} \quad \forall i \tag{3.8}$$

and then rescaling the firing rates in each columns according to:

$$\sum_i r_i^{SA} = \begin{cases} 1 & \sum_i r_i^{SA} > 0 \\ 0 & \sum_i r_i^{SA} = 0 \end{cases} \tag{3.9}$$

in order to apply mutual inhibition within each active column.

A layer of gating cells has been hardwired such that each gating cell receives afferent synapses from one state cell and one state-action cell[5], and produces efferent synapses to one action cell. It becomes active as follows:

$$h_i^G = \sum_j w_{ij}^{S-G} r_j^S + \sum_j w_{ij}^{SA-G} r_j^{SA} \tag{3.10}$$

$$r_i^G = \begin{cases} h_i^G & h_i^G > t^G \\ 0 & h_i^G \leq t^G \end{cases} \tag{3.11}$$

where t^G is another thresholding constant.

Parameter	Value
Action Cell Threshold t^A	0.2
Gating Cell Threshold t^G	0.55

Table 3.1: Table of Parameters for Section 3.1

5 The state cell and state-action cell pair that a gating cell receives input from are 'matched' such that the state-action cell represents an action taking in the same state that is encoded by the state cell. Chapter 6 describes this in more detail and describes a mechanism for self-organising such cells in a biologically plausible fashion.

3.1.5 *Simulation Procedure*

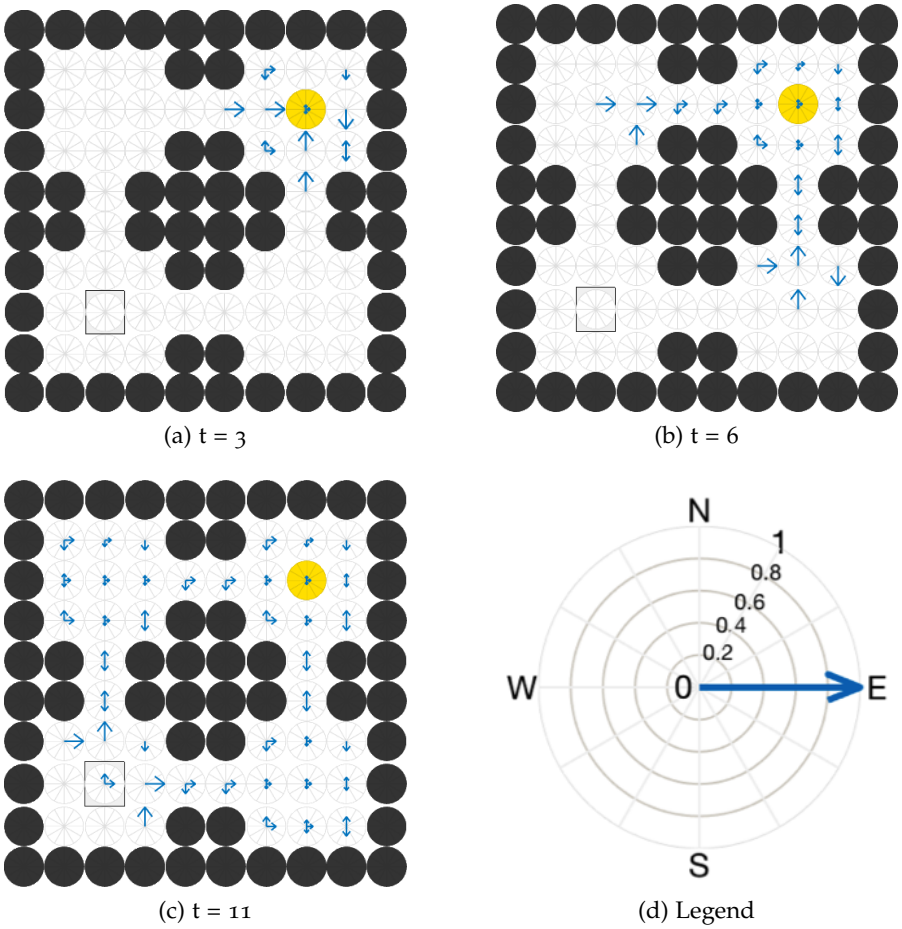
A single timestep of the model is depicted in algorithm 3.1. The state cells fire, encoding the current location of the agent using a vector of cell firing rates where only one cell is active. Each state cell represents a single location in the grid world, an encoding commonly known as "one-hot encoding" (eqs. 3.3).

At the same time, goal cells fire, encoding the location of the goal. Similarly to state cells, each goal cell represents a single state and only one goal cell is active at a time (one-hot encoding).

Activity propagates from the goal cells into the SA cells, along with recurrent activity from the SA layer itself (eq. 3.7).[6] The SA layer is composed of numerous mini-columns, each of which consists of every SA cell which relates to the same state. The activity within these columns is managed through mutual inhibition so that, provided some cells in a column are active, the summed activity within that column is kept constant. This prevents activation from decaying as the goal signal spreads laterally through the SA layer through the recurrent synapses, allowing the model to navigate in maps of indefinite size by effectively boosting activity locally.

Activity then propagates from the SA cells to the gating cells. The gating cells are under heavy inhibition such that an individual gating cell can only produce firing if it is receiving both SA and state input simultaneously (see eqn. 3.11).

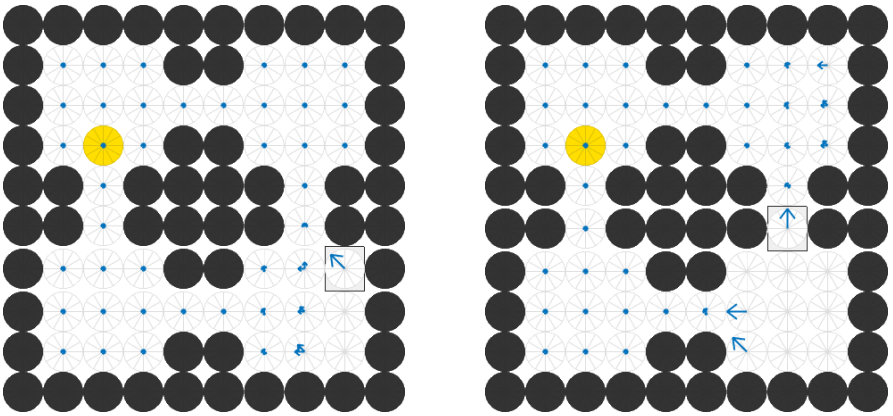
Activity propagates from the gating cells to the action cells (eqs. 3.5 and 3.6). If any action cell begins to fire (eqs. 3.5 and 3.6), the agent takes an action corresponding to the action encoded by the cell (i.e., it moves one space in the specified direction) and updates the world. All cells are then reset.



(a) t = 3 (b) t = 6

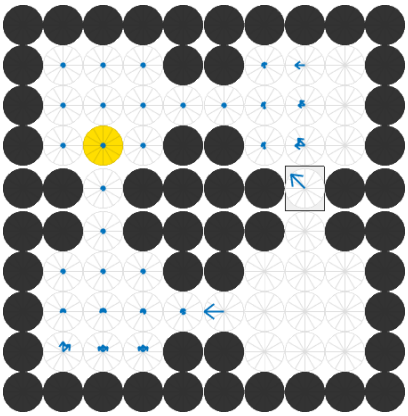
(c) t = 11 (d) Legend

Figure 3.7: *The firing rates of all state-action (SA) cells across an 8x8 grid world during the performance of a planning task.* As described in Sec. 3.1.2, each SA cell encodes a combination of one state and one action. For every state (position) there is therefore a vector of SA cells associated with that state, each representing a different action (direction of movement) that can be taken in that state. The activities of the state-action cells associated with each state are represented by a compass plot; each arrow within a compass plot represents the recorded firing rate of a state-action cell that represents moving in the direction of the arrow from that state. The golden state represents the goal, and the boxed state represents the agent's current location. The agent moves once activity from the goal reaches its current state.

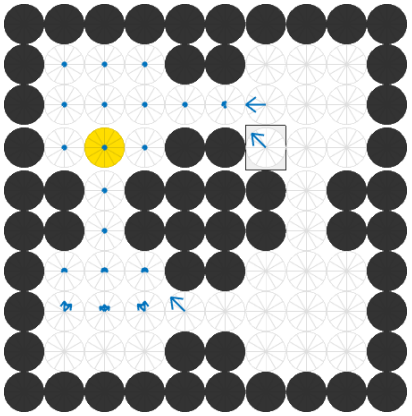


(a) State 1

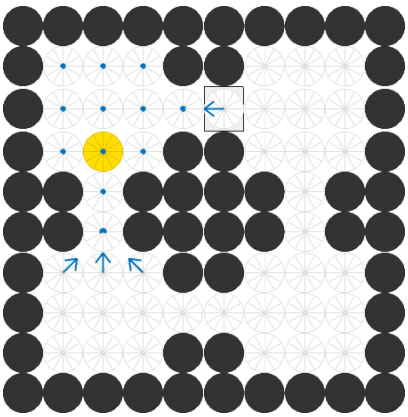
(b) State 2



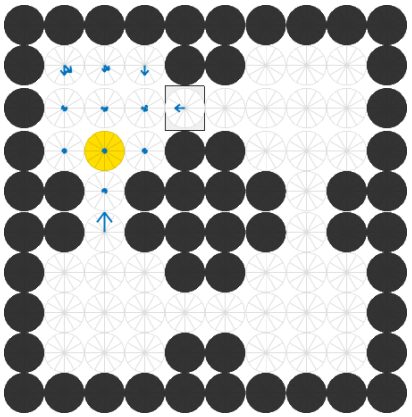
(c) State 3



(d) State 4



(e) State 5



(f) State 6

Figure 3.8: *Choosing Actions to Move Towards a Goal.* This figure shows the firing rates of cells in the SA layer immediately before taking an action, in a set of consecutive states. In (a), for example, the agent is located in the state marked with a box, and is attempting to navigate to the goal state marked in gold. Activity has propagated out from the goal state over a series of timesteps (see Fig. 3.7 for an example of activity propagation in a similar task) and reached the agent's current state, indicating an optimal action (NW). The agent therefore moves to a new state (b), the activity in the layer is reset and the planning process repeats. (See Fig. 3.7 for an explanation of how the compass plots represent SA cell activity).

3.1.6 *Results*

Figure 3.7 displays recordings taken of state-action cells during the navigation task, demonstrating how activation spreads through the map from the goal location (marked in gold) to the agent's current state (marked with a square). As described in Section 3.1.5, activity is fed into the goal state, marked in gold, and each subplot shows the evolution and propagation of this activity at a given timestep. Fig. 3.7a shows the activity in the SA layer soon after the initial stimulation of the goal state; we see that activity has spread a short way and that the SA cells which fire at the edges of the active area encode appropriate actions, although those within the area do not. This gives rise to the gating problem, discussed in Section 3.1.2. Fig. 3.7b then shows that after three more timesteps, activity has spread further through the SA layer and once again the cells on the edge of the active area encode correct actions (with one exception, possibly caused by an incomplete map). Finally, in Fig. 3.7c, activity has propagated to the agent's current state, marked by a box. Again, the SA cells firing on the edge of the active area (i.e., those activated in the current timestep) encode actions that move the agent towards the goal. One of these actions is selected, and the agent moves towards the goal.

Where Figure 3.7 shows how activity propagates through the recurrent synapses between SA cells in order to select an action that is optimal in the agent's current state, Figure 3.8 shows how actions are chosen for consecutive states during a task. Each subfigure in Fig. 3.8 shows the final firing rate of all SA cells before the agent takes an action. We see that every time the activity propagating from the goal state reaches the agent's current state (marked by a box), the most active SA cells for the marked state encode an optimal action.

We tested the model in 100 trials on each environment type (the open environment and the four-room environment) shown in Fig. 3.6. The model was able to successfully navigate from a randomly gener-

6 The SA layer receives afferent connectivity from the state and action cell layers, but these connections are only active during learning, as seen in later chapters.

ated initial state to a randomly generated goal state in 100% of trials; this was true for both environment types.

3.1.7 *Discussion*

Route planning and navigation through the environment is robust; provided that the map contains a path between the current state and the goal location, the model will find it given sufficient planning time.

In this initial simple model, there is no mechanism for changing the map according to experience or for encoding and using non-deterministic transition probabilities. This will be explored in later sections.

The model is Markovian and has the associated limitations (see Section 2.3). The proposed network also segments the state space into discrete states, which it can do even if sensory information is received in the form of a distributed pattern of sensory cell firing.

3.2 PLANNING IN GRID WORLDS WITH NOISE

3.2.1 *Statement of Problem*

Natural neurons in the brain display noisy firing. Maass 2014 asserts that "a major source of stochasticity in networks of neurons in the brain is the unreliability of synapses". Specifically, synaptic transmission is unreliable because the spike of a presynaptic neuron does not reliably produce vesicle release. A presynaptic spike causes vesicle release with about 0.1 probability and two pyramidal neurons are typically connected via five to ten release sites. The probability that no neurotransmitter at all will be released is 0.59 for a five-vesicle synapse (0.9^5) and 0.35 in a ten-vesicle synapse (0.9^{10}); the probability of inadequate neurotransmitter release (i.e., inadequate vesicle release) is much higher than these figures and so spike transmission is inherently unreliable [96]. Furthermore, the various biochemical mechanisms responsible for neuron operation are inherently stochastic.

Maass 2014 also suggests that the reliability of vesicle release appears to be related to plasticity.

The response of a postsynaptic neuron to presynaptic stimulation is variable. This is partly because of variable vesicle release (discussed above) as well as spontaneous vesicle release without an action potential (the spontaneous miniature postsynaptic current). But the postsynaptic current induced by a vesicle is also variable due to noisy biochemical mechanisms: vesicles have variable size and contain variable amounts of neurotransmitter, these transmitter molecules diffuse randomly, and the location of vesicle release will impact vesicle response.

Planning in the wavefront model is heavily dependent on the timing of SA cell activations as activity propagates through the layer, especially due to the gating mechanism used. Consequently the planning mechanism might be prematurely affected by noisy activation in certain SA cells. Furthermore, the inhibitory system acting on the SA cells is designed to keep SA neurons (if they are active at all) at a certain level of activity within each mini-column; there is the risk that this normalisation process will perceive noisy activity as meaningful activity and therefore amplify noisy firing rates so that the noise interferes with the meaningful propagation of activity through the SA layer.

3.2.2 *Hypothesis*

As described in the previous section, the gating mechanism is designed to only pass on activity to the action cells if it comes from the right SA cells at the right time. To that end, the gating cells are heavily inhibited such that they can only fire if they receive simultaneous input from both an SA cell and the corresponding state cell representing the current state of the agent, signifying that the SA cell is representing an optimal action (based on the network's internal map) for the agent's current state (eq. 3.10, 3.11). We hypothesise that if this inhibition is strengthened further (by raising the value of the

inhibitory constant t^G in the gating inhibition eq. 3.11), the gating cell will be prevented from passing noisy SA activity to the action cells.

The previous section also states that neurons are likely to receive spontaneous miniature postsynaptic currents—excitatory activation that is unrelated to the activity of presynaptic cells [96]. This means that all cells in the SA layer may be partially active at any time. However, in the normal formulation of the model described in Section 3.1, the mechanism which propagates activity through the SA layer normalises the activity in each active column to ensure that the activity does not decay as the wave propagates. If all cells are constantly receiving low-level stimulation from spontaneous postsynaptic currents, every column becomes technically 'active' and the normalisation mechanism may struggle to distinguish this noise-related activation from genuine goal-centred activation. It is known that the brain contains homeostatic mechanisms to suppress this low-level activation with an equal amount of low-level inhibition [97] and we hypothesise that if we imitate this excitation-inhibition balance by suppressing SA cells with low firing rates then—as long as goal-related activation is stronger than the level of spontaneous noise—only 'true' goal-related activation will remain in the SA layer and so the normalisation mechanism will be able to function properly.

3.2.3 *Task*

The task is essentially similar to that in Sec. 3.1. The agent is placed in a random state in an 8x8 open world, and a random state is designated as the agent's goal. The agent can move one space at a time in eight compass directions or stay still, giving nine possible actions. The task is for the simulated agent to navigate to the goal location. If the agent reaches the goal location then the task has been completed successfully. If the agent fails to reach the goal location within 100 timesteps then the agent has failed the task. The primary difference between this task and the task in the previous section (sec. 3.1) is that all activity propagates noisily (see eqs. 3.12 and 3.13 as well as parameter table 3.2).

3.2.4 *Simulation Procedure*

As described in Section 3.2.1, the postsynaptic response of a cell to stimulus is noisy due to transmission variability and spontaneous vesicle release. We have simulated this by calculating noisy layer activations such that:

$$r_i^{RND} \sim U([-c, c]) \tag{3.12}$$

$$h_i = \sum_j w_{ij} r_j + r_i^{RND} \tag{3.13}$$

where h_i is the activation of a cell i, c is a constant that determines the level of noise and r_i^{RND} is a randomly generated noise value calculated according to a uniform distribution over the interval $[-c, c]$ that shifts the 'true' activation in a positive or negative direction. If a cell i has negative activation after the addition of noise, its activity is thresholded to zero. Note that any cell in a layer may become active regardless of whether it receives any 'true' activation from presynaptic cells, mimicking the effect of spontaneous vesicle release.

Algorithm 3.2 describes one timestep. The sensory cells fire (eq. 3.3), and so do the goal cells (eq. 3.4). The goal cells feed activity into the SA layer (eq. 3.7). The amount of activity in the SA layer is thresholded to compute the SA cell firing rate as follows:

$$r_i^{SA} = \begin{cases} h_i^{SA} & h_i^{SA} \leq t_{min}^{SA} \\ 0 & t_{min}^{SA} < h_i^{SA} < t_{max}^{SA} \\ h_i^{SA} & h_i^{SA} \geq t_{max}^{SA} \end{cases} \tag{3.14}$$

where h_i^{SA} is the activation of a cell i in the SA cell layer, r_i^{SA} is the firing rate of the same cell, and t_{min}^{SA} & t_{max}^{SA} are constants that define the limits of the suppressed bandwidth. This is distinguished from simple activity thresholding, which is defined as follows:

Algorithm 3.2 *Planning in the Hardwired Model with Noise.* This describes the network's operations during one timestep, based on alg. 3.1. The steps that have been added in this version are emphasised with italics. Steps in (brackets) only occur if there is activity in the action cell layer signifying that an action has been selected for the agent's current state.

CELL FIRING State Cells & Goal Cells Fire (eqs. 3.3 & 3.4)

ACTIVITY PROPAGATION SA Cells & Goal Cells to SA Cells (3.7)

inhibition SA Cells: Bandstop Filter (3.14)

INHIBITION SA Cells: Rescale SA Activity in All Active Columns (3.9)

ACTIVITY PROPAGATION State Cells & SA Cells to Gating Cells (3.10)

INHIBITION Gating Cells: Highpass Filter (3.11)

ACTIVITY PROPAGATION Gating Cells to Action Cells (3.5)

INHIBITION Action Cells: Winner-Take-All (3.6)

(AGENT) Agent Moves to Successor State in Grid World

(RESET) All Cells Reset to Zero Activation

$$r_i^{SA} = \begin{cases} t_{max}^{SA} & h_i^{SA} > t_{max}^{SA} \\ h_i^{SA} & h_i^{SA} \leq t_{max}^{SA} \end{cases} \tag{3.15}$$

The use of a bandstop filter (eq. 3.14) rather than a simple threshold (eq. 3.15), inhibits activity when $h_i^{SA} \leq t_{min}^{SA}$ and is necessary to suppress the low-level noise that would otherwise be present in the SA cell layer as a result of noisy stimulation by state and action cells. This is necessary because the activity of the SA minicolumns is rescaled using divisive inhibition (eq. 3.9) in order to keep the level of activity constant as it propagates through the SA map. In theory, the gating cells are heavily inhibited such that they can only fire if they receive simultaneous input from both an SA cell and the corresponding state cell representing the current state of the agent, signifying that the SA cell is representing an optimal action (based on the network's internal map) for the agent's current state (eqs. 3.10 and 3.11). However, the rescaling of noisy activity would promote the low-level noise present in otherwise inactive state columns, potentially allowing this activity to overwhelm the gating cell and be passed to the action cells (eqs. 3.5 and 3.6). The bandstop filtering shown in eq. 3.14 is required to prevent this scenario.

After this point, the timestep continues as in Sec. 3.1.5. Activity propagates from the SA cells to the gating cells. The gating cells are under heavy inhibition such that an individual gating cell can only produce firing if it is receiving both SA and state input simultaneously (see eqn. 3.11). Activity then propagates from the gating cells to the action cells (eqs. 3.5 and 3.6). If any action cell begins to fire (eqs. 3.5 and 3.6), the agent takes an action corresponding to the action encoded by the cell (i.e., it moves one space in the specified direction) and updates the world. All cells are then reset.

Table 3.2 shows the relevant parameters for this section.

Parameter	Value
SA Cell Upper Threshold t^{SA}_{max}	1
SA Cell Lower Threshold t^{SA}_{min}	0.2
Noise Factor c	0.1

Table 3.2: Table of Parameters for Simulations With Noise Described in Section 3.2

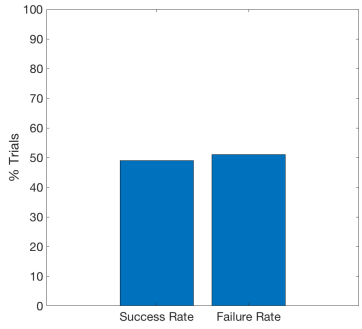
3.2.5 *Results*

Figure 3.9 shows that the model is not able to reliably plan in noisy conditions if the firing rate of SA cells are simply thresholded to remain below a certain value (eq. 3.15), but that the model can do so if both high and low firing rates are filtered using a bandstop filter (eq. 3.14). Although the model is able to achieve an approximately 50% success rate using simple thresholding, this appears to be due mostly to random movement as it requires a much larger number of timesteps compared to the bandstop version (Fig. 3.9c).

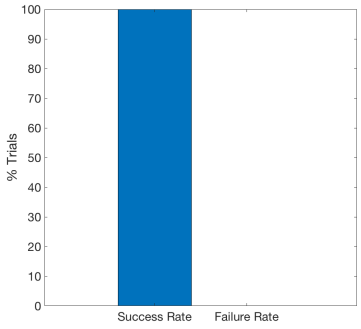
3.3 PLANNING WITH MULTIPLE SENSORY INPUTS

3.3.1 *Description of Problem*

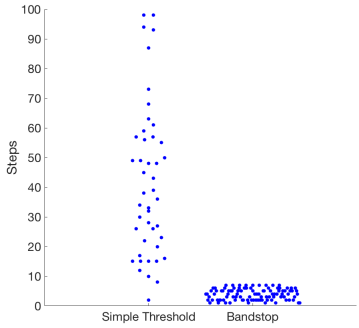
In the previous experiments, the state input to the model was *one-hot*, which is to say that a single state cell fired to indicate the precise state that the agent was occupying. In other words, only a single piece of information was required to determine the agent's current state. However, a state is usually dependent on many separate and (partially) independent factors. This means that the model must be able to learn unique combinations of factors that indicate a unique state. We model this problem using the same grid world task as before, but rather than indicate the state (and goal) as a single piece of information (0-100 for a 10x10 grid world), we indicate it using x and y coordinates. Each SA cell now encodes a triple combination of x-coordinate, y-coordinate and action, receiving hardwired synapses



(a) Model using simple threshold inhibition only, as described by eq. 3.15



(b) Model using bandstop inhibition, as described by eq. 3.14



(c) Swarm plot of the time taken to reach the goal in successful trials, when using simple threshold or bandstop conditions. The y axis describes the average number of steps taken to reach the goal for each of the two model conditions. To make individual points more visible they are jittered in the x-axis.

Figure 3.9: Performance of the model when using simple threshold inhibition (eq. 3.15) or bandstop inhibition (eq. 3.14). (a) shows that if the model is using simple threshold inhibition, it is able to achieve a roughly 50% success rate in the 8x8 open room task, while (b) shows that the use of bandstop inhibition gives 100% success. By looking at the average times for the agent to reach the goal (c), we can see that there is a very wide distribution of times in the simple threshold case. Combined with the low success rate in (b), the distribution of times in the simple threshold case suggests that the agent is moving extremely randomly and that the successful trials are mostly the result of chance. By contrast, the time-to-goal values in the bandstop case are much lower and more closely distributed, suggesting that the agent takes a mostly direct path towards the goal.

from two state cells (encoding the x-coordinate and y-coordinate of its favoured state respectively), one SA-goal cell (see below) and sends a hardwired synapse to one action cell. Goal cells are also organised by x- and y-coordinates. All cell types and connectivity that have not been mentioned are the same as in Sec. 3.1.

However, this produces a problem. There are now many states that are partially associated with the goal: all of the states that share the same x-coordinate and all of the states that share the same y-coordinate (Fig. 3.10). This is useful in certain cases (where a goal may be defined as a particular subset of factors) but prevents the model as it stands from being able to find a goal that is uniquely defined by the combination of several factors.

To take Fig. 3.10 as an example, only the goal state should be active at the first timestep. Instead we see that every state that shares an x-coordinate or a y-coordinate with the goal has received activity marking them as a potential goal state. This means that, rather than activity propagating outwards from the goal state as intended (Fig. 3.7) the activity will spread out from all of the cells active in Fig. 3.10 at once, and therefore the agent will move towards the closest one of these cells as if it is the goal state. This is illustrated in figure 3.11, where we see that the agent chooses to move towards such a pseudo-goal state, and, having reached it, chooses to remain there rather than move towards the true goal.

3.3.2 *Hypothesis*

We hypothesise that an extra layer of processing must filter the SA response to the activation they receive from the goal cells such that it filters out SA cells which only have a partial match to the goal and leaves those that have a full match. We represent this extra processing using a layer of SA-goal cells, described below.

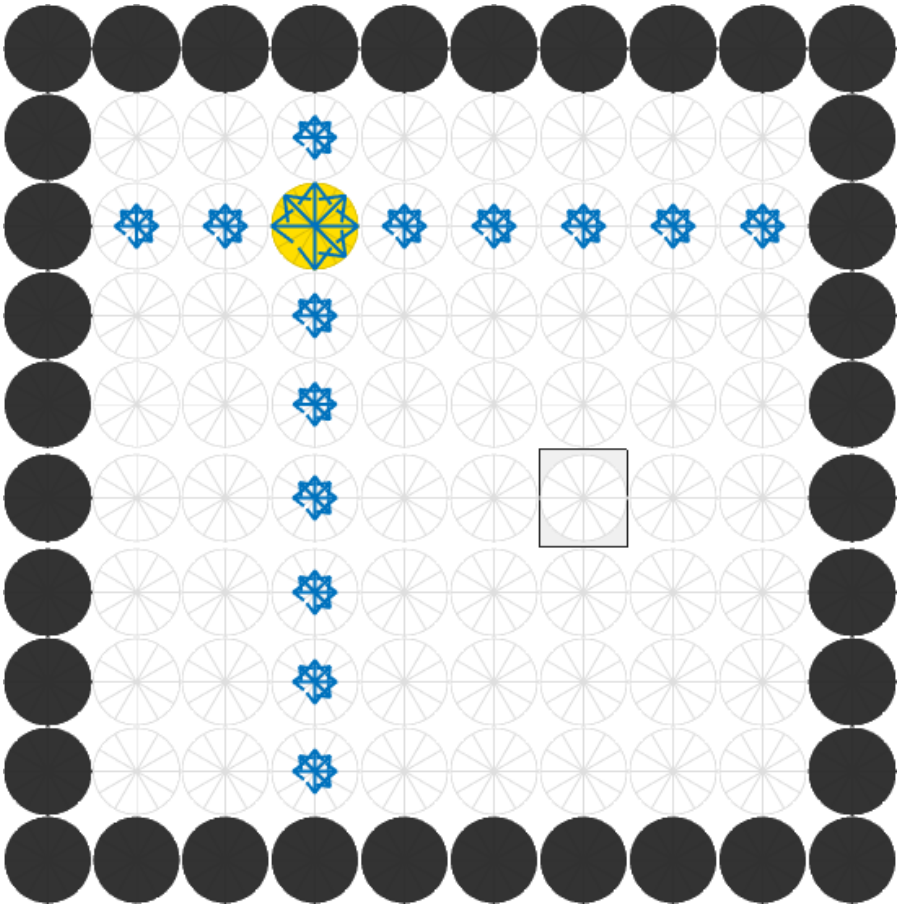
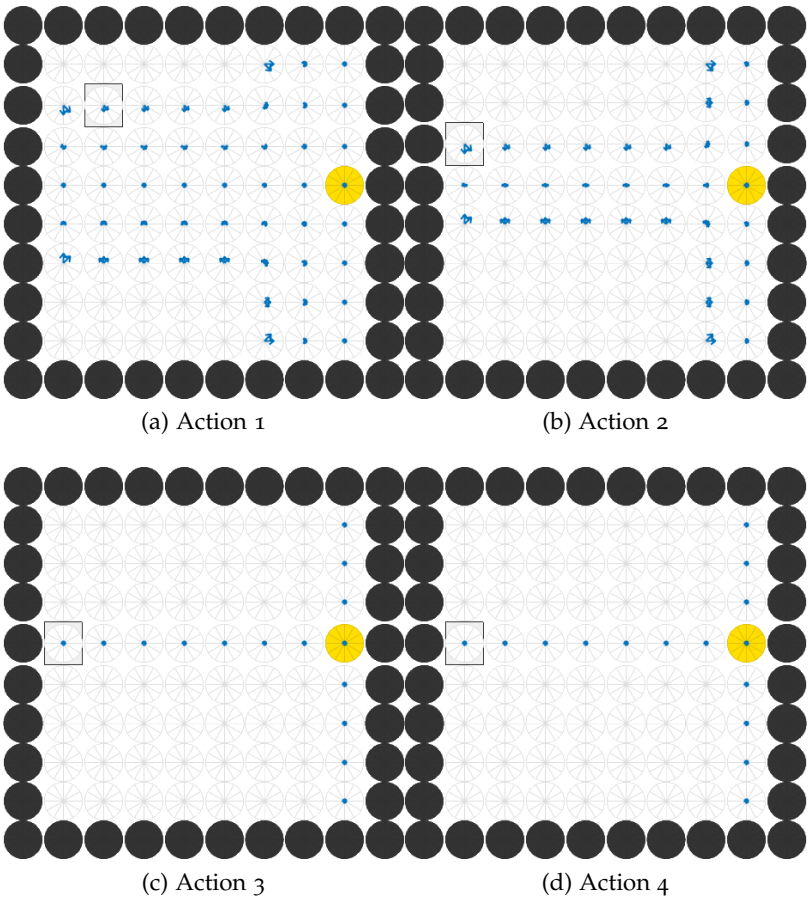


Figure 3.10: *Activation in the SA layer when stimulated by a goal cell layer encoding the goal in xy coordinates.* SA cells representing the actual goal state (in gold) receive stimulation from both the x-goal cells and the y-goal cells and so have high activation. However, the SA cells representing a state which shares only one of those coordinates also receive partial stimulation.



(a) Action 1 (b) Action 2

(c) Action 3 (d) Action 4

Figure 3.11: The SA layer activity during a sequence of actions when stimulated by a goal cell layer encoding the goal in xy coordinates. As in Fig. 3.10 all cells sharing the x- or y-coordinate of the goal become activated by goal-based stimulation, so that the network perceives them as potential goals. This has the effect of causing the agent to move towards the nearest state that shares an x- or y-coordinate with the goal and then remaining there (note that diagonal movements are the same 'length' as horizontal or vertical movements). In (a) and (b), the agent takes actions that move it towards states which share an y-coordinate with the goal. Having reached such a pseudo-goal state in (c), the agent chooses to remain there rather than move towards the true goal, and is still in the same position in (d).

3.3.3 *Task*

The task is based on a 2D grid world formulation. The environment is a 10 by 10 grid, bounded by walls to make it 8 by 8. In each trial a random location within this grid is designated as the agent's goal and another random location is designated as the agent's starting point. The agent can move one space at a time in eight compass directions[7] or stay still, giving nine possible actions.

The task is for the simulated agent to navigate to the goal location. If the agent reached the goal location then the task has been completed successfully. If the agent fails to reach the goal location within 1000 timesteps then the agent has failed the task.

3.3.4 *Methods*

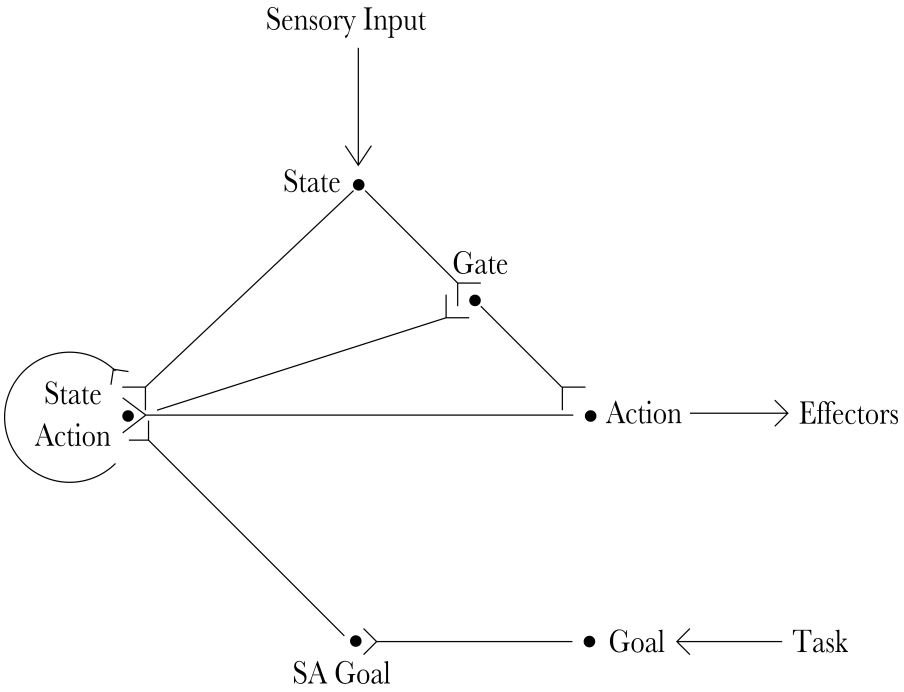


Figure 3.12: Expanded network architecture including SA Goal cells

The state cells in this experiment have been modified to represent the agent's state as a pair of x-y coordinates. The state layer consists of two sets of 10 cells, each encoding one of the coordinates from 1 to

7 N, NE, E, SE, S, SW, W, NW

Algorithm 3.3 *Planning in the Hardwired Model with XY Inputs.* This describes the network's operations during one step in the planning process, based on alg. 3.1. The steps that have been added in this version are emphasised with italics. Steps in (brackets) only occur if there is activity in the action cell layer signifying that an action has been selected for the agent's current state.

CELL FIRING State Cells & Goal Cells Fire (eqs. 3.3 & 3.4)

propagation SA Cells & Goal Cells to SA-Goal Cells (3.16)

inhibition SA-Goal Cells: Highpass Filter (3.17)

propagation SA-Goal Cells to SA Cells (3.18)

INHIBITION SA Cells: Threshold (3.15)

INHIBITION SA Cells: Rescale SA Activity in All Active States

PROPAGATION State Cells & SA Cells to Gating Cells

INHIBITION Gating Cells: Threshold

PROPAGATION Gating Cells to Action Cells

INHIBITION Action Cells: Winner-Take-All

(AGENT) Agent Moves to Successor State

(RESET) All Cells Reset to Zero Activation

10 using one-hot encoding. The network's layer of goal cells represent the goal state in the same way.

The network functions essentially the same way as described in previous sections of this chapter. Algorithm 3.3 describes one timestep. Initially, the layer of state cells fire, and so does the layer goal cells. Unlike in previous sections, activation does not pass from the goal cells to the SA cells but instead passes from the goal cells to a layer of *SA-goal cells* as follows:

$$h_i^{SG} = \sum_j w_{ij}^{GL-SG} r_j^{GL} \tag{3.16}$$

where h_i^{SG} is the activation of an SA-goal cell i, r_j^{GL} the firing rate of a goal cell j, and w_{ij}^{GL-SG} is the weight of the synapse from the goal cell to the SA-goal cell.[8]

The SA-goal cells are required to 'filter' activity propagating from the goal representation to the SA layer to ensure that only SA cells encoding the full goal state receive goal-related stimulation. Essentially, the goal-SA cells act as surrogates for the SA layer, which can be safely inhibited without affecting the other processes that occur in the SA layer. Because of this, and because the goal cells use the same x- and y-coordinate representations to encode the goal as that used by the state cells to encode the state, the goal to SA-goal connectivity w_{ij}^{SG-SA} is therefore identical to the state-to-SA connectivity w_{ij}^{S-SA}.

In this case, cells which respond to only the x-coordinate or y-coordinate of the goal will receive only partial activation (see Fig. 3.10) and so will not reach the SA-goal threshold t^{SG} (see below). Only the SA-goal cells that receive activity from both x- and y-coordinate representations will receive enough activation to push their activations above t^{SG} and so to fire. Figure 3.12 shows the connectivity of these cells within the broader network architecture. Each SA-goal cell receives hardwired synapses from two goal cells (encoding the

8 Note that the synaptic weights w_{ij}^{GL-SG} are identical to w_{ij}^{S-SA}. This is so that a goals can be fed into the model in the form of sensory input, so that a goal becomes the sensory data designated a particular desired state or set of states. It may be possible to learn w_{ij}^{GL-SG} but we have not yet found a mechanism for doing so.

x-coordinate and y-coordinate of its favoured goal respectively) and sends a hardwired synapse to one SA cell that encodes the same state.

As described above, SA-goal cells undergo thresholding after activation, thereby managing the flow of activity between the goal cells and the SA cells, preventing the goal cells from activating SA cells which are only partially associated with the goal state. This filtering works as follows:

$$r_i^{SG} = \begin{cases} h_i^{SG} & h_i^{SG} > t^{SG} \\ 0 & h_i^{SG} \leq t^{SG} \end{cases} \tag{3.17}$$

where r_i^{SG} is the firing rate of SA-goal cell i after thresholding.

Activity propagates from the SA-goal layer to the SA layer as follows:

$$h_i^{SA} = \sum_j w_{ij}^{SG-SA} h_j^{SG} + \sum_j w_{ij}^{SA-SA} r_j^{SA} \tag{3.18}$$

where $\sum_j w_{ij}^{SG-SA} h_j^{SG}$ is the summed input received from the SA goal cells and $\sum_j w_{ij}^{SA-SA} h_j^{SA}$ is the summed recurrent input from the SA cells. The synaptic weights w_{ij}^{SG-SA} are set according to:

$$w_{ij}^{SG-SA} = I \tag{3.19}$$

where I is the identity matrix.

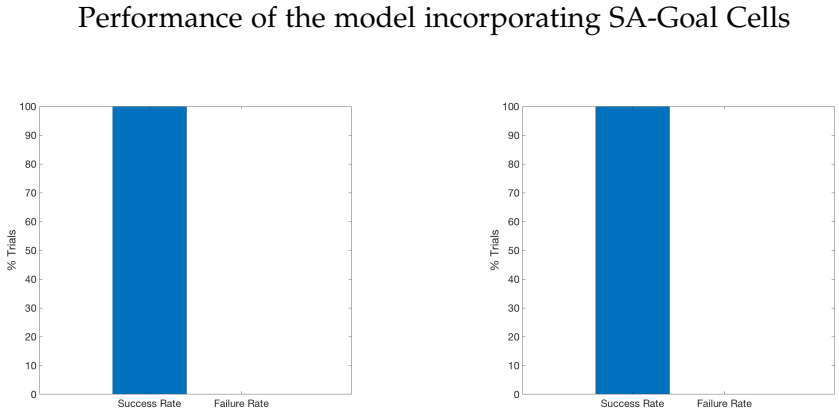
The rest of the timestep proceeds as the first model described in this chapter, with the only exception being that the inhibition threshold for the gating cell (eq. 3.11) has to be slightly raised to account for the fact that a gating cell may receive partial state activation if the agent's current state shares one coordinate with the gating cell's preferred state.

Parameter	Value
SA-Goal Cell Threshold t^{SG}	0.51
Gating Cell Threshold t^{G}	0.8

Table 3.3: Table of Parameters for simulations with multiple sensory inputs described in Section 3.3

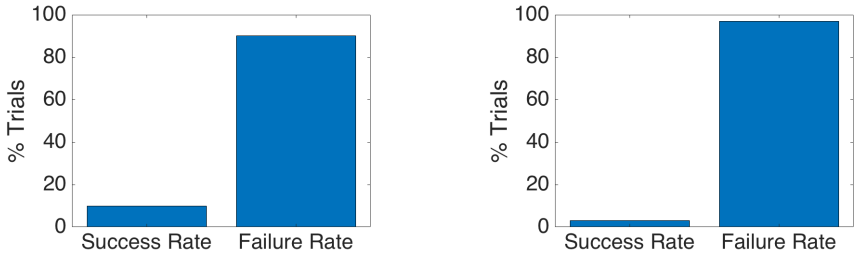
3.3.5 *Results*

Fig. 3.13 shows that the model can achieve robust performance in both an open environment and a 4-room environment when the state is described in xy coordinates. By contrasting this with Fig. 3.14, we see that the use of an SA-goal cell layer is crucial to this behaviour, and that without it the model is unable to plan successfully.



(a) Success Rate in 8x8 Open Room (b) Success Rate in 8x8 4-Room Maze

Figure 3.13: Percent success rate of the model over 100 trials with SA-Goal cells if state and goal are represented by xy coordinates. (a) shows that the model is able to achieve 100% success rates in an open environment; (b) shows that the model is able to achieve the same performance in a 4-room maze.



(a) Success Rate in 8x8 Open Room (b) Success Rate in 8x8 4-Room Maze

Figure 3.14: Percent success rate of the model without SA-Goal cells over 100 trials if state and goal are represented by xy coordinates. In this case activity is propagated straight from the goal cells to the SA cells. (a) shows that such a model is unable to achieve high success rates in an open environment; (b) shows that the model is unable to achieve high performance in a 4-room maze.

3.3.6 *Discussion*

The above results make it clear that the extended model with SA-goal cells can operate when states are represented by multiple independent factors. This is necessary in many ways. Firstly, even in simple navigation tasks of the kind represented by, for example, Morris' water maze [98], where a rat is trained to locate a fixed platform that it cannot see, hear or smell using only the remembered location of the hidden platform (goal) and the agent's knowledge of its own position (usually attributed to a place cell representation), the place cell representation of the agent's position is not encoded in a one-hot fashion but is encoded by the relative firing of several overlapping place cells with wide fields. Secondly, if the focus of planning is expanded beyond simple navigational tasks in rodents it becomes clear that being able to track several pieces of information is crucial. For example, in the coffee-making task described by Cooper and Shallice [99]—where a model is given a tray full of implements and then tasked with making coffee by putting the beans in the grinder, the water in the cup and so on—the state is represented by a combination of the positions of different objects as well as the individual states of those objects.

3.4 DISCUSSION

This is the first experimental chapter of the thesis, in which we have proposed a model with pre-wired synaptic connectivity that is able to perform navigation tasks using an internal cognitive map of the environment. This map is not domain-specific, but uses the Markovian formulation of states, actions, and transition probabilities, and the agent is able to use it to produce robust planning performance, even in the presence of noise. The model initially uses only one piece of sensory data to inform it of its current state, but we demonstrate that the model can still produce robust performance if state representations are more complex.

The models presented in this chapter, and indeed in the thesis as a whole, are intended to embody the simplest working architecture

that is able to perform model-based Markovian planning, and are not intended to simulate specific cell types in the brain. For this reason, the model has been deliberately simplified, so that it is rate-coded and uses grandmother-cell encodings of state and action, as well as winner-take-all competition, as opposed to the distributed representations that are likely to occur in the brain. Model-based planning is a complex process and so these compromises are typical of work in this area (see Ch. 2).

Although the model's design is primarily based on function, many aspects of it accord with current experimental findings. Cells with analogous properties to SA cells [88, 91–94], action cells [58] and gating cells [100] exist in the prefrontal cortex, premotor cortex and basal ganglia respectively. The reverse spread of activity is consistent with experimental observations made by Kurth-Nelson et al. [9], who found after training participants to perform a simple grid world task they produced spontaneous whole-brain magneto-encephalographic (MEG) activity[9] encoding encoding reverse paths through the environment. To quote: "If a neural representation of state k was active at time t, then a representation of one or both of the states with transitions to k was active around t + 40ms." These four-state sequences were legal moves according to the transition structure of the environment. There is also evidence for forward planning mechanisms in the brain [26], but we have not been able to find robust map-based network models which plan using a forward mechanism (see ch. 2) and other modellers have criticised the idea of planning based on forward sweeps through a synaptic map [27]. Our own research into potential forward planning mechanisms along these lines did not produce a working model and so for the rest of this thesis we will consider reverse planning only.

The following three chapters (4, 5, and 6) will consider how this architecture might self-organise in the brain through sensory-guided learning. These chapters will discuss the self-organisation of SA cells' response properties, the recurrent connectivity between SA cells, and the response properties of gating cells respectively. Chapter 6 will also

9 Decoded using a trained regression model.

show how the efferent connectivity from the gating cells to the action cells is learned. Each chapter will extend the processes described by the last chapter until Ch. 6 describes the full self-organisation of the model described in this chapter, with the exception of the SA-goal cell layer. This cell layer has similar connectivity to the SA and gating layers, and could probably self-organise in a similar way, but we have not attempted to demonstrate this experimentally.

4

SELF-ORGANISING STATE-ACTION CELLS FROM SENSORY AND MOTOR INPUTS USING HEBBIAN LEARNING

Chapter 3 demonstrated that if the proposed model is given a hard-wired map of its environment in the form of recurrent synapses between state-action cells, then it is possible for the network to perform planning in environments with and without obstacles. From a modelling perspective (see Section 2.1), it is now necessary to ask how the proposed model might come about. How do the various cell types described in the previous chapter self-organise their distinctive response properties and synaptic connectivity?

In this chapter, we focus specifically on the formation of cells with analogous properties to state-action cells. All of the alternative models reviewed in Chapter 2 encode cognitive maps using either state (place-like) cells or state-action cells.[1] The distinctive responses of these cells are extremely important for the planning mechanisms to work as described, but the question of how these cells would learn these responses has not been fully explored in other literature. Only Cuperlier 2007 and Martinet 2011 give a partial description of how these cells could learn their distinctive properties and there are drawbacks to the mechanisms described in both cases (see the relevant sections in Ch. 2 for more detail).

In this chapter, we therefore attempt to describe how state-action (SA) cells could self-organise from state and action feedback received by the network. The extended model discussed in this chapter produces novel predictions about the learning of the cognitive map substrate. Moreover, self-organising the state-action cells via biologically plausible learning mechanisms also gives extra support to the pro-

1 Sometimes embedded in, or connected to, larger units that are typically described as neocortical mini-columns [2, 81].

posal that the cognitive map substrate modelled in Ch. 3 could plausibly exist in the brain given known cellular mechanisms.

In modelling the self-organisation process, we have adhered to certain principles. State-action cells should self-organise all of their connectivity from sensory (e.g., visual) and action inputs, rather than rely on pre-existing hardwired connectivity or the alteration of specified synapses using biologically implausible algorithms. This self-organisation should rely on local learning processes, so that synapses are modified based purely on the information that is available to them in the form of pre- and postsynaptic firing rates, according to known neurobiological findings.[2]

Section 4.1 describes a method for forming SA cells, using Hebbian learning and the restriction of synaptic connectivity between layers, to produce SA cells that each receive input from a unique combination of state and action cells.

Section 4.2 describes an alternate method for forming SA cells, using a columnar formulation and lateral inhibition to produce SA cells that organise themselves into "state columns". These columns each contain a group of SA cells that encode different state-action combinations for the same state.[3] This self-organisation is tested using an information-theoretic measure. Please note that the term 'column' refers to cortical minicolumns (3.1.2) unless otherwise specfied. This method is able to produce available SA cells more efficiently than the diluted connectivity method, and inherently produces the columnar structures required for planning (see Ch. 3).

Section 4.3 shows a third method for forming SA cells, using a form of inhibition that we call "bandstop inhibition". This method is a necessary element when forming SA cells using distributed sensory information.

Finally, Section 4.4 shows how the columnar formulation can be used together with bandstop inhibition to self-organise columnar SA cell responses in cases where states are encoded by multiple pieces of

2 The proposed model is rate-coded, and so Hebbian learning was used as a time-averaged approximation of Spike-timing-dependent plasticity (STDP).

3 Winner-take-all mechanisms have a long modelling history in multiple areas [101], and are supported by anatomical and physiological evidence [102, 103]. They have also been modelled using biologically realistic spiking networks [102].

information; in this case, each state is defined as a unique combination of x- and y-coordinates.

4.1 FORMATION OF SA CELLS USING DILUTED SYNAPTIC CONNECTIVITY

4.1.1 *Statement of Problem*

All potential state-action cells receive input from state cells, which encode sensory input, and action cells (see Fig. 3.1 on page 72). In the last chapter's hardwired model, these cells encoded a unique state-action combination. How might they learn to do so based on the inputs that they receive?

4.1.2 *Hypothesis*

During learning, the SA cells must form afferent connections from the state cells and action cells such that each SA cell learns to respond to a unique combination of state and action. Moreover, all possible combinations of state and action must be represented by an even distribution of SA cells. We hypothesised that the required connectivity may be set up by *competitive learning*. In this scenario, a layer of SA cells receive afferent connections from a mixture of state cells and action cells. These connections are modified by associative Hebbian learning as the network agent explores its sensory training environment. During this process, the layer of SA cells is put under heavy mutual inhibition, such that very few cells can fire at the same time. This inhibition means that cells' receptive fields become distributed evenly across the input space, because each cell can only fire if it learns to respond to part of the input space that has not been 'claimed' by other cells.

Past competitive learning papers such as Walters and Stringer 2010 [104] showed that cells could learn combinations of inputs using competitive learning, so long as the relative activity received from each

kind of input (e.g., state and action inputs) is carefully balanced (so that one input does not dominate the other). Restricting synaptic connectivity from the inputs to the SA layer, such that only a small percentage of the potential connections between these layers are in fact permitted, is one way to produce this effect. This synaptic restriction, or *diluted connectivity*, also reduces the number of cells that are able to compete for any given input combination and prevents any given cell from coming to represent a large contiguous area of the input space.

4.1.3 *Task*

An agent is placed in a 10 by 10 grid world, either an open environment or a small maze, shown in Figs. 3.6a & 3.6b respectively. The agent is able to move in any of the eight cardinal directions. At each timestep during learning, it chooses a random action and transitions to a new state. The agent's task is to explore the environment and self-organise SA cells to encode the state-action combinations that it has experienced.

In other words, we expect that the agent, after training, should have developed state-action cells which respond strongly to an experienced state-action combination. If it does not—if the SA cells respond exclusively to sensory inputs or exclusively to action inputs, if the SA cells do not distinctly encode a single state-action combination—the agent will have performed poorly at this task.

4.1.4 *Network Model*

The architecture of the model is shown in Fig. 4.1. A layer of state cells can fire to indicate the current state of the agent within its environment. A layer of action cells can provide motor feedback about actions the agent has just taken. Both state and action representations use one-hot encoding in this experiment. Distributed sensory representations are explored in Section 4.4. A final layer—designated the

"SA layer" contains potential SA cells that receive afferent connections from the sensory and action layer.

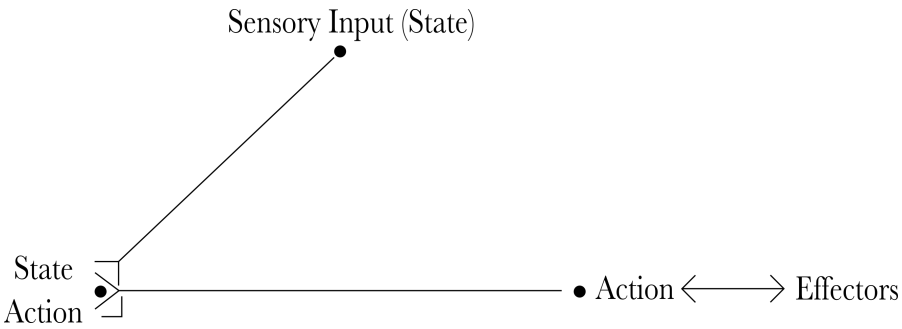


Figure 4.1: Architecture for Self-Organising SA Cells

The afferent connectivity that each SA cell receives from the state and action cells is diluted, so that the probability of any given SA cell receiving an afferent synaptic connection from any given state cell is p^S, and the probability of any given SA cell receiving an afferent connection from any given action cell is p^A. If an afferent connection does exist, its initial weight is set randomly. Figure 4.2 shows the connectivity received by each SA cell.

Algorithm 4.1 *Self-Organising SA Cells Using Competitive Learning with Diluted Connectivity.* This describes one timestep. All plasticity is Hebbian unless otherwise specified. The full sequence of events listed here occurs in every timestep.

CELL FIRING State Cells (eq. 4.1)

CELL FIRING Action Cells (4.2)

AGENT Agent Moves to Successor State in Grid World

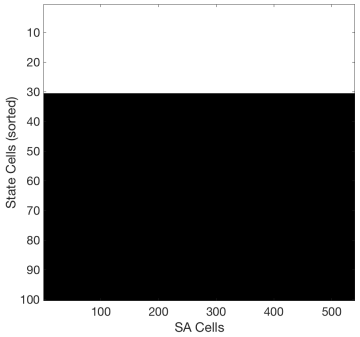
ACTIVITY PROPAGATION State Cells & Action Cells to SA Cells (4.3)

INHIBITION SA Cells: Full Winner-Takes-All (4.4)

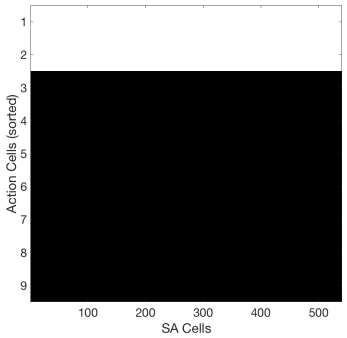
SYNAPTIC PLASTICITY State Cells & Action Cells to SA Cells (4.5, 4.7, 4.6 & 4.8)

RESET All Cells Reset to Zero Activation

Algorithm 4.1 describes one timestep of the simulation. In each timestep, a state cell fires first, representing the current state. State cell firing is one-hot, so each state cell represents a single unique state and only one state cell is active at a time.



(a) Dilution of State to SA Connectivity (sorted)



(b) Dilution of Action to SA Connectivity (sorted)

Figure 4.2: This figure shows the proportion of connectivity to each SA cell from the state (a) and action (b) cells. The x-axis shows each cell in the SA layer, the y-axis shows each input cell that can connect to these SA cells. For clarity, these cells have been sorted by their ability to connect to each SA cell so that the proportion of input cells that do and don't connect to each SA cell is clear. The white area denotes input cells that connect to the SA cell, with a synaptic weight between 0 and 1. The black area denotes input cells that do not connect to the SA cell, with a synaptic weight fixed at 0.

The state cells fire as follows:

$$r_i^S = \begin{cases} 1 & i = \text{current state} \\ 0 & i \neq \text{current state} \end{cases} \tag{4.1}$$

where r_i^S is the firing rate of a state cell i.

A random action cell is then stimulated manually, as follow:

$$r_i^A = 1 \tag{4.2}$$

where r_i^A is the firing rate of a randomly chosen action cell i.

The firing of this randomly-chosen action cell causes the agent to move in a corresponding direction from its current state.

The SA cells are now activated by a combination of state and action activity, as follows:

$$h_i^{SA} = \sum_j w_{ij}^{S-SA} r_j^S + \sum_j w_{ij}^{A-SA} r_j^A \tag{4.3}$$

The state input that the SA cells are now receiving is the same as before, because the agent has not yet moved into the successor state. Full winner-take-all inhibition is applied, which leaves only a single cell in the SA layer active:

$$r_i^{SA} = \begin{cases} h_i^{SA} & h_i^{SA} = \max_i \left\{ h_i^{SA} \right\} \\ 0 & h_i^{SA} < \max_i \left\{ h_i^{SA} \right\} \end{cases} \tag{4.4}$$

where $\max_i \left\{ h_i^{SA} \right\}$ is the activation value of the most active cell in the SA layer.

Hebbian learning strengthens the afferent synapses that this winning SA cell receives from state cells as follows:

$$\Delta w_{ij}^{S-SA} = k r_i^{SA} r_j^S \tag{4.5}$$

where k is a learning rate constant, r_j^S is the firing rate of a state cell j and r_i^{SA} is the firing rate of a SA cell i. This strengthening only occurs if the connection w_{ij}^{S-SA} is within the subset of permitted connections

(as described earlier, in Sec. 4.1.2, and shown in Fig. 4.2). Otherwise w_{ij}^{S-SA} remains 0.

At the same time, Hebbian learning also strengthens the afferent synapses that the currently firing SA cell recieves from the firing action cell as follows:

$$\Delta w_{ij}^{A-SA} = k r_i^{SA} r_j^A \tag{4.6}$$

This strengthening is restricted in the same way as the strengthening of w_{ij}^{S-SA}; any given w_{ij}^{A-SA} must be within the subset of permitted connections on order to be strengthened. The synaptic weights w_{ij}^{S-SA} and w_{ij}^{A-SA} are then rescaled according to the following rule:

$$\sum_i w_{ij}^{S-SA} = t^{S-SA} \quad \forall j \tag{4.7}$$

$$\sum_i w_{ij}^{A-SA} = t^{A-SA} \quad \forall j \tag{4.8}$$

Finally, the firing rates of all state, action and SA cells are then set to 0.

4.1.5 *Results*

We use an information-theoretic measure to test whether the model has correctly self-organised SA cell responses. This measure tests whether the response of an SA cell can be linked to a specific combination of state-action inputs. An SA cell has high single-cell information if its activity is sufficient to predict the presence or absence of a particular SA combination; in other words if that SA cell reliably responds to one state-action combination and reliably *does not* respond to any other state-action combination. If an SA cell has maximum information, that cell's responses uniquely identify a single state-action combination.

This measure is calculated as follows:

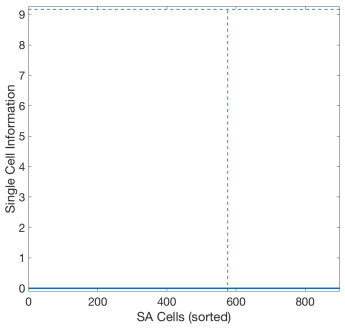
$$I(s, \vec{R}) = \sum_{r \in \vec{R}} P(r|s) \log_2 \frac{P(r|s)}{P(r)} \tag{4.9}$$

where s is a particular stimulus (a particular state-action combination) and $\vec{R}$ is is the set of recorded responses of the SA cell.

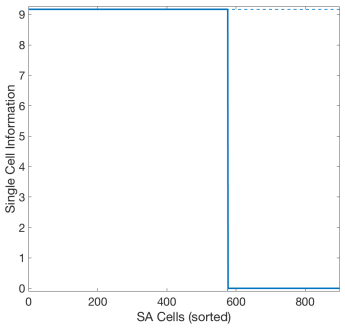
To generate this measure, we generate every combination of state-action that could be encountered in the environment on which the model was trained. This is every action, taken in every state that is not covered by a wall. (The blue area in Fig. 3.6a contains 64 free states and so 64*9 = 576 state-action combinations). For each combination, the appropriate state cells and action cells are activated, and this activity propagates to the SA cells as in eq. 4.3. These cells were subjected to WTA inhibition (eq. 4.4). The responses of these cells are recorded and used to calculate the amount of information that each SA cell's response gives about the stimulus.

Figure 4.3 shows the single cell information encoded by a layer of SA cells before and after training in an 8x8 open environment. This figure shows that, after learning, a number of cells have self-organised to encode the maximum single cell information $\log_2\left(N^{SA}\right)$where N^{SA} indicates the number of possible state-action combinations. In other words, these cells' responses uniquely identify a single state-action combination. The number of SA cells with maximum information is equal to the number of SA combinations that the model is capable of experiencing.

We found that when the model was trained in an open 8x8 environment, as in Fig. 3.6a, no SA cells fire uniquely for one state-action combination before training. However, after training, all state-action combinations are uniquely represented by one SA cell. These cells provide comprehensive coverage of the experienced state-action space and so can be used to create a comprehensive cognitive map using the learning mechanisms described in the next chapter.

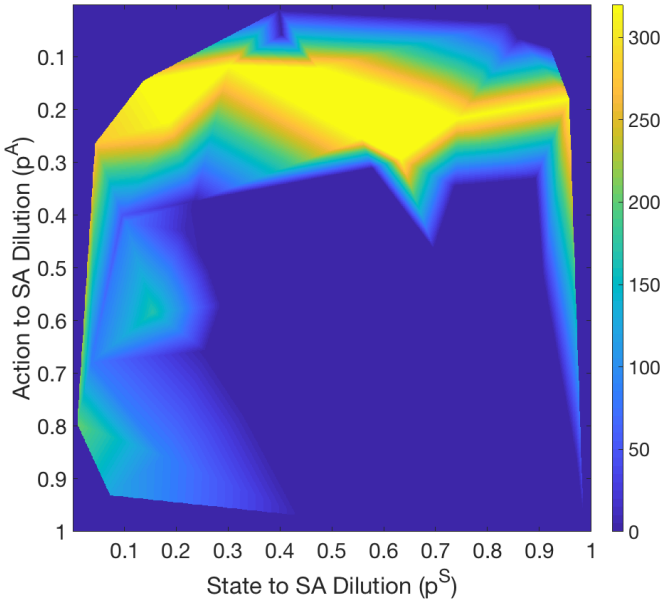


(a) The level of single cell information in the SA layer before training.

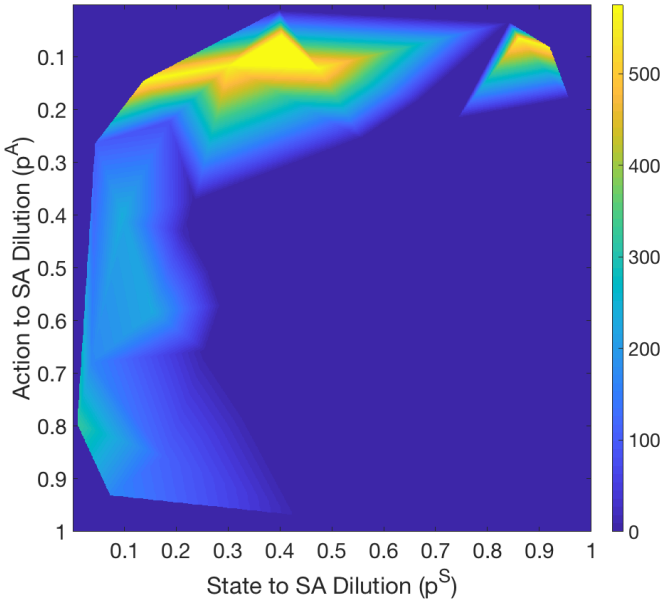


(b) The level of single cell information in the SA layer after training.

Figure 4.3: Single Cell Information Before and After Training Using Diluted Synaptic Connectivity. The model is trained in an open 8x8 environment, as in Fig. 3.6a. The horizontal dashed line indicates the maximum information that each neuron can possess, $\log_2\left(N^{SA}\right)$ where N^{SA} indicates the number of possible state-action combinations. At this stage each state-action cell learns one state-action combination, and so only N^{SA}neurons should hold significant information about the current state/action inputs. This value is indicated by a vertical dashed line. After 5000 timesteps the model has self-organised the maximum number of SA cells with maximum single cell information.



(a) The effect of altering dilution parameters in an 8x8 open environment, with five available actions per state.



(b) The effect of altering dilution parameters in an 8x8 open environment, with nine available actions per state.

Figure 4.4: The effect of altering dilution parameters p^S and p^A on the number of SA cells that—after training—display maximum single cell information. An agent explores an 8x8 open environment using either five actions (travelling in the four cardinal direction, or remaining in the current state) or nine actions (the four cardinal directions (N, E, S, W), movement diagonally, or remaining in the current state). 50 trials were run using different dilution parameter combinations, generated by a linear hypercube method [105] and the number of SA cells which self-organised to encode the maximum single cell information were displayed as a heatmap. These results have been interpolated. We see that no SA cells encode maximum information when no dilution is present (1 on the x-axis and y-axis) and we can therefore conclude that some amount of dilution is very important for the formation of SA cells unless another mechanism is used to constrain SA cell from learning to other similar SA combinations. Sections 4.2, 4.3 and 4.4 detail other possible mechanisms.

4.1.6 *Discussion*

This section describes a method of self-organising state-action (SA) cells such that after reasonably extensive training, each state-action combination available in the environment is encoded by a single SA cell (Fig. **??**). This SA cell should respond uniquely to that specific state-action combination, so that it encodes maximum single cell information (Fig. 4.3).

This method is based on competitive learning, so that SA cells compete to respond to specific state-action combinations under heavy mutual inhibition. Under standard conditions, such competitive learning may fail to learn input combinations in several important ways. Firstly, the influence of the inputs may be unbalanced. In the case of state-action cell learning, for example, there are many more possible states for the grid world task described than there are possible actions. This means that an SA cell which has learned to encode a particular SA combination will have an 'advantage' in competing for any other SA combination that involves the same action, because it is already receiving strong activation from the action input. Therefore there is a risk that if nine actions are available to the agent, only nine SA cells may develop, each one with a receptive field that covers all of the SA combinations involving that SA cell's preferred action.

A second, related problem, is that of *continuous transformation* learning. Briefly put, if an SA cell has already learned to respond to a particular SA input combination, then that cell is liable to outcompete other SA cells in learning to respond to other similar SA combinations. For example, those with the same action or state in common. This means that individual cells may form broad receptive fields that cover most or all of the input space. This tendency is biologically plausible and forms the base principle of various models of path integration and transform-invariant visual object recognition [104, 106]. However, for the practical purpose of building a reliable network model of planning in a discrete environment (as described in Ch. 3), we require SA cell responses to be more predictable and separable. If an SA cell comes to encode every state-action combination that

involves a particular action, or every state-action combination that involves a particular state, then they essentially cease to be SA cells and become state or action cells instead. If SA cells have broad, overlapping response fields then they cease to encode a unique state-action combination. The relationship between states and SA cells becomes much more complex, and so therefore does the relationship between SA cells and successor states. This in turn is likely to affect the complexity of the cognitive map and the recurrent connectivity between SA cells. We have not modelled planning under such conditions.

In this section we therefore follow the example of previous work such as Walters et al. [104] and use diluted connectivity to prevent SA cells from extending their response fields to encompass several SA combinations. By diluting the connectivity of SA cells (Fig. 4.2) we ensure that SA cells are generally unable to encode SA combinations that are adjacent in input space. The ability of the network to self-organise its SA responses depends on this dilution, and we find that the self-organisation of SA cells is quite sensitive to the precise levels of dilution assigned to each input type (Fig. 4.4) and also that to some extent the allowable range of dilution parameters changes for different state-action spaces (compare Fig. 4.4a vs. Fig. 4.4b). Furthermore, the model does not naturally form the columnar structure used in Ch. 3. The existence of state columns, and the consequent inhibition that exists within and between these columns, is important for the planning process described in that chapter because it allows the regulation of firing within and between states.

Section 4.2 will investigate an alternative method for self-organising SA cells that naturally forms a columnar structure, is more robust to changing state-action spaces, and makes more efficient use of the SA cells available in a layer.

4.2 FORMATION OF SA CELLS USING STATE COLUMNS

4.2.1 *Statement of Problem*

All potential state-action cells receive input from state cells, which encode sensory input, and action cells (see Fig. 3.1 on page 72). The previous section described a mechanism for self-organising state-action cells using competitive learning with diluted afferent synaptic connectivity from the state cells and action cells to the SA cells. However, this mechanism did not naturally form the state column structures used for planning in Ch. 3, and was sensitive to dilution parameters as well as the shape of the state-action space. In this section, we therefore propose an alternative method for self-organising SA cell responses.

4.2.2 *Hypothesis*

We hypothesised that SA cells could self-organise to uniquely represent state-action combinations, and to do so using a state-column organisation structure, using a two part process. In the first part of the process, different columns of state action cells (managed by inter-column and intra-column inhibition) would compete to represent a state input. In the second part of the process, cells within the active column would compete to represent a given action taken in that state.

4.2.3 *Network Model*

A layer of state cells can fire to indicate the current state of the agent within its environment. A layer of action cells can provide motor feedback about actions the agent has just taken. Both state and action representations use one-hot encoding in this experiment. Distributed sensory representations are explored in Section 4.4. A final layer—designated the "SA layer" contains potential SA cells which

receive afferent connections from the sensory and action layers with full connectivity and random weights.

Competitive learning schemes produce cells which receive afferent connections from a subset of inputs, and this connectivity gives the cells their learned response characteristics. Competitive learning could thus bring about the learning of states, actions, and state-action combinations. However, ensuring that the network learns an even spread of unique state-action combinations is not entirely straightforward. Each state-action cell is intended to respond to a single state-action combination as a result of the learning process, but a risk of competitive learning is that a cell may become selective for either a state or an action exclusively and outcompete all other cells when that preferred state or action is present. Such a cell would respond to any actions taken in the preferred state, or any state in which the preferred action is performed. Table 4.2 shows this occurring in an experimental context (Condition 1).

We therefore use a constant state input with intermittent action inputs. At the beginning of the timestep, cells compete to respond to a state input and a small set of cells learns to respond to that input by means of synaptic plasticity. The network then receives an action input combined with the previous state input. This gives an advantage to the small group of cells which have learned to respond to that state, such that only they can plausibly compete to represent the current state-action combination. A single cell wins the competition and becomes a full state-action cell. Alg. 4.2 describes one timestep.

4.2.3.1 *Perceiving the Current State*

In each timestep, a state cell fires first, representing the current state (eq. 4.1). State cell firing is one-hot, so each state cell represents a single unique state and only one state cell is active at a time.

Activity propagates from the state cell layer to the SA layer, activating the SA cells in that layer:

$$h_i^{SA} = \sum_j w_{ij}^{S-SA} r_j^S \tag{4.10}$$

Algorithm 4.2 *Self-Organising SA Cells Using Competitive Learning with Columnar WTA.* This describes one timestep, based on alg. 4.1. Specifically, this algorithm is identical to that described in algorithm 4.1 except for the addition of several steps, emphasised with italics. All plasticity is Hebbian unless otherwise specified. The full sequence of events listed here occurs in every timestep.

CELL FIRING State Cells (eq. 4.1)

ACTIVITY PROPAGATION *State Cells to SA Cells* (4.10)

inhibition SA Cells: Columnar Winner-Takes-All

synaptic plasticity State Cells to SA Cells (4.5 & 4.7)

CELL FIRING Action Cells (4.2)

AGENT Agent Moves to Successor State in Grid World

ACTIVITY PROPAGATION State Cells & Action Cells to SA Cells (4.3)

INHIBITION SA Cells: Full Winner-Takes-All (4.4)

SYNAPTIC PLASTICITY State Cells & Action Cells to SA Cells (4.5, 4.7, 4.6 & 4.8)

RESET All Cells Reset to Zero Activation

where h_i^{SA} is the activation of a state-action cell i, r_j^S is the firing rate of state cell j, and w_{ij}^{S-SA} is the weight of the synapse from state cell j to state-action cell i.

The potential SA cells in the SA layer are grouped in 'mini-columns'. These mini-columns are analogous to the mini-columns used in certain alternative models[4] and each group (mini-column) can compete against other mini-columns. Each mini-column should self-organise to encode one and only one state, and so the number of mini-columns must be greater than or equal to the number of states to ensure that all possible state-action combinations are learned. Similarly, the number of state-action cells in each mini-column must be greater than or equal to the number of possible actions in each state. Minicolumns have the ability to compete with each other through their connections to inhibitory interneurons, as reviewed in Buxhoeveden 2002 [95].

Competition is applied across the SA layer in such a way that only one mini-column, representing a single state, remains active. Specific-

4 Hasselmo 2005 and Martinet 2011, see Lit. Review.

ally, for each mini-column c we compute the total activation across all of the SA cells in that mini-column according to $h^c = \sum_i h_i^{SA}$. We then identify the mini-column with the largest value of h_c. The firing rates r_i^{SA} of all cells in that mini-column are set to one, while the firing rates of all other cells in all other mini-columns are set to zero. As a result of this columnar inhibition, the firing rate r_i^{SA} of SA cells in the minicolumn with the highest total activation is 1 while the rest are suppressed to 0. We will refer to this mechanism as "Columnar WTA", standing for "columnar winner-take-all inhibition".

Only one minicolumn is now active, and Hebbian learning is applied to the synapses w_{ij}^{S-SA} that project from the state layer to the SA layer (eq. 4.5). These synapses are then rescaled (eq. 4.7).

Note that rescaling is used to bound synaptic weights instead of normalisation because it creates more distinction between strengthened and unstrengthened synapses and allows for synapses to be bounded at arbitrary limits more straightforwardly (see Fig. 4.10).

The resultant strengthening of the synapses from the currently active state cell to the currently active SA minicolumn produces a *state column:* a pool of SA cells which are linked to a specific set of sensory cues representing a particular state (location). This is depicted in Fig. 4.5.

4.2.3.2 *Taking an Action*

A random action cell is then stimulated manually (eq. 4.2). The firing of this randomly-chosen action cell causes the agent to move in a corresponding direction from its current state. The SA cells are now activated by a combination of state and action activity (eq. 4.3)

The state input that the SA cells are now receiving is the same as before, because the state cells have not yet updated to represent the new state. Cells in the SA layer have already formed strengthened synapses from the firing state cells (eq. 4.5) and so the most active SA cells are those in the state column corresponding to the current state. Full winner-take-all inhibition is applied, which leaves only a single cell in the SA layer active (eq. 4.4). The incoming state cell input (eq. 4.3) ensures that the most active SA cells in the layer are those repres-

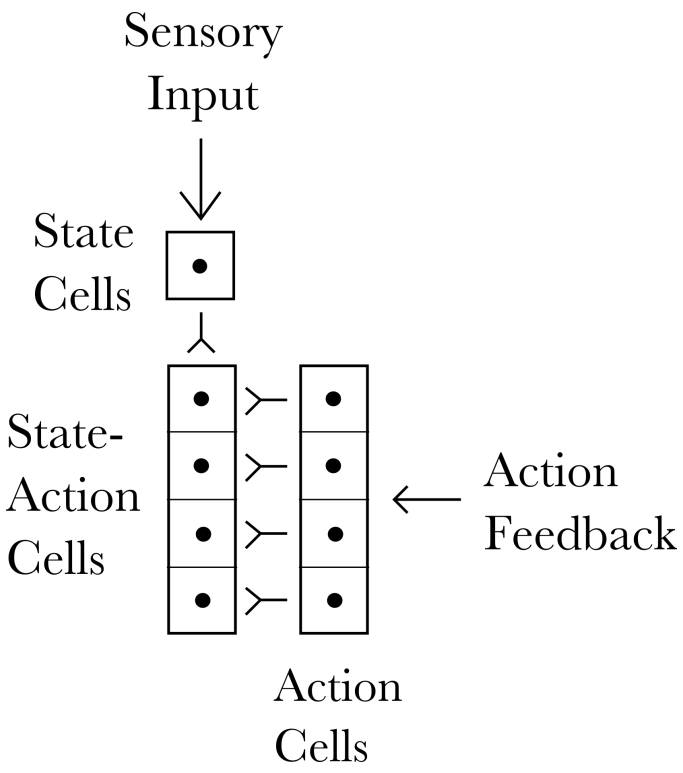


Figure 4.5: Diagram of a Single SA Minicolumn. The column contains a group of SA cells that have self-organised so that they each respond to a different action taken in the same state. Each SA cell responds to a different, unique, state-action combination. Note that the depicted connectivity has been simplified for clarity.

enting the current state. So the SA cell which wins the competition to fire will be one from the appropriate state mini-column. Hebbian learning strengthens the afferent synapses that this cell receives from state cells (eq. 4.5) and action cells (eq. 4.6) and both sets of synaptic weights are rescaled (eqns. 4.7 and 4.8).

Through these strengthened connections from the state and action cells the SA cell has learned to be activated by the current combination of state and action. Table 4.3 shows all parameters that were added or altered from previous sections.

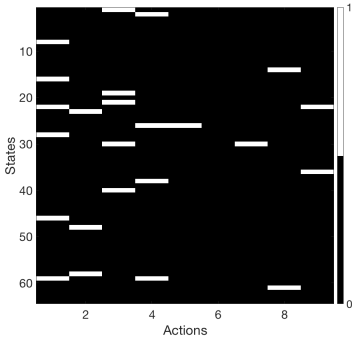
Parameter	Value
Learning Rate (k)	100
State Cell to SA Cell Rescaling (t^{S-SA})	1
Action Cell to SA Cell Rescaling (t^{A-SA})	0.25

Table 4.1: Table of Parameters for Section 4.2

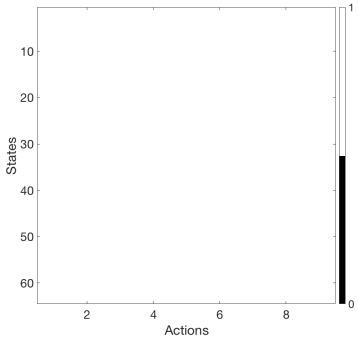
4.2.4 *Results*

Figure 4.7 shows the single-cell information (eq. 4.9) encoded by a layer of SA cell before and after training in an open 8x8 environment. This figure shows that after learning there are a number of cells with maximum information (i.e., cells whose responses uniquely identify a single state-action combination) equal to the number of SA combinations that the model is capable of experiencing. Figure 4.6 shows that the network was able to self-organise state-action cells that respond uniquely to every combination of state and action received during the task, given enough experience. These cells provide comprehensive coverage of the experienced state-action space and so can be used to create a comprehensive cognitive map using the learning mechanisms described in the next chapter.

The procedure used to produce these results deliberately separates the learning of state and action inputs in order for SA cells to become selective for particular SA combinations, as shown in Fig. 4.8. Table

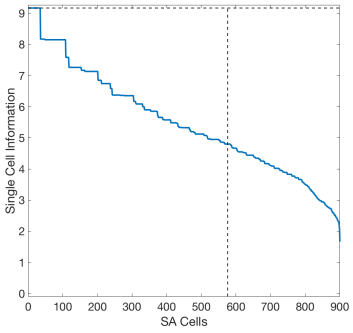


(a) The number of SA cells that fire with $r_i^{SA} > 0.90$ for each state-action combination before training.

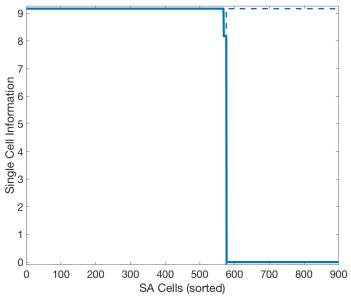


(b) The number of SA cells that fire with $r_i^{SA} > 0.90$ for each state-action combination after exploring the environment for 5000 timesteps, using the columnar WTA learning procedure described in section 4.2.

Figure 4.6: *Firing of SA Cells Before and After Columnar WTA Learning*. The model is trained in an open 8x8 environment, as in Fig. 3.6a. (a) demonstrates that before training, a small number of SA cells will fire uniquely for one state-action combination (see also Fig. 4.7a) due to the random initial synaptic connectivity between state cells, action cells and SA cells. A few state-action combinations are therefore uniquely represented by one SA cell before training. (b) then shows that after training, all state-action combinations are uniquely represented by one SA cell.

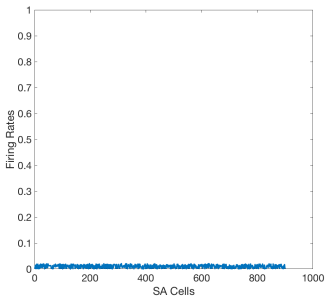


(a) Before Training

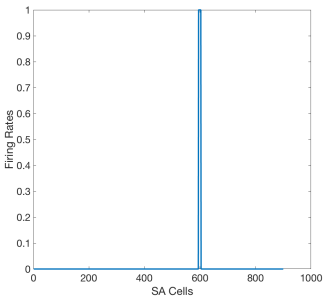


(b) After Training

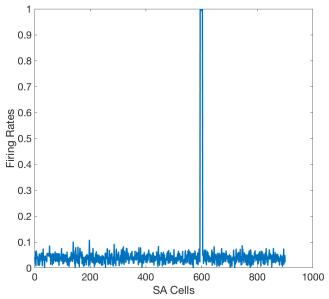
Figure 4.7: Single Cell Information Before and After Training Using State Columns. The model is trained in an open 8x8 environment, as in Fig. 3.6a. The horizontal dashed line indicates the maximum information that each neuron can possess, $\log_2\left(N^{SA}\right)$ where N^{SA} indicates the number of possible state-action combinations. Each state-action cell should only learn one state-action combination, and so only N^{SA}neurons should hold significant information about the current state/action inputs. This value is indicated by a vertical dashed line. After 5000 timesteps the model has nearly perfect SA cells for all possible SA combinations.



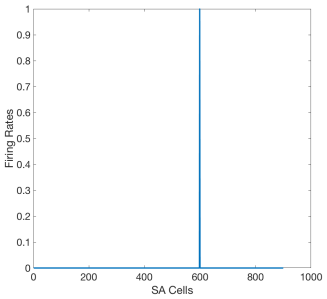
(a) Initial SA Activation from State Input



(b) SA Firing After Columnar WTA

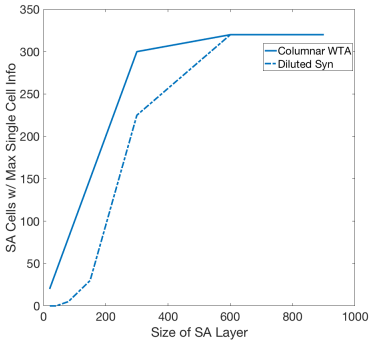


(c) SA Activation From State & Action Input

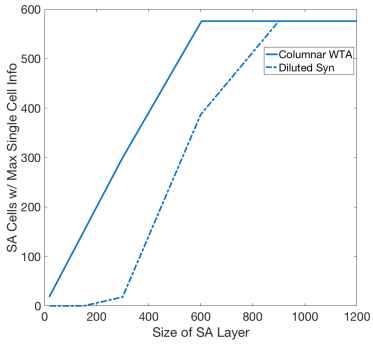


(d) SA Firing After Full WTA

Figure 4.8: Activation of SA cells over the course of the first timestep in the SA learning task. Initially, in (a), the SA layer receives state input (eq. 4.1). Since no SA cells currently represent this state, all SA cells receive a low level of activation. After lateral connectivity between columns brings about columnar winner-takes-all competition (b), a single column fires strongly. Due to Hebbian learning (eq. 4.5), this column now becomes a "state column", a minicolumn representing the current state. In (c), the SA cells are stimulated by a combination of state and action input (eq. 4.3). Since the state column which fired in (b) has learned to represent the current state, the SA cells in this column receive high activation. They are therefore the primary competitors to represent the current input combination of state and action, and we see that after *full* winner-takes-all competition (eq. 4.4) only one of the cells in this column is able to win the competition (d). Due to Hebbian learning (eqs. 4.5 & 4.7) this cell will learn to represent the current state-action combination.



(a) Comparison of SA self-organisation methods in an 8x8 open environment, with five available actions per state.



(b) Comparison of SA self-organisation methods in an 8x8 open environment, with nine available actions per state.

Figure 4.9: An agent explores an 8x8 open environment using either five actions (travelling in the four cardinal directions or remaining in the current state) or nine actions (the four cardinal directions, movement diagonally, or remaining in the current state). The y-axis of each graph denotes the number of SA cells that show maximum single cell information after training (eq. 4.9) and the x-axis shows the number of SA cells available to the model. These figures show that both the columnar WTA method and the diluted connectivity method can self-organise full SA representation of the environment provided that enough SA cells are available in the layer. However, the columnar WTA method is able to represent more state-action combinations if the suppler of SA cells is limited.

4.2 demonstrates that this is necessary; we test the model in three conditions:

1. The network receives state and action inputs simultaneously (but otherwise exactly as they would be in the standard simulation procedure).

2. State and action inputs are presented and learned separately as in the standard procedure, but no columnar WTA inhibition is used.

3. The standard procedure, with separate state and action learning and columnar WTA inhibition.

We find that in Condition 1, very few cells become responsive to particular state-action combinations. Only 9 different SA cells fired maximally during exploration (one for each available action) because these cells had self-organised to be purely action-selective: one of these 9 cells would always fire if its preferred action was taken in any state. This lack of true SA response is indicated by the fact that no SA cell encoded the maximum possible information $\log_2\left(N^{SA}\right)$ that would indicate that they had learned a strong association with a single state-action combination. More SA cells become active in the second condition, when state and action inputs are presented and learned separately as in the standard procedure, but no columnar WTA inhibition is used. However, none of them encode the maximum possible information. The last condition shows the standard procedure, with separate state and action learning and columnar WTA inhibition. The maximum number of SA cells (576) fire, and they all encode the maximum possible information, demonstrating that these cells have learned strong relationships with a single state-action combination.

Fig. 4.9 compares the self-organisation of SA cells with different SA layer sizes using either the competitive learning with diluted connectivity method as described in Sec. 4.1, or with the columnar WTA method described in this section. We find that the columnar WTA method described in this section makes optimal use of limited SA

cells by learning as many state-action combinations as there are cells in the SA layer. This contrasts with the dilution method described in sec. 4.1, which learns considerably fewer state-action combinations than there are SA cells available. Specifically, when the number of SA cells is low, a diluted network struggles to encode any state-action combinations, because cells with the right connectivity do not exist. Even with larger SA layers, the diluted method requires more SA cells to fully encode an environment than the columnar method, in order to ensure that cells with appropriate connectivity exist. The dilution method can only encode every possible state-action in the environment if there are roughly twice as many SA cells as are SA combinations. The columnar WTA method learns more SA combinations for any given SA layer size, up to the point where all SA combinations have been learned.

The network procedure described in this section used a rescaling mechanism (eq. 4.7 & 4.8) to bound the weight vectors from the state cells and the action cells to the SA cells. We compared this to the results of renormalising the synaptic weight vectors of individual SA cells i according to:

$$\sqrt{\sum_i \left(w_{ij}^{S-SA}\right)^2} = 1 \quad \forall j \tag{4.11}$$

and:

$$\sqrt{\sum_i \left(w_{ij}^{A-SA}\right)^2} = 0.25 \quad \forall j \tag{4.12}$$

We found that both methods were able to self-organise SA cells but that rescaling according to equations 4.7 & 4.8 produced a larger difference between strengthened synapses and unstrengthened synapses (Fig. 4.10). We therefore use rescaling according to equations 4.7 & 4.8 throughout the rest of this thesis. Fig. 4.11 shows that the rescaling mechanism is relatively robust—the network is able to self-

Condition	Split State & Action Learning	Columnar WTA Inhibition
1	False	False
2	*True*	False
3	*True*	*True*

(a) Condition Definitions

Condition	SA Firing	SA Cells w/ Max Information
1	9	0
2	29	0
3	576	576

(b) The ability of the network to self-organise columnar SA cells under the three different conditions described by (a). Note that the number of SA cells firing with maximum information is calculated by testing the layer at the end of an exploration run, as described in sec. 4.1.5. Only one SA cell represents each state-action combination (Fig. 4.6) and only one SA cell will remain active at the end of a learning timestep (Fig. 4.8). Under condition 1 (when state and action inputs are always presented simultaneously, and there is no columnar WTA inhibition) only nine cells are ever able to win the WTA competition to represent the inputs coming into the SA layer (only one of these will ever fire at a time). Furthermore, no cells manage to fire exclusively for one state-action combination and so no cells encode maximum information. In Condition 2, more cells manage to win the WTA competition but still no cells encode maximum information. In Condition 3, 576 cells win the competition at some point during the learning process (the same as the number of experienced state-action combinations) and the same number of cells learn to encode maximum information.

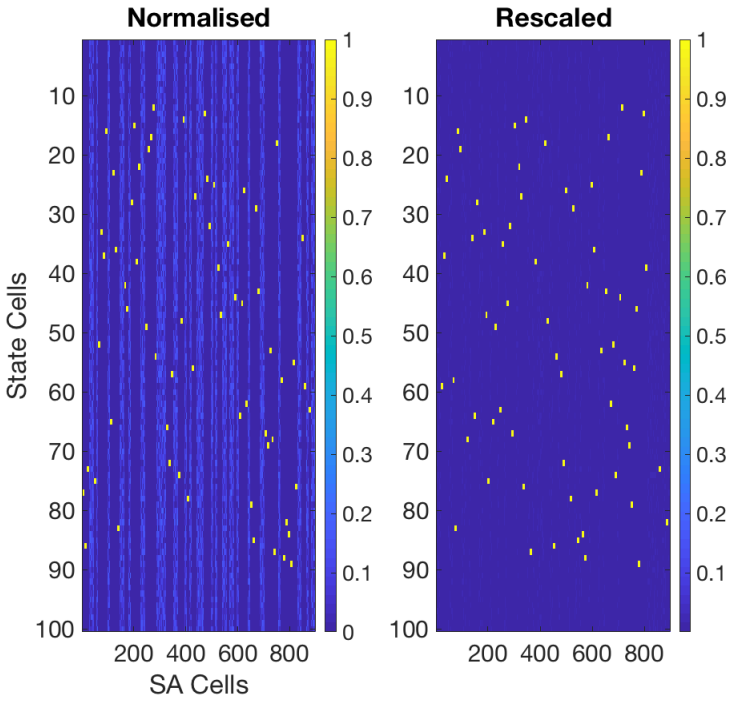
Table 4.2: The ability of the network to self-organise columnar SA cells under the different conditions (using the paradigm described in this section) depends on two factors: splitting the learning of states (when a state column forms) and the learning of state-action combinations (when an SA cells forms), and the use of columnar WTA inhibition. We test the effect of these factors using three conditions (a) corresponding to three different combinations of splitting state & action learning and columnar WTA inhibition. Table (b) shows the effect of those conditions on the model's ability to self-organise SA cells. Only when state and action learning is split and columnar WTA is used (condition 3) do the appropriate number of SA cells win the WTA competition and learn to encode potential SA combinations with maximal single cell information.

organise the maximum number of SA cells using a wide range of rescaling parameter values.

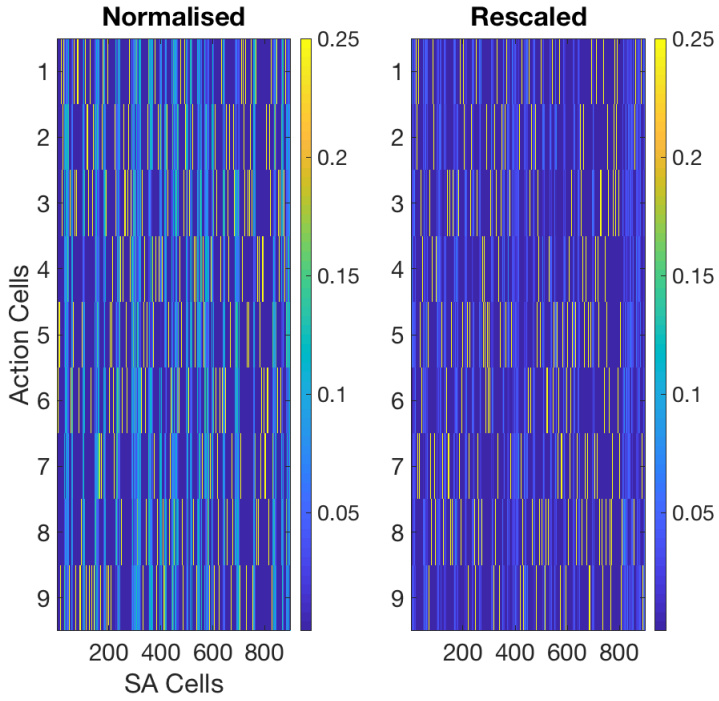
4.2.5 *Discussion*

The experiment described above showed that it was possible for SA cells to self-organise into state columns using a columnar mechanism with lateral inhibition and intermittent action inputs.This is reflected in Figure 4.7, which shows that there are a number of cells with maximum information (i.e., cells whose responses uniquely identify a single state-action combination) equal to the number of SA combinations that the model is capable of experiencing (64*9 = 576). Figure 4.6 then shows that each state-action combination is represented by an SA cell. This, coupled with the fact that the maximum possible number of SA cells encode the maximum possible information (indicating that they have learned to respond selectively to one state-action combination), demonstrates that the model has learned to fully represent all possible state-action combinations using SA cells. Table 4.2 demonstrates that both the columnar lateral inhibition and intermittent action inputs are required for the model to self-organise true SA cell responses using this method.

Although the dilution method described in Sec. 4.1 is able to self-organise SA cells without columnar winner-take-all inhibition or intermittent action feedback, the columnar WTA method described in this section has several advantages over the dilution method. Firstly, it naturally forms the columnar structure used to control the propagation of activity during planning (see below). Secondly, the columnar method makes better use of SA cells. If there are not enough SA cells to encode all possible state-action combinations in the environment, the columnar method will self-organise state columns for as many states as possible (Fig. 4.9). However, because the dilution method relies on diluted connectivity, an SA cell must have appropriate connectivity in order to encode a given state-action combination. That means that if an SA cell with the appropriate connectivity does not exist, the state-action combination cannot be learned. An interesting



(a) State cell to SA cell synaptic weights when normalised or rescaled.



(b) Action cell to SA cell synaptic weights when normalised or rescaled.

Figure 4.10: The effect of bounding the synaptic weight vectors using normalisation (so the sum of the squared synaptic weights on each postsynaptic cell i equals 1) or rescaling (so that the summed synaptic weights on each postsynaptic cell i equals 1) on the connectivity between state, action and SA cells. Normalisation and rescaling are defined by eqs. 4.11 & 4.12 and 4.7 & 4.8. We can see that rescaling produces a larger difference between synaptic weights, primarily by reducing the weight of non-strengthened synapses. This is easiest to observe in (a).

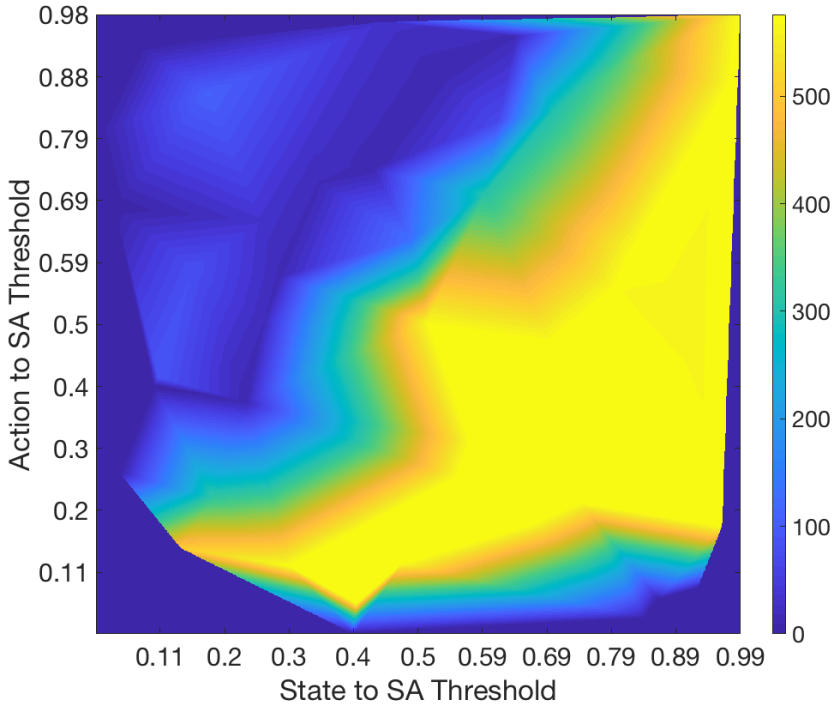


Figure 4.11: The effect of altering rescaling parameters t^{S-SA} and t^{A-SA} on the number of SA cells that—after training—display maximum single cell information. An agent explores an 8x8 open environment using nine actions (the four cardinal directions, movement diagonally, or remaining in the current state). 50 trials were run using different rescaling threshold parameter combinations, generated by a linear hypercube method [105] and the number of SA cells which self-organised to encode the maximum single cell information were displayed as a heatmap. These results have been interpolated. We see that the columnar WTA method is reasonably robust to threshold parameter changes, provided that the state to SA threshold t^{S-SA} is roughly equal to or higher than the action to state threshold t^{A-SA}. This is likely to prevent SA cells from becoming too influenced by action input.

side-effect of this is that a learner using the dilution method with a limited number of SA cells will only be able to learn a specific set of states and actions that are appropriate for the connectivity of those cells. By contrast, since the columnar WTA method allows for full connectivity and a high learning rate, we would expect that the SA layer would encode the most recent states, 'overwriting' previously learned SA cell combinations and therefore naturally 'forgetting' old or unused state-action combinations. Since real-life state spaces are very large, this is a potentially important feature of the columnar WTA method. Another possibility is that in future work, the columnar WTA method would operate with a lower learning rate and so combine states and actions in more complex representations. This could lead to the formation of state-action responses more similar to those seen in the brain [88, 94, 107] but would require the transitions between states to be represented in much more complex ways.

From a modelling perspective, the columnar WTA approach is useful because it naturally ensures the formation of state mini-columns. That is, if each SA cell encodes a combination of one state and one action, then for every state (position) there is therefore a vector of SA cells associated with that state, each representing a different action (direction of movement) that can be taken in that state. If the SA layer self-organises such that SA cells representing the same state are drawn from the same mini-column (as in Hasselmo 2005 [81] and Martinet 2011 [2]) then it becomes possible for inhibitory mechanisms to control the *relative activity between states* (by controlling the relative level of overall activity between mini-columns) whilst independently controlling *the relative activity between different actions that can potentially be taken in each state* (by controlling the relative activities of SA cells within the same mini-column).

In Ch. 3 we use this columnar structure to keep the level of activity in each active mini-column constant, allowing the propagating activity to spread through the SA layer without decaying. The columnar WTA learning model described in this section shows that state-action cells naturally self-organise into state columns given the two conditions above (separate state and action learning, and lateral colum-

nar inhibition) described in Table 4.2a. The Columnar WTA learning model is therefore able to form a structure that allows for easy management of the propagation of activation through the layer during planning. The requirement for different inputs (state, or state & action) and competition types at different points in each timestep is potentially concerning from the point of view of biological realism, however, as it is unclear how the specific order of events described in the simulation procedure for this experiment would be implemented in a more biologically realistic continuous-time model.

The next section will describe a final method for self-organising SA cell responses using a form of inhibition that forces SA cells that already encode a state-action combination not to compete to represent additional state-action combinations.

4.3 FORMATION OF SA CELLS USING BANDSTOP INHIBITION

4.3.1 *Statement of Problem*

The previous sections showed that state-action cells could self-organise using either diluted connectivity or columnar WTA inhibition. Here we show a third mechanism that could be used to self-organise SA cell responses.

4.3.2 *Hypothesis*

The main difficulty in self-organising cells that respond to a combination of (state and action) inputs is to balance the inputs such that cells learn to respond to unique input combinations rather than responding purely to one type of input and becoming invariant to the other. Without this mechanism, an SA cell which encodes State1-GoNorth will be likely to win the competition to represent State1-GoSouth; it already receives strong afferent synapses from firing state cells representing State 1 and so it is more strongly stimulated than other competing cells.

Previous sections showed two potential methods for overcoming this tendency: either by using diluted connectivity (sec. 4.1) or by making learning a two-step process, so that SA cells become selective for state inputs first and then for action inputs later (sec. 4.2). We hypothesised that the same result could be produced more simply by a form of inhibition that prevented SA cells which had already learned a particular state-action combination from learning a second one. We hypothesised that this form of inhibition would have the effect of a bandstop filter, inhibiting SA cells with medium firing rates. Such firing rates are a sign that the SA cell is receiving state input without action input, or vice versa, and is therefore currently encoding a different SA combination. Therefore, these SA cells should be inhibited for the current (different) state-action combination. Equation 4.13 gives the mathematical formulation for this inhibition.

4.3.3 *Task*

The task is identical to that used in the previous experiment. An agent is placed in a 10 by 10 grid world (either an open environment or a small maze, shown in Figs. 3.6a & 3.6b respectively). The agent is able to move in any of the eight cardinal directions. At each timestep, it chooses a random action and transitions to a new state.

As before, the agent's task is to explore the environment and self-organise SA cells to encode the state-action combinations that it has experienced. We expect that the agent, after training, should have developed state-action cells which respond strongly to an experienced state-action combination. If the SA cells do not—if the SA cells respond exclusively to sensory inputs or exclusively to action inputs, if the SA cells do not distinctly encode a single state-action combination—the agent will have performed poorly at this task.

4.3.4 *Network*

The previous model consisted of a layer of state cells, which encode the agent's current state using a one-hot encoding, and a layer of action cells, which likewise encode the action just taken by the agent. These layers are unchanged from those in the previous section unless otherwise noted.

Unlike the previous section, in this experiment the SA cell layer is a simple layer of cells and lacks a columnar structure. The connectivity between the sensory, action and SA layers is the same as before.

4.3.5 *Simulation Procedure*

Algorithm 4.3 *Self-Organising SA Cells Using Competitive Learning with Bandstop Inhibition.* This describes one timestep, based on alg. 4.1. The steps that have been added in this version are emphasised with italics.. All plasticity is Hebbian unless otherwise specified. The full sequence of events listed here occurs in every timestep.

CELL FIRING State Cells (eq. 4.1)

CELL FIRING Action Cells (4.2)

AGENT Agent Moves to Successor State in Grid World

ACTIVITY PROPAGATION State Cells & Action Cells to SA Cells (4.3)

inhibition *SA Cells: Bandstop Inhibition* (4.13)

INHIBITION SA Cells: Full Winner-Takes-All (4.4)

SYNAPTIC PLASTICITY State Cells & Action Cells to SA Cells (4.5, 4.7, 4.6 & 4.8)

RESET All Cells Reset to Zero Activation

Algorithm 4.3 describes one timestep. The state cells are updated according to the current state of the world, using a one-hot encoding. A randomly chosen action cell also fires, causing the agent to move in a corresponding direction from its current state. Both state cells and action cells propagate activity to the layer of proto-SA cells (eq. 4.3).

'Bandstop' inhibition is then applied to cells in the proto-SA layer. This inhibits activity within a certain bandstop, inhibiting the mid-level activity typical of an SA cell which only partially represents the current state-action input. This prevents one SA cell from successfully competing for all state-action combinations involving the same state, or all state-action combinations involving the same action, and so prevents the formation of single cells that respond to all of these combinations or cells that are constantly overwriting themselves. At the same time, SA cells with very low firing (i.e., those which are not yet affiliated with any SA combination) must be allowed to fire, so a simple thresholding rule is not appropriate here.

Bandstop inhibition takes place by resetting the firing rate r_i^{SA} of each SA cell i as follows:

$$r_i^{SA} = \begin{cases} h_i^{SA} & h_i^{SA} \leq t_{min}^{SA} \\ 0 & t_{min}^{SA} < h_i^{SA} < t_{max}^{SA} \\ h_i^{SA} & h_i^{SA} \geq t_{max}^{SA} \end{cases} \tag{4.13}$$

where h_i^{SA} is the activation of a cell i in the SA cell layer, and t_{min}^{SA} & t_{max}^{SA} are constants that define the limits of the suppressed bandstop. After bandstop inhibition has been applied, inhibiting those SA cells which encode a different state-action combination for the same state and therefore preventing them from using their 'unfair advantage' to compete with 'naive' SA cells with no specific encoding, winner-takes-all competition is applied by setting the firing rate of the most active SA cell to 1 and the firing rates of all other SA cells to 0 (eq. 4.4). Because of the bandstop inhibition, this winner-takes-all competition takes place purely between SA cells that genuinely encode the current state-action combination and 'naive' SA cells that can learn to encode the current combination if it is not already represented.

The synapses from the state cell layer to the SA layer are updated (eqn. 4.5), as are the synapses from the action layer to the proto-SA cells (eqn. 4.6). These synapses are then rescaled according to eqs. 4.7 and 4.8. It should be noted that the magnitude of the change in synaptic weight is enough to take them from below t_{min}^{SA} to above

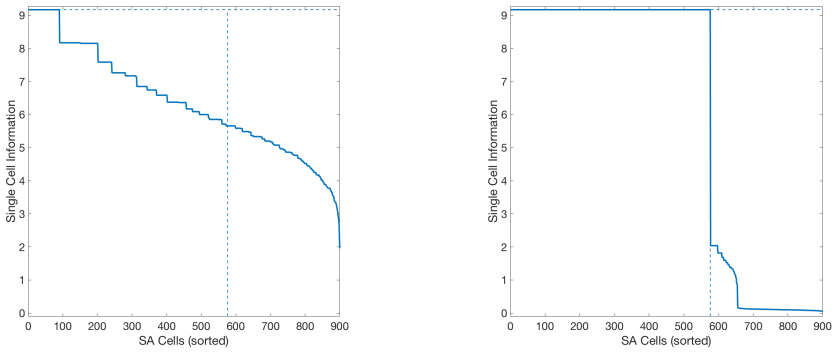
t^{SA}_{max} in one timestep. There is therefore no change of cells becoming 'trapped' in the inhibited area between t^{SA}_{min} and t^{SA}_{max}.

Table 4.3 shows all parameters that were added or altered from previous sections.

Parameter	Value
SA Cell Upper Threshold t^{SA}_{max}	1.5
SA Cell Lower Threshold t^{SA}_{min}	0.5
Action Cell to SA Cell Rescaling (t^{A-SA})	1

Table 4.3: Table of Parameters for Section 4.3

4.3.6 *Results*

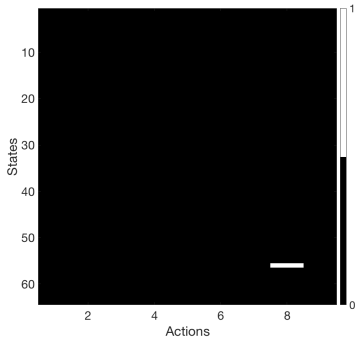


(a) The level of single cell information in the SA layer before training.

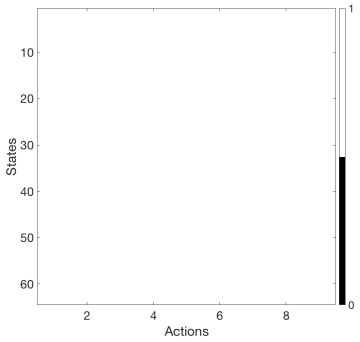
(b) The level of single cell information in the SA layer after exploring the environment for 5000 timesteps, using the bandstop learning procedure described in section 4.3.

Figure 4.12: *Average Single Cell Information Stored in SA Cells After Bandstop Learning*. The model is trained in an open 8x8 environment, as in Fig. 3.6a on page 81 for 5000 timesteps. The horizontal dashed line indicates the maximum information that each neuron can possess, $\log_2\left(N^{SA}\right)$ where N^{SA} indicates the number of possible state-action combinations (64*9=576). If each state-action cell learns one state-action combination, only N^{SA}neurons should hold significant information about the current state/action inputs. This value is indicated by a vertical dashed line.

As in section 4.2, the network was able to self-organise state-action cells for every combination of state and action received during the

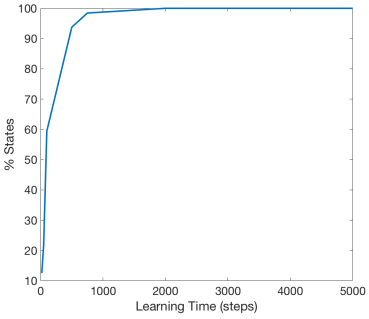


(a) The number of SA cells that fire with $r_i^{SA} > 0.90$ for each state-action combination before training.

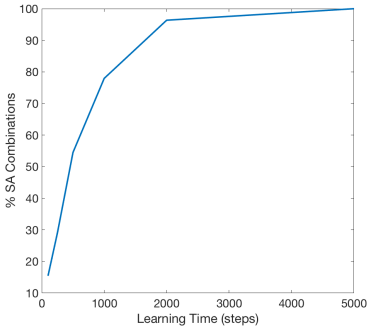


(b) The number of SA cells that fire with $r_i^{SA} > 0.90$ for each state-action combination after exploring the environment for 5000 timesteps, using the bandstop learning procedure described in section 4.3.

Figure 4.13: *Firing of SA Cells Before and After Bandstop Learning.* The model is trained in an open 8x8 environment, as in Fig. 3.6a for 5000 timesteps. The number of SA cells which fire for each state-action combination is recorded before and after training. (a) demonstrates that before training, a small number of SA cells will fire uniquely for one state-action combination (see also Fig. 4.12a) due to the random initial synaptic connectivity between state cells, action cells and SA cells. A few state-action combinations are therefore uniquely represented by one SA cell before training. (b) then shows that after training, all state-action combinations are uniquely represented by one SA cell.



(a) The number of unique states encountered during learning



(b) The number of unique SA combinations encountered during learning

Figure 4.14: This figure records how many states (Fig. 4.14a) and SA combinations (Fig. 4.14b) are encountered when the agent is allowed to explore for different numbers of timesteps. It shows that the agent has explored more than half of an open 10 by 10 state map after ~100 timesteps and almost every possible state after ~500 timesteps.

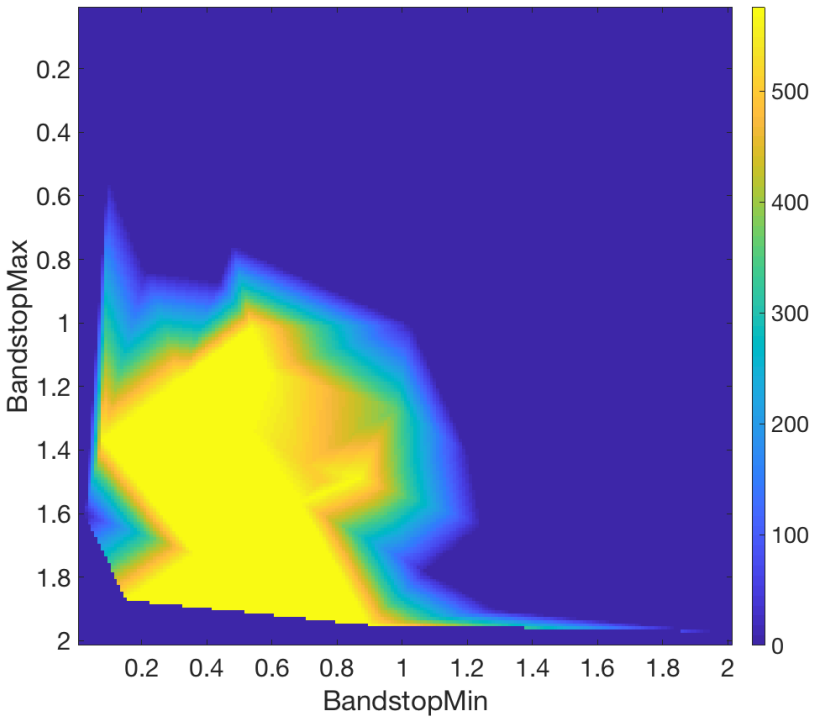


Figure 4.15: Information encoded in the SA layer after using the bandstop inhibition mechanism, dependent on the placement of the window $\left(t^{SA}_{min}, t^{SA}_{max}\right)$ for bandstop inhibition. This figure shows the number of SA cells which encode the maximum single cell information about specific SA combinations: $\log_2\left(n^S * n^A\right)$. The total number of possible SA combinations is 576. We see that the optimal results are achieved when the lower bound $t^{SA}_{min} < 1$ and the upper bound $t^{SA}_{max} > 1$.

task. Figure 4.12 shows that SA cells after learning display a very high level of single cell information, indicating that nearly all of the available state-action combinations are reliably represented by an SA cell. Figure 4.13 shows that each state-action combination is represented by an SA cell. This, coupled with the fact that the maximum possible number of SA cells encode the maximum possible information (indicating that they have learned to respond selectively to one state-action combination), demonstrates that the model has learned to fully represent all possible state-action combinations using SA cells.

Figure 4.14 shows that the agent has explored more than half of an open 10 by 10 state map after ~100 timesteps and almost every possible state after ~500 timesteps. The agent will have learned to represent at least one state-action in each state and, assuming that the agent learns how they are connected at the same time as described in the next chapter, this means (assuming the presence of a read-out mechanism) that the agent should be able to form a (convoluted) path from almost any state to almost any other state after only ~500 timesteps (Fig. 6.8). However, it takes considerably longer (about 2000 timesteps) to fully learn all of the SA combinations available in the environment and so to learn all of the possible transitions between those states, which is required for optimal planning.

Figure 4.15 shows how the lower and upper bounds $\left(t^{SA}_{min}, t^{SA}_{max}\right)$ for bandstop inhibition affect the ability of the network to self-organise SA representations. Within a particular region of the parameter space, the network is able to learn the maximum possible number of SA combinations whereas outside this space very few combinations are learned. Essentially, the level of activation of free SA cells (without either a state or action already linked to them) hovers around 0.2, so that the lower threshold t^{SA}_{min} must be above this value to allow those cells to continue to fire. The level of activation of a partial SA cell (an SA cell which either encodes the same state as the state the agent is currently in, or the same action as the agent just took, but not both) is about 1, because that is the amount of activation that such an SA cell will receive from a state or action cell with an activation of 1 and a synaptic weight of 1 (the synaptic weight is determined by the

rescaling eqs. 4.7 and 4.8, and therefore by constants t^{S-SA} and t^{A-SA}). Therefore the higher threshold t^{SA}_{max} should be greater than 1, so as to prevent those partial SA cells from firing whilst allowing full SA cells (which have an activation of about 2) to fire. The section below on learning SA cells with multiple sensory representations covers a similar situation.

When learned using the appropriate parameters, these cells provide comprehensive coverage of the experienced state-action space and so can be used to create a comprehensive cognitive map as demonstrated in Chapter 5. The use of the bandstop method does not by itself produce the state column organisation assumed by the previous chapter, but it is more flexible than the columnar winner-takes-all method described in Sec. 4.2 because the window of the bandstop filter can be moved to accomodate different levels of competing inputs. In the next section we demonstrate this by showing how a columnar winner-take-all approach can be combined with bandstop inhibition to form state-action cells when the agent's state is defined by several pieces of information.

4.3.7 *Discussion*

This chapter describes three approaches towards modelling the self-organisation of state-action cells.

The first approach, described in Sec. 4.1, uses standard competitive learning with diluted connectivity to reduce the ability of individual SA cells to develop receptive fields that cover large areas of the input space. By preventing SA cells from encoding all possible states for a given action (or vice versa) this method reduces the tendency of SA cell response fields to expand. However, because this method relies on reducing the potential connectivity of SA cells, it does not appear to make efficient use of the SA cells that are available. The right SA cell must be available with the right connectivity in order to learn any given SA combination.

The second approach, described in Sec. 4.2, uses alternating sensory and motor feedback and a system of competing cell columns to

produce SA cell responses, and this method is more complex and prescriptive than the second. This columnar winner-takes all method requires that proto-SA cells must experience a period of sensory feedback without motor feedback, followed by a period of simultaneous sensory feedback and motor feedback. It also requires that synapses update and cells activate in a strictly timed schedule, making it difficult to implement as a continuous time model. Finally, it relies on the ability of minicolumns to compete to represent inputs. Table 4.2 shows that if the timing of inputs deviates from the suggested order, or if the competition between minicolumns is removed, then the model does not properly organise SA cell responses.

There is some basis for columnar competition in neurophysiology: large basket cells are known to be able to inhibit neurons in neighbouring and distant columns [95] and are "therefore the primary source of lateral inhibition across columns within the layer that contains their somata" [108]. If large basket cells are able to allow lateral competition between columns, small basket cells have local and dense connectivity that target the soma of neurons in a particular column but no others, therefore making them potential vectors for within-column WTA of the sort that produces the final SA cell. The columnar WTA model also allows the model to organise the SA layer in a way that represents the structure of the environment using state columns (allowing you to regulate inhibition in the planning phase).

The third approach—bandstop inhibition—is simpler and in some ways more powerful. The bandstop window can be matched to the properties of the cell layer and to the properties of the task, and so this mechanism may be applicable to a wider variety of tasks with more complex sensory representations. Bandstop inhibition is not commonly used in the modelling literature but seems to arise naturally from recent experimental [109] and theoretical [110] work on inhibitory plasticity.

The next section will apply a combination of these methods to the problem of learning SA cell representations with distributed state representation inputs.

4.4 LEARNING SA CELLS WITH MULTIPLE SENSORY INPUTS

4.4.1 *Statement of Problem*

The previous sections in this chapter have described different methods for learning SA cells when the state representation is one-hot, that is to say, when there is one piece of information available that totally and uniquely defines the agent's state. This is not the case for most tasks (see the section on planning using XY coordinates in Ch. 3) as an agent's state is usually defined by several pieces of (partially) independent information. In this case the neural representation of the state will be distributed, with multiple state cells active simultaneously. Moreover individual state cells will be active for multiple different states. For example, when making coffee, the amount and position of the coffee grinds are an important indicator of your state, but so is the position of the cup, the kettle, etc. It is therefore important to see whether the model can learn to represent states that are defined in this fashion. A simple way to do so is to replace the single one-hot state representation with an xy-coordinates representation, thus defining the state with two independent pieces of information.

If the agent's current state is $\begin{pmatrix} 3 \\ 4 \end{pmatrix}$ and it then moves west, encountering the state $\begin{pmatrix} 4 \\ 4 \end{pmatrix}$ for the first time, it will need to learn a new SA cell or state column to represent this new state. If there were only one piece of information, as in the previous sections, this would not be a problem: none of the existing SA cells have a connection to this new state, so the unused SA cells will compete to represent it. One of them will win that competition and thereafter will represent this new state. However, when the distributed xy-coordinate representation is used, there is already an SA cell that is partly associated with the state $\begin{pmatrix} 4 \\ 4 \end{pmatrix}$ because that state shares the same y-coordinate

as the SA cell's true state: $\begin{pmatrix} 3 \\ 4 \end{pmatrix}$. This 'false' SA cell will therefore be more stimulated than the unused SA cells and will probably win the competition, either now representing both states or overwriting its previous associated state to represent this new state. Both are undesirable outcomes.

4.4.2 *Hypothesis*

We hypothesise that the columnar WTA mechanism used in Section 4.2 can also learn states using a distributed xy state representation, provided that inhibition is imposed to prevent SA cells from winning the competition if they are only partially associated with the current state. This can be done using the bandstop inhibition described by eq. 4.13. Such a mechanism will provide the columnar structure hypothesised by Ch. 3, such that a state column will form for each unique XY combination and therefore for each state.

4.4.3 *Method*

Alg. 4.4 describes one timestep. The state cells fire, representing the current state. The state cells in this experiment have been modified to represent the agent's state as a pair of x-y coordinates. The state layer consists of two sets of 10 cells, each encoding a coordinate from 1 to 10 using one-hot encoding. Activity propagates from the state cells to the SA cells (eq. 4.10) and the SA cells are subjected to bandstop inhibition (eq. 4.13) and then to columnar winner-takes-all inhibition (as described in Sec. 4.2.3). The synapses between the state cells and the SA cells are updated (eq. 4.5) and rescaled (eq. 4.7).

A random action cell is activated, and the agent takes the corresponding action (eq. 4.2). Activity propagates from the state layer and the action layer to the SA layer (eq. 4.3) and the layer is subject to full winner-takes-all inhibition (eq. 4.4). The synaptic weights between the action cells and the SA cells are updated (eq. 4.6) and rescaled

Algorithm 4.4 *Self-Organising SA Cells Using Competitive Learning with Distributed Representations.* This describes one timestep, based on alg. 4.2. The steps that have been added in this version are emphasised with italics. All plasticity is Hebbian unless otherwise specified. The full sequence of events listed here occurs in every timestep.

CELL FIRING State Cells (eq. 4.1)

ACTIVITY PROPAGATION State Cells to SA Cells (4.10)

inhibition SA Cells: Bandstop Inhibition (4.13)

INHIBITION SA Cells: Columnar Winner-Takes-All

SYNAPTIC PLASTICITY State Cells to SA Cells (4.5 & 4.7)

CELL FIRING Action Cells (4.2)

AGENT Agent Moves to Successor State in Grid World

ACTIVITY PROPAGATION State Cells & Action Cells to SA Cells (4.3)

INHIBITION SA Cells: Full Winner-Takes-All (4.4)

SYNAPTIC PLASTICITY State Cells & Action Cells to SA Cells (4.5, 4.7, 4.6 & 4.8)

RESET All Cells Reset to Zero Activation

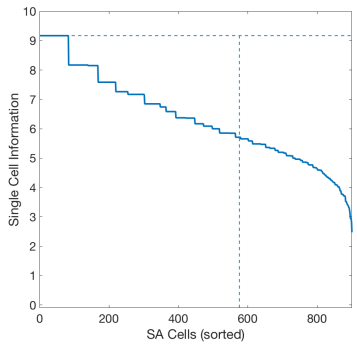
(eq. 4.8) and so are the synaptic weights between the state cells and the SA cells (eq. 4.5) and 4.7). All cell activations are then reset for the next timestep. Table 4.4 shows all parameters that were added or altered from previous sections.

Parameter	Value
Action Cell to SA Cell Rescaling (t^{A-SA})	0.25

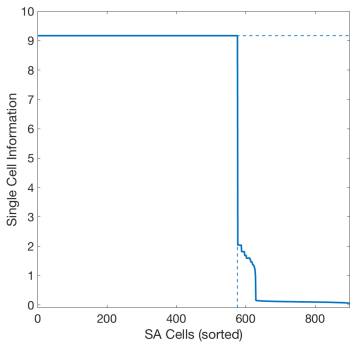
Table 4.4: Table of parameters for the model described in 4.3.

4.4.4 *Results*

Figure 4.16 shows that the model is able to learn SA cells with full single-cell information even when the state is represented in a distributed manner by two pieces of independent information (the x- and y-coordinates). Furthermore, Figure 4.13 shows that each state-action combination is represented by an SA cell.

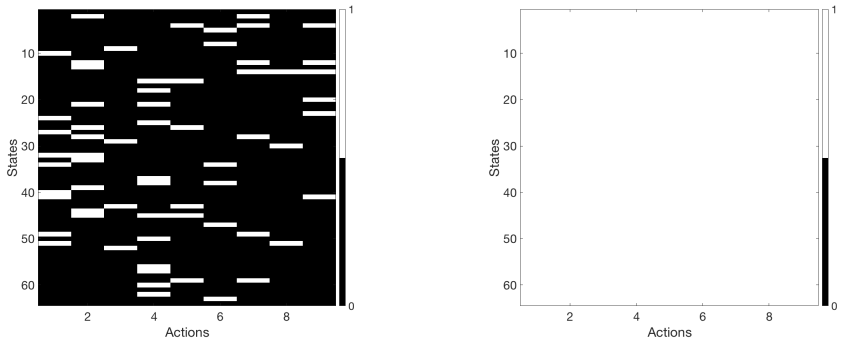


(a) Single cell information in the SA layer before training.



(b) Single cell information in the SA layer after 5000 trials of exploration with distributed sensory information, using the method described in section 4.4.

Figure 4.16: *Single cell information learned by SA cells in a task using overlapped distributed xy representations of the agent's state.* Maximum information is indicated by the horizontal dashed line, and the maximum number of SA cells that can form without redundancy is indicated by the vertical dashed line.



(a) The number of SA cells that fire with $r_i^{SA} > 0.90$ for each state-action combination before training.

(b) The number of SA cells that fire with $r_i^{SA} > 0.90$ for each state-action combination after 5000 trials of exploration with distributed sensory information, using the method described in section 4.4.

Figure 4.17: *Firing of SA cells in a task using overlapped distributed xy representations of the agent's state.* The number of SA cells which fire for each state-action combination is recorded before and after training. (a) demonstrates that before training, a small number of SA cells will fire uniquely for one state-action combination (see also Fig. 4.16a) due to the random initial synaptic connectivity between state cells, action cells and SA cells. A few state-action combinations are therefore uniquely represented by one SA cell before training. (b) then shows that after training, all state-action combinations are uniquely represented by one SA cell.

4.4.5 *Discussion*

The model is able to successfully form SA cells when the model uses overlapping distributed xy representations of the agent's state. Successfully self-organising SA cells using the method described in this section seems to require both the columnar mechanisms described in section 4.2 as well as the bandstop inhibition method described in section 4.3. The primary problem is that an SA cell which already encodes a state with a given x-coordinate or y-coordinate will be 'unfairly' advantaged when competing to represent other states with the same x- or y-coordinate. Simple competitive learning will therefore fail to produce SA cells that only encode one state. The simple formulation of the columnar winner-takes-all mechanism described in Sec. 4.2 will also fail for this reason, producing state columns that respond to more than one state.

To learn a state representation in the XY task therefore requires a method to prevent cells from 'unfairly' competing. One possibility is the diluted connectivity approach described in Sec. 4.1. In this case, the connectivity between the state cell layer and any given SA cell layer has been pruned, so that the SA cell is only capable of responding to certain state cells. This makes it harder for an SA cell to form a large receptive field that covers much of the state space, because there are many states which that SA cell is incapable of learning to respond to. However, the diluted connectivity method seems likely to be inefficient in this case. It is inherent in this mechanism that only an SA cell with the right initial connectivity can learn any given xy coordinate combination. The more complex the state representation becomes, the more cells would be needed to produce a SA cell with the right connectivity. We have therefore chosen to use a bandstop inhibition mechanism instead.

An important point to note is that this chapter and the previous chapter have presented the model in two fundamentally different 'modes' with fundamentally different dynamics. In the 'planning' mode, described in Ch. 3, activity spreads outwards through the SA layer. This activity is constrained by normalising the level of activity in

each SA column. By contrast, in the 'learning' mode described above, SA cells are subject to various forms of winner-takes-all inhibition such that only one SA column or one SA cell is active at any given time. The levels and types of inhibition are therefore fundamentally different between these two modes. Furthermore, the synapses between state, action, and SA cells are fixed during planning but are plastic during learning. These differences between the model's learning and planning modes is necessary because the planning and learning modes require fundamentally different kinds of cell responses (a propagating wave vs. a single representative cell). Producing these different representations requires different inhibition. By the same token, allowing synapses to be plastic during the planning mode seems likely to lead to incorrect learning.

The next two chapters describe how the model forms recurrent synapses between SA cells to encode the transition structure of the environment, and how the model learns the gating cells that are required for gating motor output during planning. The SA map learning mechanism is not directly affected by the state representation (provided that SA cells still form properly) and the gating cells form using competitive learning mechanisms similar to those described in this chapter.

5

SELF-ORGANISING A STATE-ACTION COGNITIVE MAP USING TRACE LEARNING

Our first simulation chapter 3 showed that a biologically plausible neural network can perform explicit action selection using a columnar layer of state-action (SA) cells. Please note that the term 'column' refers to cortical minicolumns (Sec. 3.1.2) unless otherwise specfied. The following chapter 4 then showed that these state-action (SA) cells can self-organise through competitive learning using a combination of sensory and action feedback. In the current chapter we show that a network based on the previous chapter can self-organise SA cells as before whilst simultaneously learning a cognitive map in the recurrent connections between the SA cells.

As a first step, section 5.1 demonstrates map learning using a simple network of hardwired state cells. These state cells are set up so that they each represent a different state, and they activate if and only if the agent is currently in their preferred state. We introduce a learning rule that can encode a cognitive map in the recurrent synapses between these cells $(state \longrightarrow state)$.

Section 5.2 then extends these ideas to show that this learning rule can also facilitate learning a cognitive map in the recurrent connections between state-action cells $(state + action \longrightarrow state + action)$ at the same time as these state-action cells self-organise.

The result of this chapter is a network architecture and simulation procedure that can self-organise state-action cells whilst exploring an environment and simultaneously learn to encode the structure of that environment in the recurrent synapses between these SA cells.

5.1 SELF-ORGANISING A COGNITIVE MAP IN THE RECURRENT CONNECTIONS BETWEEN STATE CELLS

5.1.1 *Statement of Problem*

A cognitive map representing a state space should encode the topological relations between states. We are faced with two problems: firstly, how should these relationships be encoded in a neural network? Secondly, how can the network learn that encoding in a biologically plausible manner during exploration?

5.1.2 *Hypothesis*

A simple way to encode the topology of a discrete environment in a neural network is to have a layer of state cells that each encode a certain state, with cells encoding neighbouring states wired together. The synapses between state cells would then encode the topological relationship between those states, and the network as a whole would encode the topology of the state space. Furthermore, activity propagating through the layer of state cells would be influenced by this topology, allowing planning. Fig. 5.1 gives a diagram of this network structure. This is of course highly simplified when compared to, for example, recorded place cells, but this simplicity makes it a good initial formulation for considering the self-organisation of a cognitive map.

An agent progressing through an environment will at each step pass from one state to its neighbour. A cell representing the original state will therefore fire just before the cell representing that neighbouring successor state.

We therefore hypothesise that the synapses between neighbouring state cells can be strengthened using a Hebbian learning rule that incorporates a memory trace on one or both of the pre/post-synaptic cells. This trace forces a cell to continue to adjust its afferent or efferent synaptic weights for a period after its firing ceases, linking it to cells that fire during this time. This learning rule therefore

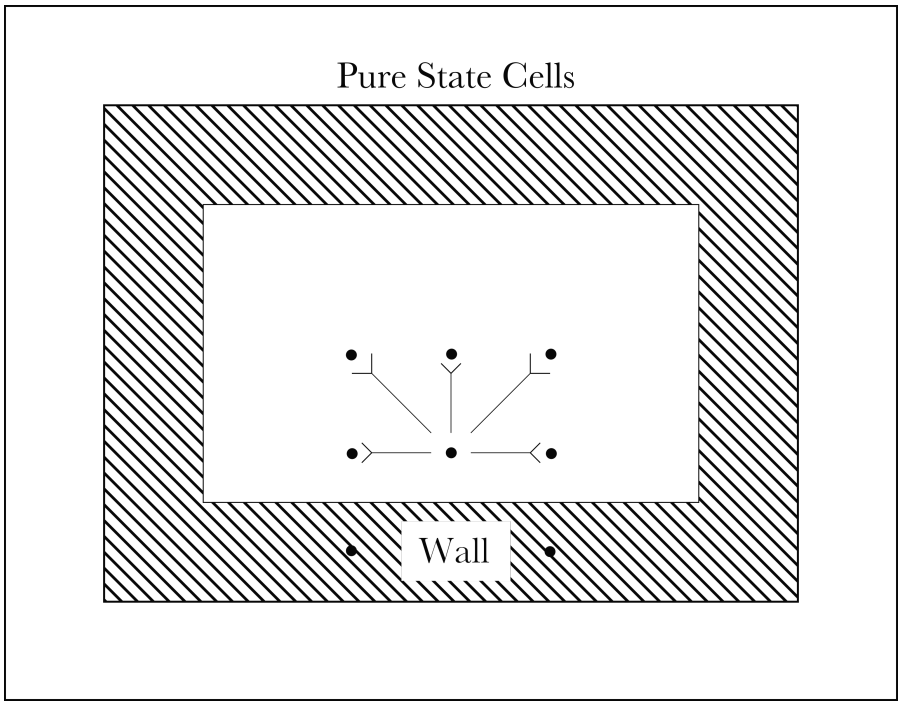


Figure 5.1: *Ideal Connectivity of a Pure State Cell.* In this simplified experiment, each state cell represents a certain position in the grid world. After learning it should ideally send synapses to its immediate neighbours (and also receive synapses from them) but would not connect or receive synapses from states that the agent is unable to reach in one step.

strengthens the synapses between two state neurons that fire in sequence, recording their spatio-temporal connection.

To test this theory, we implemented a trace learning rule in two tasks: the straight line task and the random exploration task. The straight line task demonstrates that a learning rule can encode a short linear path, and investigates what form of trace learning rule should be used to produce a reverse model similar to that used in Ch. 3. This task is described in Sections Secs. 5.1.3, 5.1.4 and 5.1.5. The random exploration task demonstrates that a trace learning rule can learn to encode all of the transitions in an 8x8 open field environment. This task is described in Secs. 5.1.6, 5.1.7 and 5.1.7.

5.1.3 *Straight Line Task*

An agent was simulated within a square open 2D world defined by a 8 by 8 grid. It was placed at a particular position in this world and moved east for several timesteps. We investigated whether the application of a trace learning rule could be used to connect neighbouring state cells that fire in succession. In our simulations, we tested three different forms of trace learning rule given by eq. 5.1, 5.2, 5.3.

5.1.4 *Network Model for the Straight-Line Task*

The network for this experiment consisted of a single 8x8 layer of state cells organised as in Fig. 5.1. Each state cell was set to become active when the agent was in a certain grid square. All-to-all recurrent synapses connected state cells, with an initial weight of 0. In other words, connectivity was full but initially of zero strength.

The agent was placed in a random position within the grid and the appropriate state cell was activated. The direction 'east' was chosen and the agent transitioned to the next grid square in this direction with 100% probability. The state cell representing the agent's new state was activated and the state cell representing the agent's old state ceased to fire.

At this point we tested one of the following three different forms of the trace learning rule to update the recurrent connection weights w_{ij} between state cells at each timestep t as follows:

$$\delta w_{ij} = k r_i^t \bar{r}_j^t \tag{5.1}$$

$$\delta w_{ij} = k \bar{r}_i^t r_j^t \tag{5.2}$$

$$\delta w_{ij} = k(\bar{r}_i^t r_j^t + r_i^t \bar{r}_j^t) \tag{5.3}$$

where r_i^t is the current firing rate of postsynaptic state cell i at time t, r_j^t is the current firing rate of presynaptic state cell j at time t, and $\bar{r}^t$ is a memory trace of the activity of the pre- or post-synaptic cell. In these simulations, we set the memory trace $\bar{r}^t$ to be equal to the firing rate of the cell at the previous timestep, i.e., r^{t-1}. These trace rules are based on those proposed by Rolls and Milward 2000 [111].

A new timestep then began. The agent then moved, a new state cell fired and the learning process repeated.

At the end of the simulation we used a method of backward induction to test whether the network had learned the spatio-temporal structure of the grid world state space. We activated either the final visited state cell, the initial visited state cell, or both of these state cells:

$$r_i^S = 1 \tag{5.4}$$

where r_i^S is the firing rate of a state cell i. Activity then propagated from this cell as follows:

$$h_i^S = \sum_j w_{ij}^{S-S} r_j^S \tag{5.5}$$

where h_i^S is the activation of a state cell i and $\sum_j w_{ij}^{S-S} r_j^S$ is the summed recurrent input that this state cell receives from the other state cells. Divisive inhibition is used to normalise activation within the layer as follows:

$$r_i = \frac{h_i}{\max h} \tag{5.6}$$

where h_i is the activation of a state cell i, r_i is the firing rate of that cell, and max h is the maximum cell activation in the layer. Parameters are given in Table 5.1.

Parameter	Value
Learning Rate (k)	1

Table 5.1: Table of Parameters for Section 5.1

5.1.5 *Results of the Straight-Line Task*

Figure 5.2 shows the observed activity in the layer if the first or last cell on the agent's path (or both) is stimulated as described in Sec. 5.1.4 and activity is allowed to propagate. The results shown demonstrate that each trace equation (5.1, 5.2, 5.3) produces a different form of cognitive map and so affects the resultant spread of activity through the recurrent synapses in different ways.

In equations 5.1, 5.2 and 5.3, the trace value $\bar{r}^t$ represents the agent's previous state and the firing rate r^t represents the agent's current state. Whether the trace value is on the presynaptic or postsynaptic term in the learning rule determines the direction of the synapses that are strengthened through learning.

In the left column of Fig. 5.2, the trace was put on the presynaptic term (eqn. 5.1), so that the cell which fired in the previous timestep reinforced its synapse to the cell which is firing in the current timestep, learning a forward model. That is, connections are strengthened from cells representing former states to cells representing later states. Stimulating the agent's initial position therefore causes activity to spread forward along its path (top left) while stimulating the agent's final position does not cause activity to spread backwards (bottom left).

In the middle column, the trace was put on the postsynaptic term (eqn. 5.2), so that the cell which is firing in the current timestep reinforces its connection to the cell which fired in the previous timestep, learning an inverse model. That is, connections are strengthened from cells representing later states to cells representing former states. Stim-

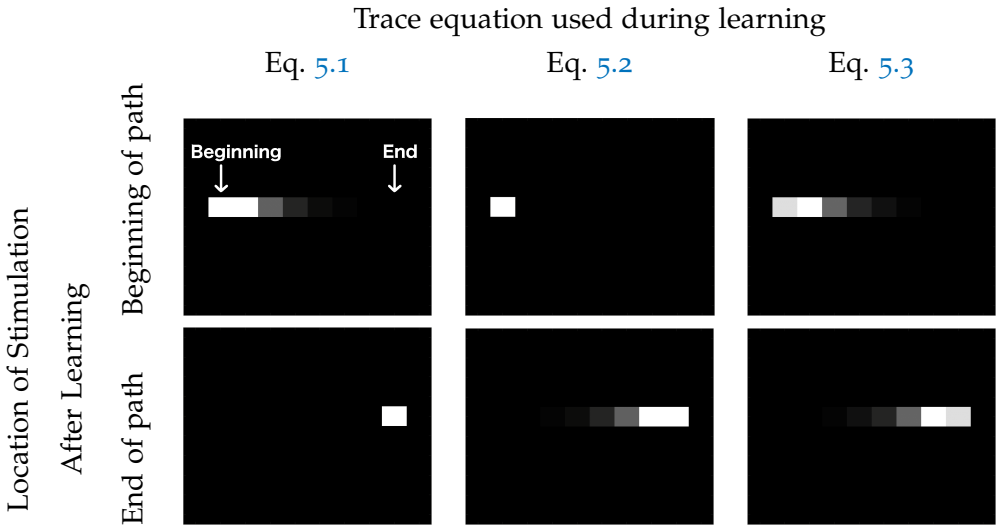


Figure 5.2: An agent moves east across a 10x10 space during learning with one of three different trace learning rules 5.1, 5.2, 5.3 used to modify the recurrent connections between the state cells. After learning, a state cell is stimulated either at the beginning or end of the path traversed during learning. The figure shows the spread of activity through the network. The activity of the state cells is represented on a grayscale, where white indicates high firing. Each column uses a different trace rule and each row shows the spread of activity from either the beginning or end of the learned path. It can be seen that the form of the trace learning rule determines how activity spreads through the network. Putting the memory trace on the presynaptic neuron (eq. 5.1) produces a forward model, allowing activity to spread forwards from the beginning of the path but not backwards from the end of the path. Putting the memory trace on the postsynaptic neuron (eq. 5.2) produces the opposite effect, a reverse model. Putting a trace on both neurons produces bidirectional connectivity, allowing activity to travel along the path in both directions.

ulating the agent's initial position therefore produces no effect (top middle) but stimulating the agent's final position causes activity to spread backward along its path (bottom middle).

In the right column, we have applied both versions of the trace rule simultaneously (eqn. 5.3), and so produced bidirectional synapses. Activity can flow through the path in either direction.

We also tried a variant procedure where cells were allowed to fire for two timesteps so that each cell overlapped with the previous cell. This situation may be regarded as more realistic, where state cells have overlapping receptive fields. Application of a simple Hebb rule created bidirectional synapses functionally identical to those produced by Eqn. 5.3.

5.1.6 *Random Exploration Task*

The wavefront model introduced in Chapter 3 relies on the recurrent connections encoding a reverse causal model. That is, a model which is able to give a set of state-action combinations that will lead to a desired state.[1] Figures 5.2 confirms that trace learning eqn. 5.2, in which the trace is in the postsynaptic term, is best for encoding the reverse direction of travel.[2]

To illustrate this, Figure 5.3 shows a simple circumstance where an agent visits a location i at time $t-1$ and then a location j at time t. When the agent reaches location j, the firing rates r and memory traces $\bar{r}$ of state cells i and j representing these locations will be as follows:

$$r_i = \text{low} \qquad r_j = \text{high}$$
$$\bar{r}_i = \text{high} \qquad \bar{r}_j = \text{low}$$

1 A forward model instead predicts the state that will results from performing any given state-action combination.

2 The state-only model described in this section does not explicitly encode actions or their consequences, and so we do not describe it as 'causal'. The state-action model described in Section 5.2 will encode actions and their consequences explictly.

In this case, the synapse w_{ij} from cell j to cell i representing the reverse direction of travel will be maximally strengthened by placing the trace on the postsynaptic cell r_i as in eq. 5.2.

Similarly to the straight-line task, an agent was simulated in a 8x8 grid world. However, in this experiment, the agent explored this world randomly and we investigated whether the application of the trace learning rule eqn. 5.2 could be used to form a complete map of the environment.

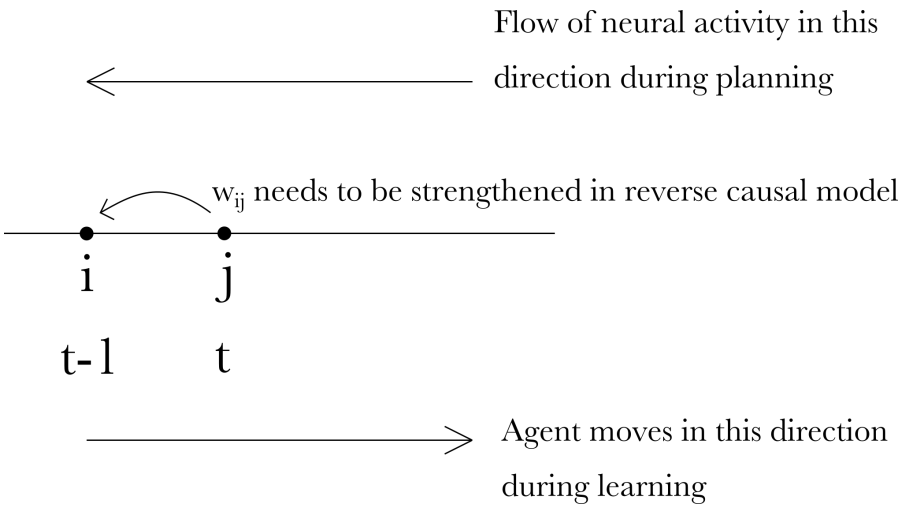


Figure 5.3: Trace Learning between two locations

5.1.7 *Network Model for the Random Exploration Task*

The network for this experiment consisted of a single 8x8 layer of state cells organised as in Fig. 5.1. Each state cell was set to become active when the agent was in a certain grid square. All-to-all recurrent synapses connected state cells, with an initial weight of 0. In other words, connectivity was full but initially of zero strength.

The agent was placed in a random position within the grid and the appropriate state cell was activated. A cardinal direction was chosen randomly and the agent transitioned to the next grid square in this direction with 100% probability. The state cell representing the agent's new state was activated and the state cell representing the agent's old state ceased to fire.

At this point we updated the recurrent connection weights w_{ij} between state cells using the trace learning rule given by 5.2. A new timestep then began. The agent then moved, a new state cell fired and the learning process repeated.

5.1.8 *Results of the Random Exploration Task*

Wishing to verify that the model described above (utilising trace learning eqn. 5.2) could indeed produce *full* and *accurate* cognitive maps of the environment, we created a ground truth transition matrix and compared the synapses learned by the model (specifically, using the inverse trace rule described in eq. 5.2 and tested in the middle column of Fig. 5.2) to this ground truth transition matrix. We counted a synapse as encoding a transition if its weight w_{ij} is greater than 0.01. Note that the default value of such synapses before learning is 0.

Having produced a ground truth transition matrix and a learned transition matrix, we can compare the two to calculate the following terms:

- True positive (tp): a transition that appears in both the true and learned transition matrices. This is a possible transition that has been learned correctly.
- False positive (fp): a transition that appears in the learned transition matrix but not in the true matrix. This is a transition that the network thinks is possible but is not.
- True negative (tn): a transition that does not appear in the true or learned transition matrices. This transition is correctly understood to be impossible.
- False negative (fn): a transition that does not appear in the learned matrix but does appear in the true matrix. This is a possible transition that the agent has failed to learn.

These terms can be used to calculate the precision (eq. 5.7) and recall (eq. 5.8) of the network. The precision represents the proportion of the learned transitions that are genuine (as opposed to false positives);

the recall[3] measures the proportion of true transitions existing in the environment that the network has successfully learned [112, 113].

$$\text{Precision} = \frac{tp}{tp + fp} \tag{5.7}$$

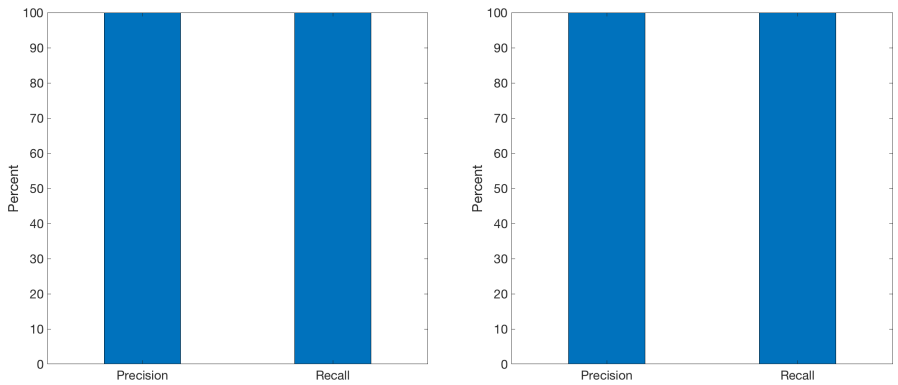
$$\text{Recall} = \frac{tp}{tp + fn} \tag{5.8}$$

where tp is the total number of true positives found when comparing the learned and ground truth transition matrices, fp is the total number of false positives, and fn is the total number of false negatives.

Figure 5.4 gives the final precision and recall measurements when the trace learning rule 5.2 is employed in an open 8x8 environment and a four-room 8x8 environment. We find that the model achieves the maximum possible precision and recall in both environments. Achieving maximum precision means that all of the transitions that the model has learned are genuine. Achieving maximum recall means that the model has successfully learned all of the transitions that exist in the environment. Fig. 5.4 therefore shows that the network can learn a perfect map of either an open field or a 4-room environment.

Furthermore, Figure 5.5 shows how the precision and recall of the proposed model varies with the amount of time that the model is given to explore the environment. This figure shows that precision (how accurately the model represents experienced transitions) is constantly at its maximum of 100%, while recall (how much of the true transition matrix has been learned) increases as the amount of available learning time increases. In other words, the model's knowledge of the transitions it has already experienced is perfect (the model always learns genuine transitions without false positives), and the model learns more about the environment with more experience.

3 Sometimes called sensitivity.



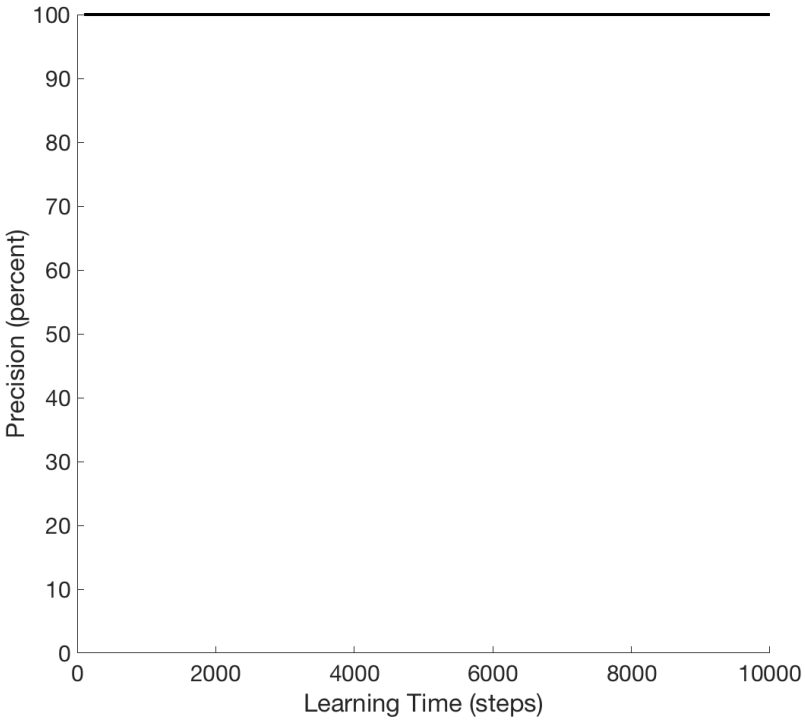
(a) Open 8x8 Environment (b) 4-Room 8x8 Environment

Figure 5.4: This figure shows the precision and recall achieved by the model after 5000 timesteps of exploration in either an open 8x8 environment (a) or an 4-room 8x8 environment (b). Precision and recall can vary between 0 and 1, and these proportions have been plotted as a percentage. We see that the model has maximum precision and recall in both environments.

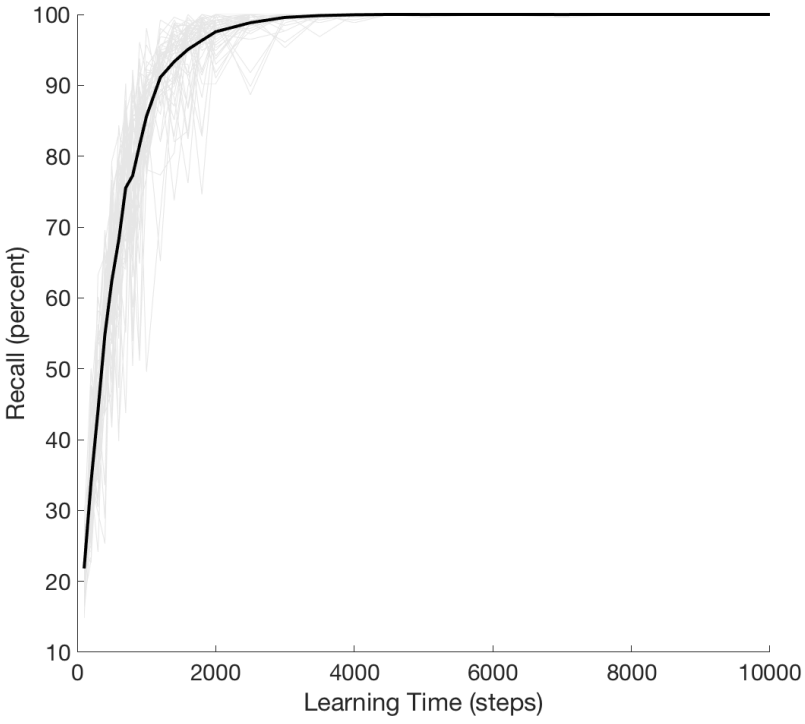
5.1.9 *Discussion*

The results of the straight line experiment (Fig. 5.2) and the random exploration experiment (Fig. 5.5) demonstrated that it was possible to use a trace learning rule to learn topological maps of a state space within a simple neural network comprised of state cells.

The trace learning rules used here are relatively biologically plausible in that they use biological information that is locally available at the synapse—i.e., the activities of the pre- and post-synaptic neurons. A recent paper by Rogerson et al. [114] reviewed evidence that molecular mechanisms exist for allowing neurons to associate events that occur close together in time. This review focuses in particular on the changes in neuronal excitability that are bought about by the transcription factor cyclic AMP-responsive element-binding protein (CREB). Neurons which express high levels of CREB are more excitable, and therefore more likely to experience plasticity. Rogerson et al. review evidence that shows that during an event, a subset of neurons start displaying high CREB. This increases their excitability so that they encode this event, and because this high CREB effect lingers, the same neurons are very likely to be involved in encoding subsequent stimuli. With time, the level of CREB expressed by these



(a) Precision



(b) Recall

Figure 5.5: The Effect of Learning Time on Precision and Recall In a Four-Room Environment. (a) shows how the precision of the learned map varies with different amounts of exploration time. (b) similarly shows how the recall of the learned map varies with different amounts of exploration time. In *both* figures, the results of 50 trials at each learning time are given as grey lines and the average of all trials is given as a thick black line.

neurons wanes, and so does their excitability. In this way a set of neurons can learn to encode a set of stimuli received during a defined temporal window.

Further experimental support for trace learning mechanisms was reported by Li and DiCarlo [115], who showed that neurons in the inferior temporal cortex learned to respond to retinal images that occured close together in time. Modelling by Evans [116] showed that trace learning could naturally occur in a 'spiking' neural network model of the visual system if the synaptic time constants were sufficiently long, such that the length of time over which a postsynaptic neuron receives input and its afferent synapses remain plastic is long enough to produce trace learning (albeit on a shorter time scale than the mechanisms described by Rogerson et al. [114]). Finally, work by Perry et al. [117] showed that the performance of human subjects on a psychophysical task (the learning of view invariant visual representations) approximated the results of a neural network model implementing trace learning.

5.2 SIMULTANEOUS SELF-ORGANISATION OF STATE-ACTION CELLS AND A COGNITIVE MAP ENCODED IN THE RECURRENT CONNECTIONS BETWEEN THEM

5.2.1 *Statement of Problem*

The overarching aim of this chapter is to explore how the cognitive map used to perform planning tasks in Chapter 3 might be learned in a biologically plausible fashion. The previous section successfully demonstrates that a local learning rule incorporating a short-term memory trace can connect cells representing neighbouring states according to the direction of passage of the agent through those states during training.

But this kind of state map is too simple for complex planning, especially in non-spatial tasks, because it does not encode the relationship between the agent's own actions and state transitions. A map described purely in terms of state cells only describes the topology

of the environment, and does not explicitly describe how an agent could transition between states.

In order to perform planning tasks we must also encode how specific actions lead to specific state transitions. In other words, we need to produce a causal map rather than a merely topological one. This section must therefore show how state-action cells can learn to encode a causal map using the lessons learned from section 5.1.

5.2.2 *Hypothesis*

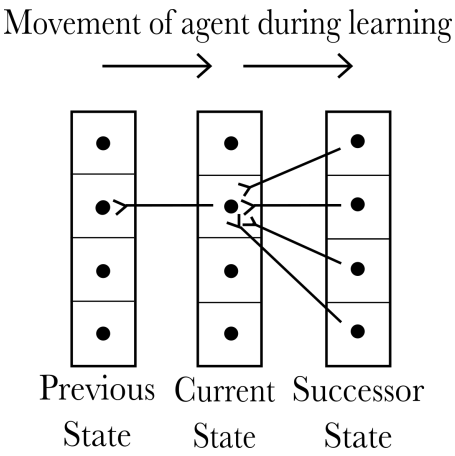


Figure 5.6: The pattern of connectivity between SA Cells encoding a reverse causal model. All SA cells within a given column encode the same state, and each SA cell encodes a different state-action combination.. We hypothesise that each SA cell should receive a set of afferent synapses from all of the SA cells encoding its predicted successor state, allowing cells in a successor state to activate the SA cell responsible for entering that successor state. Likewise, each SA cell sends an efferent synapse to any SA cells that will result in the SA cell's own encoded state.

We hypothesise that the same trace learning rules used to connect neighbouring state cells in the previous section 5.1 can also connect state-action cells, producing various kinds of cognitive maps. In particular, we hypothesise that trace learning can produce connectivity as illustrated in Fig. 5.6. This diagram shows how cells can encode an inverse causal model of the relationship between a successor state and the state-action combination which produces that successor state. That is to say, connections are strengthened *from* all of the SA cells

representing the state that the agent ends up in *to* the SA cell that encodes the state-action combination that brings about that transition.

The self-organisation of state-action cells was demonstrated in the previous chapter. This section will show a combined learning process that self-organises SA cells to respond to particular state-action combinations (as in the previous chapter) whilst also learning a causal cognitive map in the recurrent connections between the SA cells as described above.

5.2.3 *Task*

An agent is placed in a 8 by 8 grid world, either an open environment or a 4-room maze, visible in Figs. 3.6a & 3.6b respectively. The agent was able to move in any of the eight cardinal directions or remain stationary. At each timestep, it chooses a random action and transitions to the appropriate successor state. The agent's task is to explore the environment, self-organise SA cells to encode the state-action combinations that it experiences, and self-organise recurrent connectivity between these SA cells to encode the transitions that it experiences.

In other words, we expect that the agent, after training, should have recurrent connectivity between (self-organising) state-action cells which represents the transitions available in the environment. If it does not—if the recurrent connectivity encodes transitions that do not exist in the environment, or if the recurrent connectivity fails to encode transitions that do exist in the environment—the agent will have performed poorly at this task.

5.2.4 *Network Model*

The architecture of the network model used in this experiment is depicted in Fig. 5.7. A layer of state cells encode the current state of the agent within its environment. A layer of action cells encode motor feedback about the action the agent has just taken. Both state and ac-

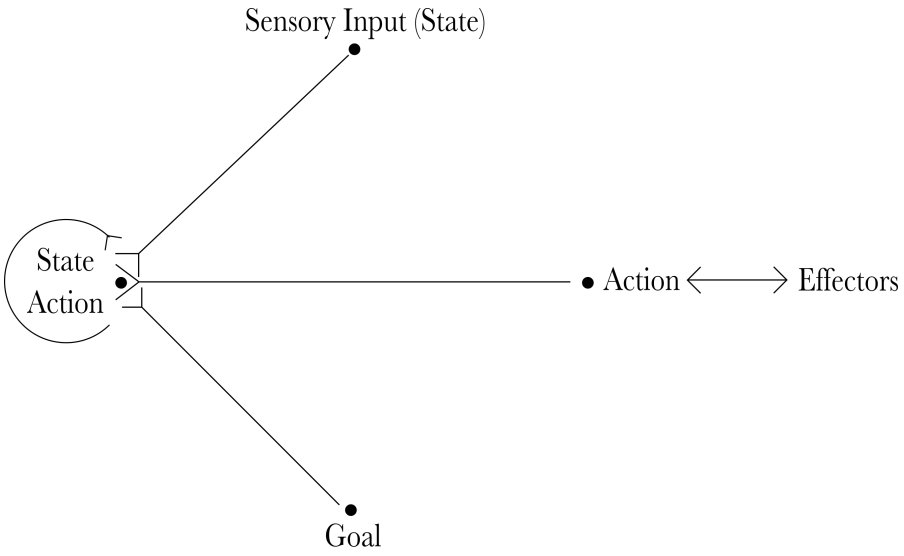


Figure 5.7: Architecture for Learning a Cognitive Map in the Recurrent Connections Between Self-Organising SA Cells

tion representations use one-hot encoding in this experiment. A final layer contained potential SA cells.

These SA cells receive full afferent connectivity from the sensory and action layers with initially random weights. The layer of SA cells is structured into a series of mini-columns. The number of these columns was made equal to the number of states and the number of cells in each column equal to the number of actions possible. The number of proto-SA cells therefore equalled the number of SA combinations necessary to learn. There is all-to-all recurrent connectivity between cells in the SA layer, but all recurrent synapses begin with a starting weight of zero.

5.2.4.1 *Perceiving the Current State*

A specific sequence of events occurs in each timestep of the learning phase, as listed in Algorithm 5.1. The relevant parameters are listed in Table 5.2. First of all, a state cell fires, representing the current state. State cell firing is one-hot, so each state cell represents a single unique state and only one state cell is active at a time.

$$r_i^S = \begin{cases} 1 & i = \text{current state} \\ 0 & i \neq \text{current state} \end{cases} \tag{5.9}$$

Algorithm 5.1 *Self-Organising SA Cells and Learning SA Map.* This describes one timestep, expanded from alg. 4.2. The steps that have been added in this version are emphasised with italics. All plasticity is Hebbian unless otherwise specified. The full sequence of events listed here occurs in every timestep.

CELL FIRING State Cells (eq. 5.9)

ACTIVITY PROPAGATION State Cells to SA Cells (5.10)

INHIBITION SA Cells: Columnar Winner-Takes-All

SYNAPTIC PLASTICITY State Cells to SA Cells (5.11 & 5.12)

synaptic plasticity Trace Learning: SA Cells to SA Cells (5.13 & 5.14)

FIRING Action Cells (5.15)

AGENT Agent Moves to Successor State in Grid World

ACTIVITY PROPAGATION State Cells & Action Cells to SA Cells (5.16)

INHIBITION SA Cells: Full Winner-Takes-All (5.17)

SYNAPTIC PLASTICITY State Cells & Action Cells to SA Cells (5.11, 5.12, 5.18 & 5.19)

synaptic plasticity Set Memory Trace: SA Cells

RESET All Cells Reset to Zero Activation

where r_i^S is the firing rate of a state cell i.

Activity propagates from the state cell layer to the SA layer, activating the SA cells in that layer as follows:

$$h_i^{SA} = \sum_j w_{ij}^{S-SA} r_j^S \tag{5.10}$$

where h_i^{SA} is the activation of a state-action cell i, w_{ij}^{S-SA} is the weight of a synapse from a state cell j to that state-action cell, and r_j^S is the firing rate of state cell j.

The potential SA cells in the SA layer are grouped in 'mini-columns'. These mini-columns are analogous to the mini-columns used in certain alternative models.[4] Each mini-column should self-organise to encode one and only one state, and so the number of mini-columns must be greater than or equal to the number of states to ensure that all possible state-action combinations are learned. Similarly, the number of state-action cells in each mini-column must be greater than or equal to the number of possible actions in each state.

Mini-columns are able to compete with all other minicolumns to fire, and competition is applied across the SA layer in such a way that only one mini-column, representing a single state, remains active. Specifically, for each mini-column c we compute the total activation across all of the SA cells in that mini-column according to $h^c = \sum_i h_i^{SA}$. We then identify the mini-column with the largest value of h_c. The firing rates r_i^{SA} of all cells in that mini-column are set to one, while the firing rates of all other cells in all other mini-columns are set to zero. As a result of this columnar WTA inhibition, the firing rates r_i^{SA} of SA cells in the minicolumn with the highest total activation are 1 while the rest are suppressed to 0.

Only one minicolumn is now active, and Hebbian learning is applied to the synapses w_{ij}^{S-SA} that project from the input state layer to the SA layer as follows:

$$\Delta w_{ij}^{S-SA} = k^{IN} r_i^{SA} r_j^S \tag{5.11}$$

4 Hasselmo 2005 and Martinet 2011, see Lit. Review.

where k^{IN} is a learning rate constant, r_j^S is the firing rate of a state cell j and r_i^{SA} is the firing rate of an SA cell i. These synapses are then rescaled so that:

$$\sum_i w_{ij}^{S-SA} = 1 \forall j \tag{5.12}$$

The resultant strengthening of the synapses from the currently active state cell to the currently active SA minicolumn produces a *state column:* a pool of SA cells which are linked to a specific set of sensory cues representing a particular state (location). This is depicted in Fig. 4.5.

5.2.4.2 *Learning an Inverse Causal Model*

As seen in Fig. 5.6, the learned recurrent connectivity between SA cells after learning should be many-to-one, such that an SA cell receives afferent synapses from every SA cell in its successor state. At this point in the timestep, the trace term $\bar{r}_i^{SA}$ encodes the SA cell that fired in the previous timestep, while the current firing of SA cells in the current state r_j^{SA} encodes all SA cells in the successor state. The desired connectivity therefore occurs if the recurrent synapses between the cells in the SA layer are updated according to a version of the trace rule described by eqn. 5.2. This rule is as follows:

$$\Delta w_{ij}^{SA-SA} = k^{TR} \bar{r}_i^{SA} r_j^{SA} \tag{5.13}$$

where k^{TR} is a learning rate constant, $\bar{r}_i^{SA}$ is a memory trace of the postsynaptic SA cell i where the memory trace $\bar{r}_i^t$ is set to be equal to the firing rate r_i^{t-1} of the cell at the previous timestep $t-1$, and r_j^{SA} is the firing rate of the presynaptic SA cell j. These synapses are then rescaled so that:

$$\sum_i w_{ij}^{SA-SA} = 4 \quad \forall j \tag{5.14}$$

The implementation of the trace learning rule 5.13 in the recurrent connections between the SA cells allows the network model to learn

an inverse causal model of how the agent's actions lead to transitions between states.

5.2.4.3 *Taking an Action*

A random action cell is then stimulated manually as follows:

$$r_i^A = 1 \tag{5.15}$$

where r_i^A is the firing rate of a randomly chosen action cell i.

The firing of this randomly chosen action cell causes the agent to move in a corresponding direction from its current state. The SA cells are now activated by a combination of state and action activity as follows:

$$h_i^{SA} = \sum_j w_{ij}^{S-SA} r_j^S + \sum_j w_{ij}^{A-SA} r_j^A \tag{5.16}$$

The state input that the SA cells are now receiving is the same as before, because the state cells have not yet updated. Cells in the SA layer have already formed synapses from the firing state cells (eq. 5.11) and so the most active SA cells are those in the state column corresponding to the current state. A new form of winner-take-all inhibition is applied, which leaves only a single cell in the SA layer active:

$$r_i^{SA} = \begin{cases} h_i^{SA} & h_i^{SA} = \left\{\max_i h_i^{SA}\right\} \\ 0 & h_i^{SA} < \left\{\max_i h_i^{SA}\right\} \end{cases} \tag{5.17}$$

where $\max_i \left\{h_i^{SA}\right\}$ is the activation of the most active cell in the SA layer.

The incoming state cell input (eq. 5.16) ensures that the most active SA cells in the layer are those representing the current state and so the SA cell which wins the competition to fire will be one from the relevant state column. Hebbian learning strengthens the afferent synapses that this cell receives from both the input state cells (through eqns. 5.11 and 5.12) and input action cells through the following equations:

$$\Delta w_{ij}^{A-SA} = k^{IN} r_i^{SA} r_j^A \tag{5.18}$$

where the synapses are then rescaled so that:

$$\sum_i w_{ij}^{A-SA} = 0.25 \quad \forall j \tag{5.19}$$

Through these strengthened connections from the sensory and action cells the SA cell has learned to be activated by the current combination of state and action.

At this point the firing rates (r^{SA}) of the SA cells are recorded and will become $\bar{r}_j^{SA}$ in the next timestep. The activity of all cells is then set to zero.

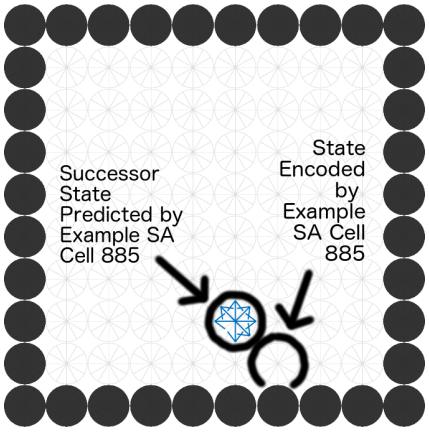
Parameter	Value
Input Learning Rate (k^{IN})	100
Trace Learning Rate (k^{RC})	10

Table 5.2: Table of Parameters for self-organising model incorporating layer of SA cells described in Section 5.2

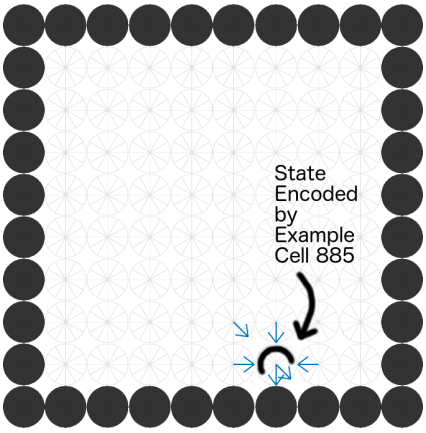
5.2.5 *Results*

Figure 5.8 shows the connectivity of a selected SA cell after exploration. This connectivity is effectively equivalent to the connectivity hypothesised at the beginning of this section (Fig. 5.6). The example SA cell has learned to encode a particular state-action combination which transitions an agent from its current state to another.

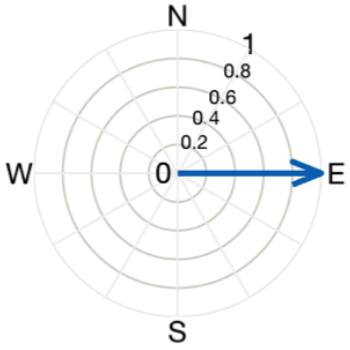
The example SA cell therefore receives afferent synapses from all SA cells encoding the state predicted by the example cell, allowing planning activity to propagate backwards from the goal location as in Ch. 3. The example SA cell sends efferent synapses to the surrounding SA cells that transition an agent into the example SA cell's own state. In other words the example SA cell receives synapses from all SA cells linked to its predicted state (Fig. 5.8a) and sends synapses to all SA cells that predict its own state (Fig. 5.8b).



(a) Cells Sending Afferent Synapses to Example SA Cell 885 (NE)



(b) Cells Receiving Efferent Synapses from Example SA Cell 885 (NE). Note that moving towards a wall returns the agent to its current state, so SA cells which represent movement south or south-east from the state encoded by cell 885 also recieve efferent synapses from cell 885. These cells are represented by blue arrows pointing towards the wall.



(c) Compass Plot Legend. Each state is represented by a compass plot; each arrow within a compass plot represents the synaptic weight of a state-action cell that represents moving in the direction of the arrow from that state. These synapses may be afferent (Fig. 5.8a) or efferent (Fig. 5.8b).

Figure 5.8: This figure shows the SA cells that are connected to an example state-action cell (885) after 5000 timesteps of training. Each state in the environment is denoted by a compass plot that shows the magnitude of the afferent or efferent synapses between the SA cells associated with that state and SA cell 885 (Fig. 5.8c explains the workings of each compass plot). Fig. 5.8a shows the cells that send afferent synapses to SA cell 885 and so can potentially contribute to its activity. Fig. 5.8b shows the cells that receive efferent synapses from SA cell 885 and which it may propagate its activity to.

To test the ability of the model to learn the transition matrix for its environment, we used the same technique as that described in Sec. 5.1.5, comparing the model's learned transition matrix (encoded in the recurrent synapses between SA cells) to the true transition matrix. We created the true transition matrix by algorithmically subjecting the agent to every combination of state and action that could be encountered in the environment on which the model was trained. This is every action, taken in every state that is not covered by a wall. (The blue area in Fig. 3.6). We then recorded the state transition that occurred to produce the ground truth transition matrix.

After producing the ground truth transition matrix, we then decoded the recurrent synapses w^{SA-SA} from the above experiment to derive the network's own learned transition matrix. In the case of a cognitive map encoded in a layer of state-action cells, as described in this section, an encoded transition is a prediction that a state-action combination will produce movement to a successor state. In neural terms, a strong synapse from a presynaptic SA cell i to a postsynaptic SA cell j encodes a prediction that the state-action combination represented by SA cell j will transition the agent to the successor state represented by presynaptic SA cell i. This connectivity is illustrated in Fig. 5.6.

We decoded the recurrent connectivity in the SA layer to identify which predicted transitions were being encoded by these recurrent connections. This decoding process took place as follows.

Again, we generated every combination of state-action that could be encountered in the environment on which the model was trained. For each combination, the appropriate state cells and action cells are activated, and this activity propagates to the SA cells as in eq. 5.16. The firing rates of these cells was then rescaled so that all firing rates were between 0 and 1, and any SA cell which was firing at a rate > 0.99 was selected. These cells were considered to represent that state-action combination. We could then decode the agent's learned transition matrix by analysing each synapse w_{ij}^{SA-SA} and recording it as a transition from the state-action combination represented by SA cell i to the state represented by SA cell j. We count a synapse

as encoding a transition if its weight w_{ij} is greater than 0.01. The default value of such synapses before learning is 0. Note that because the SA synapses encode a backwards model of the environment,[5] the direction of the synapse must be reversed to get the direction of the transition.

Having produced a ground truth transition matrix and a decoded learned transition matrix, we can compare the two to calculate the following terms:

- True positive (tp): a transition that appears in both the true and learned transition matrices. This is a possible transition that has been learned correctly.
- False positive (fp): a transition that appears in the learned transition matrix but not in the true matrix. This is a transition that the network thinks is possible but is not.
- True negative (tn): a transition that does not appear in the true or learned transition matrices. This transition is correctly understood to be impossible.
- False negative (fn): a transition that does not appear in the learned matrix but does appear in the true matrix. This is a possible transition that the agent has failed to learn.

where tp is the total number of true positives found when comparing the learned and ground truth transition matrices, fp is the total number of false positives, and fn is the total number of false negatives.

These terms can be used to calculate the precision (eq. 5.7) and recall (eq. 5.8) of the network. The precision represents the proportion of the learned transitions that are genuine (as opposed to false positives); the recall[6] measures the proportion of true transitions existing in the environment that the network has successfully learned. Figure 5.9 shows both measures for the model described in this section.

5 The reverse causal model encoded in the recurrent synapses using a trace learning rule with the memory trace in the postsynaptic term represents how a given state may be reached by a state-action combination, rather than how a given state-action combination produces a new state. This is discussed more thoroughly in Ch. 3.

6 Sometimes called sensitivity.

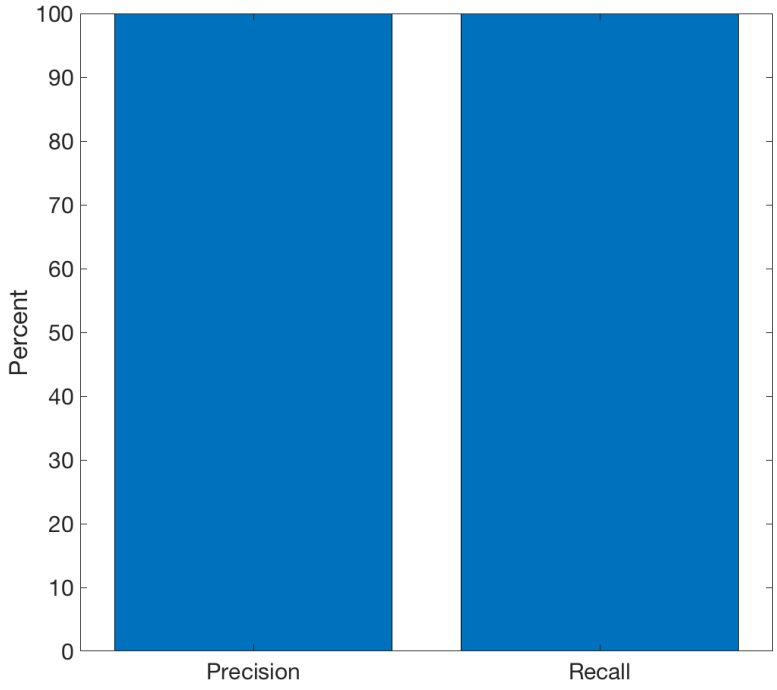


Figure 5.9: This figure shows the precision and recall achieved by the self-organising columnar SA model described in Section 5.2. This model was run for 5000 timesteps of exploration in an open 8x8 environment. Precision and recall can vary between 0 and 1, and is plotted here as a percentage. We observe that both precision and recall are at 100%. The model is able to represent all of the true transitions existing in the environment, and represents them all correctly.

5.2.6 *Discussion*

Section 5.1 shows that trace learning is able to link neighbouring state cells in such a way as to store the topological relationships between states in the synaptic connectivity between these cells (Fig. 5.1).

The same principle applies in a layer of SA cells. At each timestep the network receives sensory & delayed action feedback and—if no SA cell already responds to this input combination—an SA cell learns to represent this combination. This SA self-organisation is a two-step process, with a column of SA cells learning to represent the current state and one SA cell in that column learning to further respond to an action take in that state. This means that the self-organisation process produces a 'state column' made up of SA cells that all respond to the current state.

The agent's action transitions the agent into a new state (location) at the next timestep where it receives a new set of sensory cues, prompting the network to learn another 'state column'. Trace learning strengthens recurrent connections between the single[7] postsynaptic SA cell (representing the former state and the action that was taken in that state) and the SA cells in the mini-column representing the current state. This results in the many-to-one causal synapses we illustrated in Fig. 5.6 and showed in Fig. 5.8. The resultant network architecture encodes a causal cognitive map of its environment in terms of states and actions, such that it can plan using the inductive process described in Ch. 3.

Using a columnar architecture appears to be necessary for the planning stage but also greatly improves the efficiency of learning during exploration: the columnar WTA allows an SA cell to receive synapses from *every* SA cell in the resultant state (i.e., in the active state column), rather than simply the SA cell that is next to fire. The network described in this section has 9 possible actions that it can perform in each state and so using a columnar architecture will on aver-

7 Winner-take-all competition always leaves one cell active, with the exception that if there are no cells firing in the layer then WTA competition will not generate an active cell.

age learn a full map after only experiencing 11% of the possible SA combinations.

The SA mapping mechanism described in this section relies on the use of a 'memory trace' that allows a presynaptic neuron to continue strengthening synapses to a postsynaptic neuron even after the postsynaptic neuron has stopped firing (eq. 5.2). There are certain biological mechanisms that produce similar effects. In particular, experimental research has shown that cells which have previously experienced high firing rates may show increased excitability for some time afterwards, an effect which appears to be caused by the protein CREB [114]. The raised excitability produced by CREB could potentially allow connections to form between SA cells that fire in succession, producing connectivity similar to that produced by a trace rule. CREB is often implicated in the formation of episodic memory traces. Evans and Stringer demonstrated a somewhat similar effect by extending the amount of time that postsynaptic activity or presynaptic activity takes to decay in a spiking cell model and showing that this allows these cells to remain active long enough to receive and associate several synaptic inputs in sequence [116].

The effects of CREB are not a perfect fit for the trace learning rule, however. CREB aids episodic memory formation by increasing the excitability of a cell that has previously fired, making it more likely to fire again in response to further stimuli. This is in contrast to the memory trace employed in the previous experiment, which allows a presynaptic neuron to continue strengthening synapses to a postsynaptic neuron *even after the postsynaptic neuron has stopped firing* and so allows the SA cell which fired in the previous timestep to connect the the currently active SA column without strengthening its connections from the currently firing state and action cells. If the SA cell encoding the previous state-action combination remains active (due to CREB-based excitability) as the agent moves to the next state, we would expect it to start forming connectivity to this new state, interfering with the self-organisation of a new SA cell.

Secondly, the simulation procedure used in this section requires careful timing of inputs (as in Sec. 4.2), competition and trace form-

ation/learning. Recurrent connectivity between SA cells is learned according to a function of present SA firing and previous SA firing (via a memory trace). This means that it is very important which cells are firing when the SA memory trace is calculated. To form the synapses from state-column to SA cell shown in Fig. 5.6, the post-synaptic trace $\bar{r}$ must be set once WTA inhibition has produced a single SA cell for the previous timestep, but trace learning must occur while the whole state column is active for the current state. An alternative is to have the trace formation and trace learning happen at the same time, which will connect the specific SA cell that fired in the previous timestep (representing the state-action combination from that timestep) to the specific SA cell firing at the end of the current timestep. This would produce connectivity similar to the model described in the next section, which would be less efficient but should still work. Another alternative is to have trace formation and trace learning occur while the entire state column is still active, but this would produce false connectivity because it would connect all SA cells in the current state column to those in the previous state column, regardless of whether a causal connection exists. In effect this would produce a state-to-state cognitive map similar to that described in the previous section.

6

SELF-ORGANISATION OF A NEURAL READ-OUT MECHANISM FOR ACTION SELECTION

The previous experimental chapters of this thesis have described a family of neural networks for planning solutions to a navigation task in a simple grid environment. The question of how such a network should output and use the results of that plan is not trivial. To produce an action based on an internal planning process requires several linked processes: firstly, the network must have a mechanism to determine when the planning process is complete to some degree of satisfaction; secondly, the network must have a mechanism to determine from the planning process what action is considered best to take in the current state. In other words, the network must read off the right action at the right time (figs. 6.1 and 6.2).

The work in this chapter therefore combines two problems: firstly, by what process should the model output an action? Secondly, what neural mechanism is able to perform this process, and how might it self-organise in the brain?

6.1 SELF-ORGANISATION OF GATING CELLS USING BANDSTOP INHIBITION

6.1.1 *Statement of Problem*

Most of the literature to date [2, 80, 81] posits that a gating mechanism is most suitable for this task. That is to say that an element of these models ensures than an action cannot be passed to the motor effectors unless a certain set of conditions are met. This gating mechanism is generally implemented as literal "gating cells" which receive input from both the sensory mechanism (representing the current state) and the planning mechanism, and which will only fire when they re-

ceive sufficient input from both. In other words, the signals from the route planning layer are gated by the current state of the agent before being passed to the action layer. These gating cells act as conduits of information from the planning mechanism to the motor effectors.

However, there remains the question of how these gating cells acquire their distinctive connectivity (Fig. 6.3) and properties.

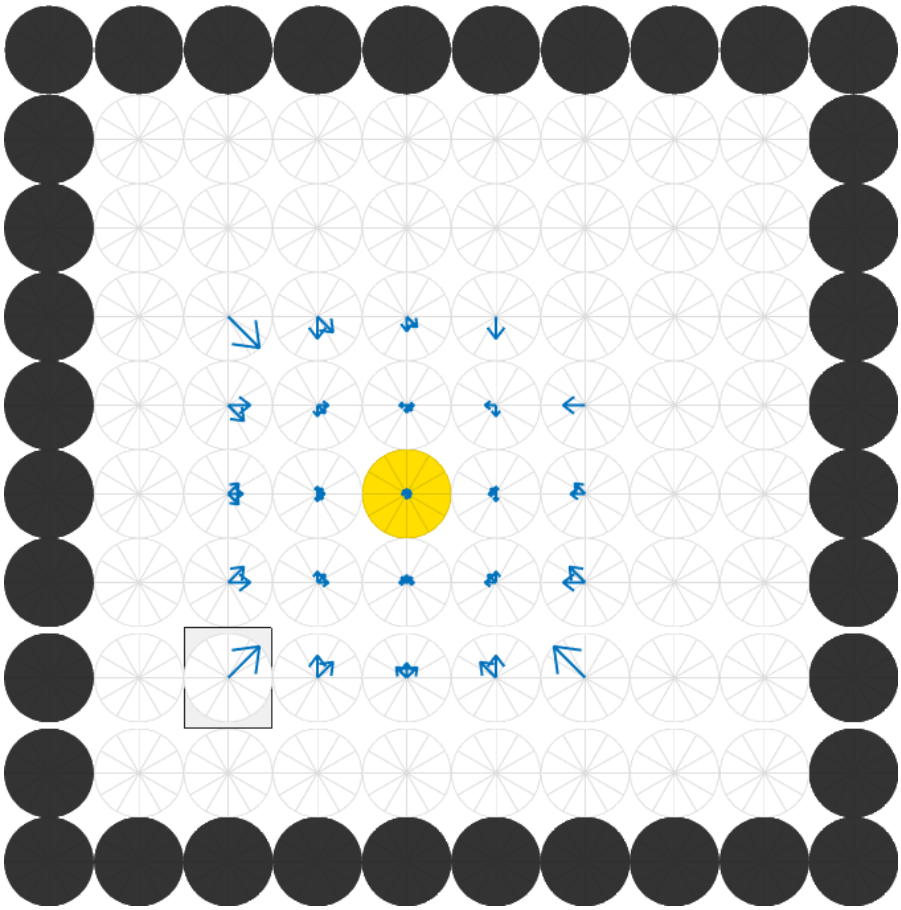
6.1.2 *Hypothesis*

We hypothesise that the self-organisation of the gating cells can be achieved by a form of competitive learning fundamentally similar to that of self-organising state-action cells. This is because SA cells must also learn to respond to a specific combination of two inputs. We therefore hypothesise that the bandstop inhibition method described in ch. 4 can self-organise gating cells' connectivity.

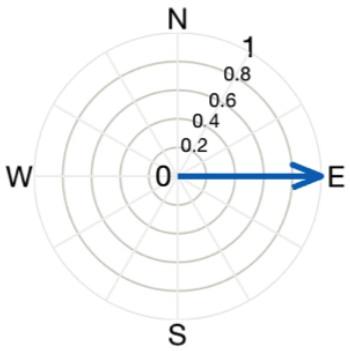
6.1.3 *Task*

An agent is placed in a 10 by 10 grid world (either an open environment or a small maze, shown in Figs. 3.6a & 3.6b respectively). The agent is able to move in any of the eight cardinal directions. At each timestep, it chooses a random action and transitions to a new state. The agent's task is to explore the environment and self-organise gating cells to encode the state-action combinations that it has experienced (along with SA cells as in the previous two chapters).

We expect that the agent, after training, should have developed gating cells which respond strongly to an experienced state-SA combination. If it does not—if the gating cells respond exclusively to sensory inputs or exclusively to SA inputs, if the gating cells do not distinctly encode a single state-SA combination—the agent will have performed poorly at this task.

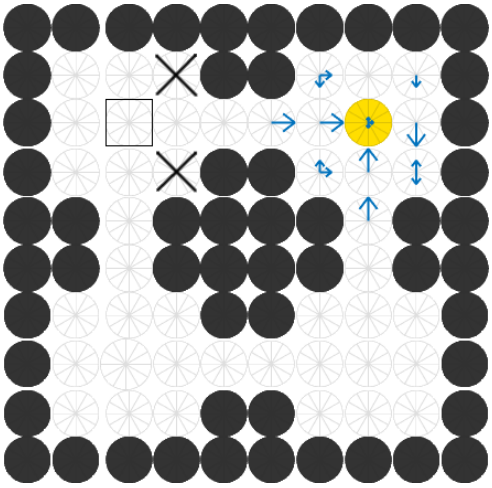


(a) SA Firing Rates During a Navigation Task in an 8x8 Open Environment

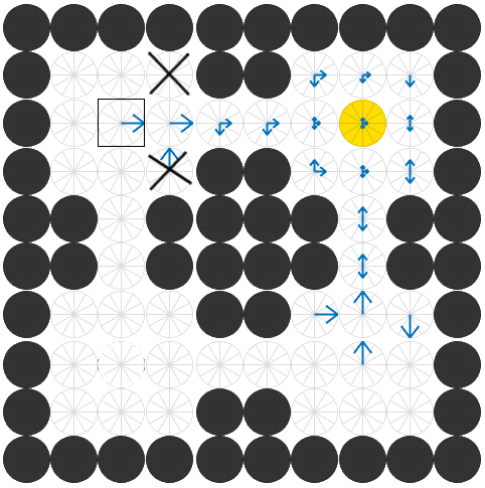


(b) *Compass Plot Legend.* Each state is represented by a compass plot; each arrow within a compass plot represents the firing rate of a state-action cell that encodes movement in the direction of the arrow from that state.

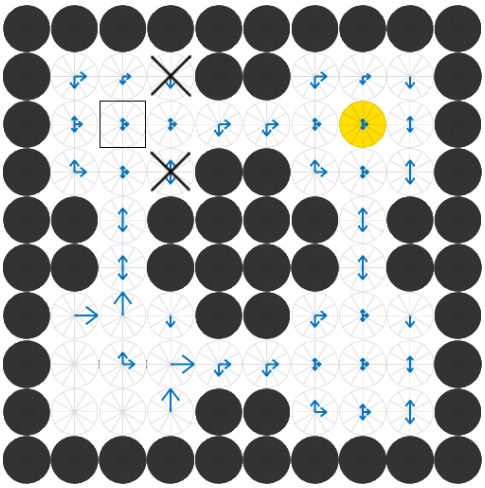
Figure 6.1: Illustration of the problem of reading out the correct action for the current state. Each compass plot represents the firing rate of all SA cells which encode a particular state (see Fig. 6.1b). The golden state is the goal location and the boxed state is the agent's current state. The planning wavefront (ch. 3) has activated a large number of SA cells, most of which encode an action (NW, SW, W) that will not move the agent towards the goal. The model cannot therefore simply sum or average the activity in the SA layer but has to gate this activity by the current state of the agent before it is passed to the action output layer. Furthermore, this read-out must happen at a specific time, when the propagating wavefront has reached the agent's current state but not passed it. (see Fig. 6.2).



(a) t = 3



(b) t = 6



(c) t = 11

Figure 6.2: The Gating Problem. This figure shows compass plots of state-action activity during planning in the 4-room environment over timesteps t = 3, 6 and 11 (taken from Fig. 3.7). If the agent is occupying the state marked by the square, then the model should only read out activity from the SA cells marked by the square, and not (for example) those marked by crosses. At the same time, the agent should not read out activity from the marked SA cells at t = 3 (when the wave of activity has not yet reached the agent's current state) or at t = 11 (when the wave of activity has passed the agent's current state) but only at t = 6 (when the wave of activity has reached the agent's current state).

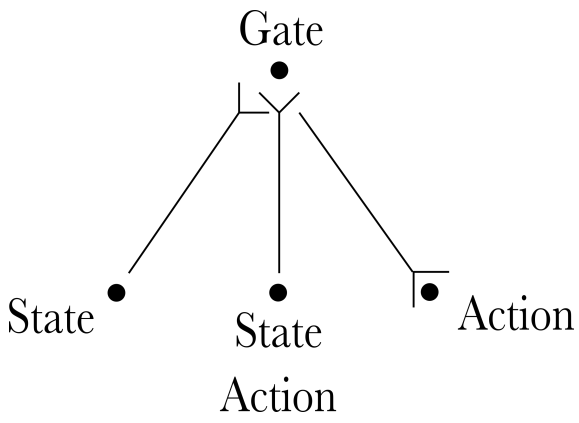


Figure 6.3: Connectivity of a Gating Cell

6.1.4 *Network Model*

The architecture of the network is shown in Fig. 6.4. A layer of state cells encode the agent's current state using a one-hot encoding. A layer of action cells encode the action that the agent is taking or has just taken using a one-hot encoding. A layer of SA cells receive afferent synapses from the state layer and action layer. A layer of gating cells receive afferent synapses form the state layer and the SA layer. Where connectivity exists, it is all-to-all with random weights. The only exception is the recurrent connectivity within the state-action layer, which is all-to-all with zero weights, as described in Ch. 5.

Algorithm 6.1 shows the sequence of events that happens in one timestep. Parameters are given in Table 6.1. We can see that the initial sequence of events (those not in italics) in each timestep is exactly the same as in algorithm 5.1, described in Sec. 5.2 At the end of this part of the process a single SA cell is active, as is a single state cell and a single action cell.

Activity propagates from state cells and SA cells to gating cells, as follows:

$$h_i^G = \sum_j w_{ij}^{S-G} r_j^S + \sum_j w_{ij}^{SA-G} r_j^{SA} \tag{6.1}$$

where h_i^G is the activation of a gating cell i, $\sum_j w_{ij}^{S-G} r_j^S$ is the activity contributed by state inputs and $\sum_j w_{ij}^{SA-G} r_j^{SA}$ is the activity contributed by SA inputs.

Algorithm 6.1 *Full Model with Self-Organisation of Gating Layer.* This describes one timestep, based on alg. 5.1. The steps that have been added in this version are emphasised with italics. All plasticity is Hebbian unless specified otherwise.The full sequence of events listed here occurs in every timestep.

CELL FIRING State Cells (eq. 5.9)

ACTIVITY PROPAGATION State Cells to SA Cells (5.10)

INHIBITION SA Cells: Columnar Winner-Takes-All

SYNAPTIC PLASTICITY State Cells to SA Cells (5.11 & 5.12)

SYNAPTIC PLASTICITY Trace Learning: SA Cells to SA Cells (5.13 & 5.14)

FIRING Action Cells (5.15)

AGENT Agent Moves to Successor State in Grid World

ACTIVITY PROPAGATION State Cells & Action Cells to SA Cells (5.16)

INHIBITION SA Cells: Full Winner-Takes-All (5.17)

SYNAPTIC PLASTICITY State Cells & Action Cells to SA Cells (5.11, 5.12, 5.18 & 5.19)

PLASTICITY Set Memory Trace: SA Cells

activity propagation State Cells & SA Cells to Gating Cells (6.1)

inhibition Gating Cells: Bandstop (6.2)

inhibition Gating Cells: Full Winner-Takes-All (6.3)

synaptic plasticity State Cells & SA Cells to Gating Cells (6.4, 6.5, 6.7 & 6.8)

synaptic plasticity Gating Cells to Action Cells (6.6 & 6.9)

RESET All Cells Reset to Zero Activation

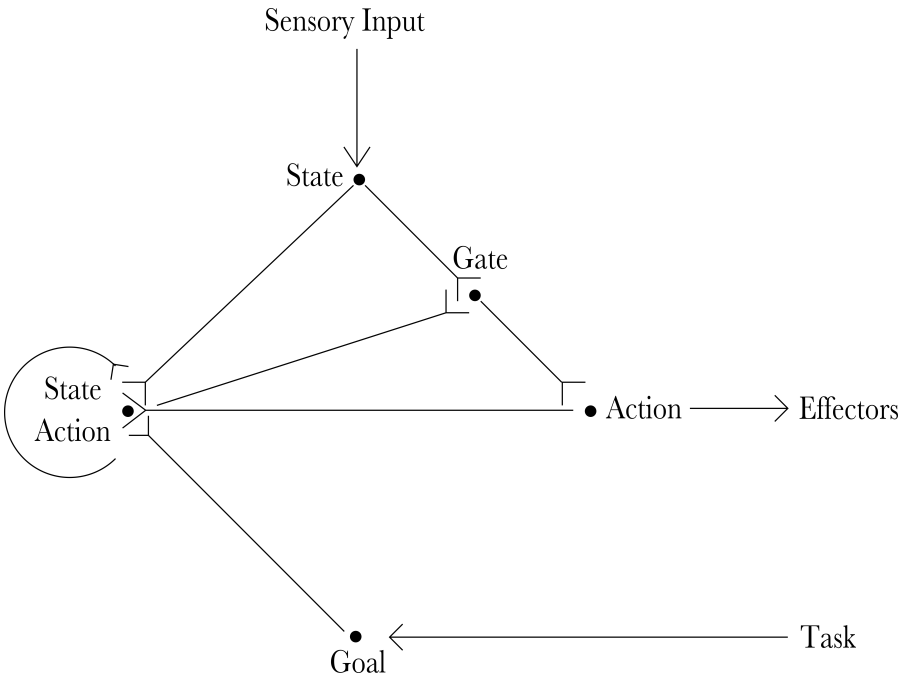


Figure 6.4: Architecture of the proposed model with self-organising gating layer.

We then apply 'bandstop' inhibition to the gating cells. This inhibits its activity within a certain bandstop, inhibiting the mid-level activity typical of a gating cell which only partially represents current sensory-SA input. This prevents one gating cell from successfully competing for all state-SA combinations involving the same state, or all state-SA combinations involving the same SA combination, and so prevents the formation of single gating cells that respond to many state-SA combinations or gating cells that are constantly overwriting themselves. Bandstop inhibition takes place as follows:

$$r_i^G = \begin{cases} h_i^G & h_i^G \leq t_{min}^G \\ 0 & t_{min}^G < h_i^G < t_{max}^G \\ h_i^G & h_i^G \geq t_{max}^G \end{cases} \tag{6.2}$$

where h_i^G is the activation of a cell i in the gating cell layer, r_i^G is the firing rate of that cell and t_{min}^G & t_{max}^G are constants that define the limits of the suppressed bandstop.

After bandstop inhibition has been applied, winner-takes-all competition is applied by setting the firing rate of the most active gating cell to 1 and the firing rates of all other gating cells to zero as follows:

$$r_i^G = \begin{cases} 1 & r_i = \max_i \left\{ r_i^G \right\} \\ 0 & r_i \neq \max_i \left\{ r_i^G \right\} \end{cases} \tag{6.3}$$

where $\max_i \left\{ r_i^G \right\}$ is the activity of the most active cell in the gating layer.

The synaptic weights between the state cells and gating cells are updated as follows:

$$\Delta w_{ij}^{S-G} = k r_i^G r_j^S \tag{6.4}$$

and the synaptic weights between SA cells and gating cells are updated as follows:

$$\Delta w_{ij}^{SA-G} = k^G r_i^G r_j^{SA} \tag{6.5}$$

and the synaptic weights between gating cells and action cells are updated as follows:

$$\Delta w_{ij}^{G-A} = k^G r_i^A r_j^G \tag{6.6}$$

where k is a learning rate constant.

These synapses are then rescaled so that:

$$\sum_i w_{ij}^{S-G} = 0.5 \quad \forall j, \tag{6.7}$$

$$\sum_i w_{ij}^{SA-G} = 0.5 \quad \forall j, \tag{6.8}$$

$$\sum_i w_{ij}^{G-A} = 1 \quad \forall j, \tag{6.9}$$

It should be noted that the magnitude of the change in synaptic weight is enough to take them from below t^{SA}_{min} to above t^{SA}_{max} in one timestep. There is therefore no change of cells becoming 'trapped' in the inhibited area between t^{SA}_{min} and t^{SA}_{max}. Relevant parameters are given in Table 6.1.

Parameter	Value
Gating Cell Upper Threshold t^{G}_{max}	1.5
Gating Cell Lower Threshold t^{G}_{min}	0.5
Gate Learning Rate (k^{G})	100

Table 6.1: Table of Parameters for Section 6.1

6.1.5 *Results*

To test the self-organisation of gating cells we used an information-theoretic measure. This measure tests whether the response of a gating cell can be linked to a specific combination of state and SA inputs. A gating cell has high single-cell information if its activity is sufficient to predict the presence or absence of a particular state cell & SA cell combination; in other words if that gating cell reliably responds to one state cell & SA cell combination and reliably *does not* respond to any other state-SA combination.

To generate this measure, we generate every combination of state cell and SA cell that could be encountered in the environment on which the model was trained (every state that was not covered by a wall, combined with every cell in the SA layer). For each combination, the appropriate state cell and SA cell are activated, and the activity propagates to the gating cells as in eq. 6.1. We then thresholded the gating cells as follows:

$$r^{G}_{i} = \begin{cases} h^{G}_{i} & h^{G}_{i} > t^{G} \\ 0 & h^{G}_{i} \leq t^{G} \end{cases} \tag{6.10}$$

where t^G is a thresholding constant. We use thresholding (eq. 6.10) than the bandstop and WTA inhibition described earlier in this chapter (eqs. 6.2 and 6.3) because this is the form of inhibition that gating cells experience during the planning process (eq. 3.11).

Gating cells that were still firing at a high rate after inhibition were recorded. The responses of these cells were used to calculate the amount of information that each gating cell's response give about the stimulus. This calculation was as follows:

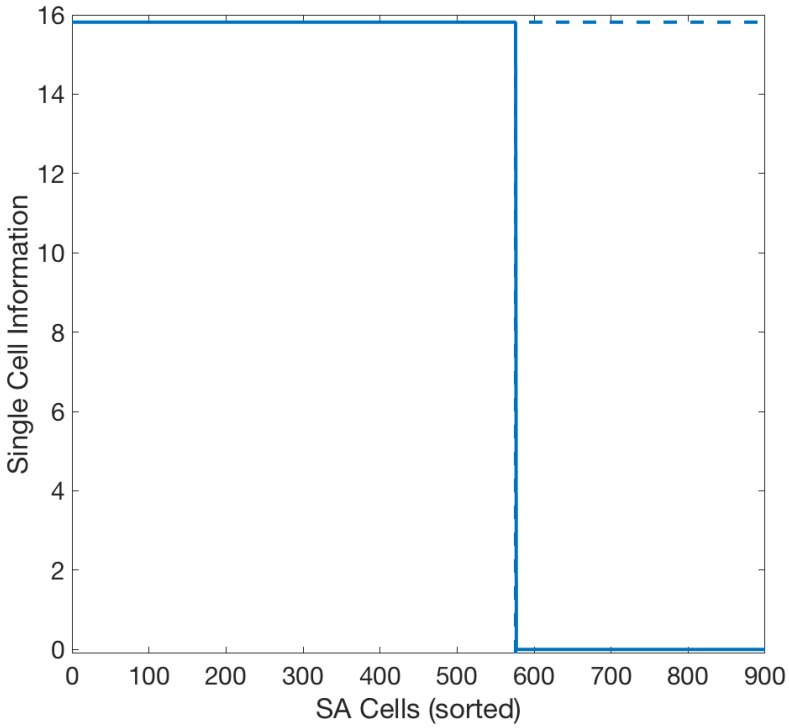
$$I(s, \vec{R}) = \sum_{r \in \vec{R}} P(r|s) \log_2 \frac{P(r|s)}{P(r)} \tag{6.11}$$

where s is a particular stimulus (a particular state-SA combination) and $\vec{R}$ is the set of recorded responses of the gating cell. The maximum possible information for a gating cell is equal to:

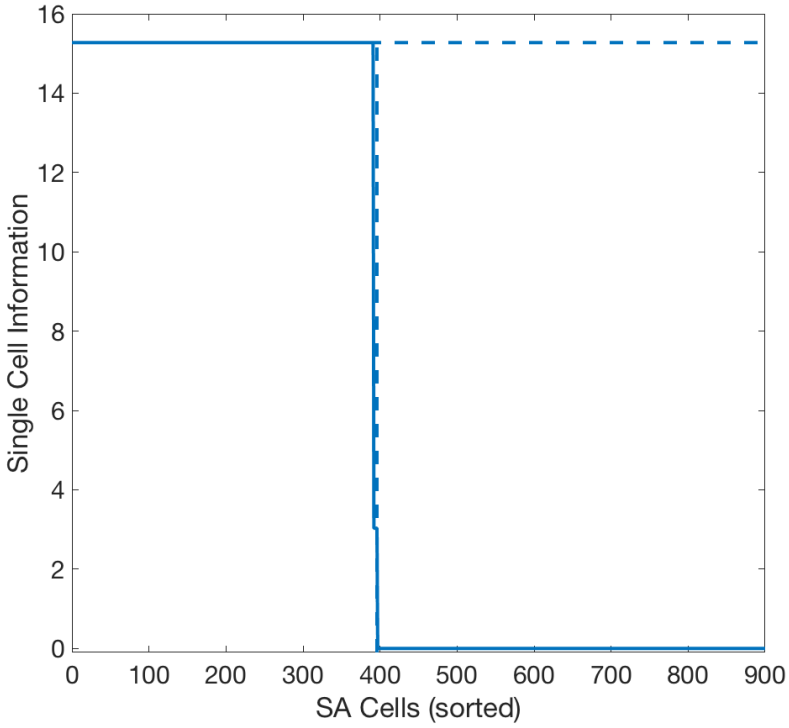
$$\log_2 (N_{ST} N_{SA}) \tag{6.12}$$

where N_{ST} is the number of free states in the environment (see Fig. 3.6) and N_{SA} is the number of cells in the SA layer.

Figure 6.5 shows that the model was able to self-organise gating cells with the maximum possible information in both an open and a 4-room environment, and was able to do so for every state-SA combination that needed to be encoded. The maximum possible information for a gating cell is calculated according to eqn. 6.12: $\log_2 \{64 * 900\} = 15.81$ for the open environment and $\log_2 \{44 * 900\} = 15.27$ for the 4-room environment. If a gating cell encodes the maximum possible information, this means that it has learned to respond uniquely to a single combination of state and SA cells. Figure 6.5 shows that the maximum possible number of gating cells are able to learn the maximum possible information in both open and more complex 4-room environments.

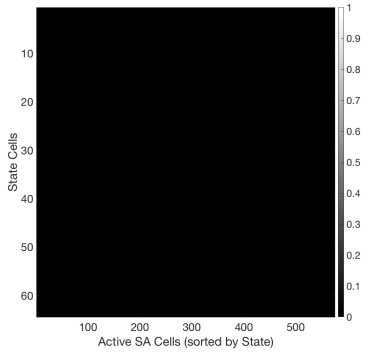


(a) Open Environment (8x8 World). $N_{ST} = 64$. $N_{ST}N_A = 576$.

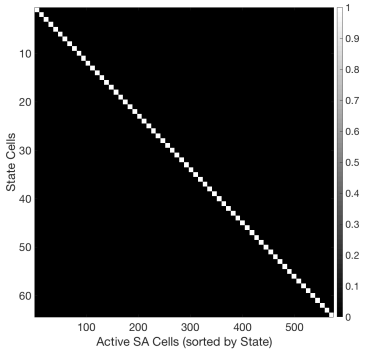


(b) 4 Room Environment (8x8 World). $N_{ST} = 44$. $N_{ST}N_A = 396$.

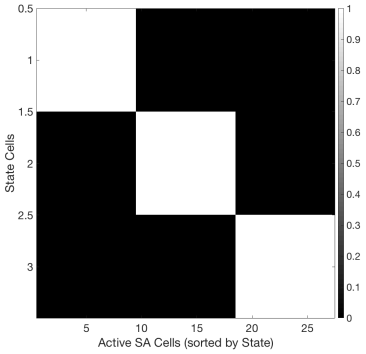
Figure 6.5: Single cell information analysis of gating cells after learning. A horizontal dashed line indicates the maximum possible information that a gating cell can encode. This is calculated as $\log_2(N_{ST}N_{SA})$, where N_{ST} is the number of available states and N_{SA} is the total number of SA cells. A vertical dashed line indicates the maximum number of gating cells that can self-organise; this is effectively equal to $N_{ST}N_A$ where N_A is the number of available actions (9). We see that in both the five-action experiment and the nine-action experiment, the maximum number of gating cells that can self-organise have learned to encode the maximum possible single-cell information.



(a) The number of gating cells that fire with with $r_i^G > 0.90$ for a single combination of state and SA cells, before training in a 10x10 open environment. Because there are only 576 state-action combinations available in the environment, only 576 SA cells will become active over the course of exploration, and so the x-axis has a length of 576.



(b) The number of gating cells that fire with with $r_i^G > 0.90$ for a single combination of state and SA cells, after training in a 10x10 open environment. Because there are only 576 state-action combinations available in the environment, only 576 SA cells will become active over the course of exploration. We have sorted the x-axis according to the SA cells' associated state (see main figure legend).



(c) As Fig. 6.6b, zoomed.

Figure 6.6: The number of gating cells that encode each state-SA combination before and after training. Fig. 6.6a shows that no gating cell fires uniquely for any combination of state and SA cells before the exploration period. Fig. 6.6b shows that after training, gating cells fire uniquely for certain combinations of state and SA cells. Unlike the equivalent figures (4.6, 4.13 and 4.17) in Ch. 4, which show that at least one SA cell responds uniquely to every state-action combination after training, there is not a unique gating cell for every state and SA combination. This is because an SA cell (which encodes a unique combination of state and action) will only fire in conjunction with its associated state during exploration. Most state and SA cell combinations are therefore invalid: the cells in the gating layer will never experience this combination of inputs. We have therefore sorted the x-axis so that the valid state and SA cell combinations, which the agent can actually experience, lie along a diagonal, and we see that one gating cell has learned to respond to each of these valid combinations. Fig. 6.6c zooms in on this diagonal (from Fig. 6.6b) and shows that there are in fact several valid state-SA combinations for each state cell. This is because a number of SA cells exist for each state, each representing a different action in that state. We see that a gating cell fires uniquely for all of the state-SA combinations available for each state.

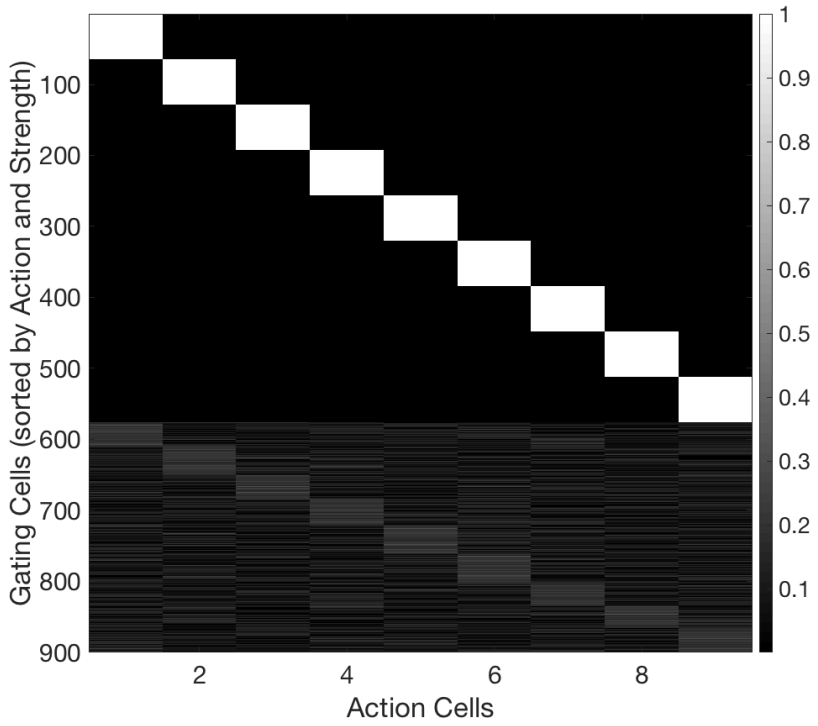


Figure 6.7: Efferent synaptic connectivity of gating cells after self-organisation in an 8x8 open environment using the method described in Section 6.1. The x-axis shows the postsynaptic action cells, and the y-axis shows the presynaptic gating cells. For clarity, the y-axis has been sorted so that cells with a strong efferent connection to a particular action cell are grouped together. We see that the majority of gating cells (specifically $N_{ST}N_A = 576$) form a strong synapse to one and only one action cell. This is what we expect: as in Fig. 6.5a the maximum number of gating cells that can self-organise is 576, and these cells have also formed strong efferent connections to action cells, allowing them to pass on activity to the action cells and so produce action output as in Ch. 3.

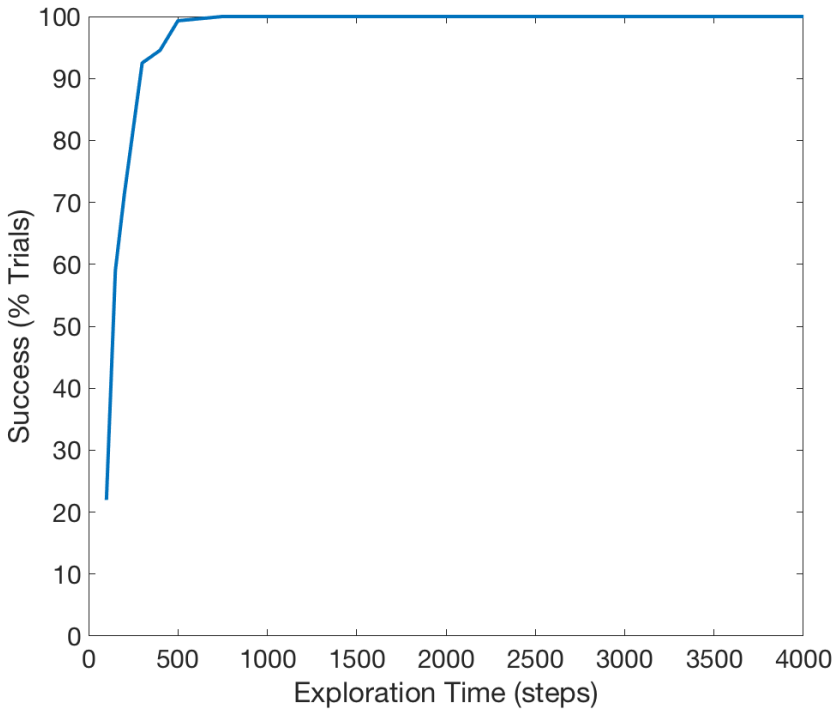


Figure 6.8: The percentage of successful trials if the model explores for a certain number of timesteps and is then used (instead of a hard-wired model) to perform planning tasks identical to those in Section 3.1.

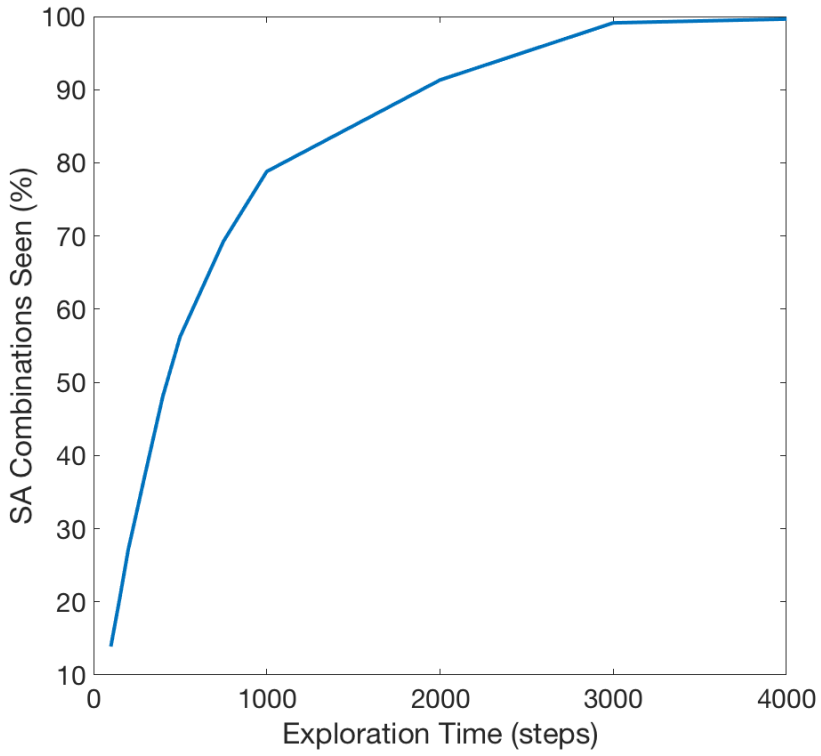


Figure 6.9: The percentage of SA combinations that the model has experienced after a certain number of timesteps. The model explores randomly, and so as its knowledge of the map grows more comprehensive, it becomes less likely to experience unknown SA combinations. The amount of environmental knowledge that the model contains therefore increases very quickly during early exploration but it takes many timesteps for the model's knowledge of the environment to become comprehensive.

Figure 6.6 shows that these gating cells covered every valid[1] combination of state cell and SA cell; in other words, that at least one gating cell fires uniquely for each valid combination of state and SA cell, and therefore that the gating layer is able to pass output from any SA cell to the action layer provided that they have the appropriate efferent connectivity to the action cells. Figure 6.7 indicates that this efferent connectivity has also self-organised; it shows that the appropriate number of gating cells have formed strong connections to one and only one action cell, allowing them to pass on activity to that cell and so produce action output as in Ch. 3.

Figure 6.8 then explores the learned model's ability to plan (according to the mechanisms in Sec. 3.1). This figure shows the percentage of correct navigation trials (navigating from one random state to a random goal state) that this model is able to achieve after exploring for a given number of timesteps. During exploration, the model uses the learning mechanisms described in this chapter; during planning the model uses the planning mechanism described in Sec. 3.1. We see that the percentage of successful planning trials rises very quickly, even with relatively little exploration time. Figure 6.9 makes this clearer by showing the number of state-action combinations that the model experiences during learning. By comparing the percentage success rates in Fig. 6.8 with the percentage of SA combinations experienced in Fig. 6.9, we can see that the model's ability to plan grows much faster than its experience of the environment. We also see that the model reaches 100% planning success long before it has experienced all SA combinations in the environment. In fact, only about 65% knowledge of the environment seems to be required for the model to navigate perfectly.

1 Unlike the equivalent figures (4.6, 4.13 and 4.17) in Ch. 4, which show that at least one SA cell responds uniquely to every state-action combination after training, there is not a unique gating cell for every state and SA combination. This is because an SA cell (which encodes a unique combination of state and action) will only fire in conjunction with its associated state during exploration. Most state and SA cell combinations are therefore invalid: the cells in the gating layer will never experience this combination of inputs.

6.1.6 *Discussion*

In this chapter we hypothesised that the similarity between gating cells and state-action cells meant that the mechanisms which self-organise SA cells could also self-organise gating cells. As described in this chapter, the purpose of these cells is to pass activity from the SA cells to the action cells if and only if the SA activity is related to the agent's current state. A gating cell should therefore receive activity from one SA cell and the state cell (or combination of state cells) that encode that SA cell's preferred state (see Fig. 3.5), and should be able to propagate activity to that SA cell's preferred action. Gating cells are under heavy inhibition such that they can only produce firing if they are receiving both SA and state input (see eqn. 3.11), meaning that they only pass on activity from an SA cell when that SA cell matches the current state.

The results of this experiment demonstrate that the procedure described in this chapter can fully self-organise the required connectivity between state cells, SA cells and gating cells (Fig. 6.5). We also show that the resultant cells can be used to successfully plan solutions to grid world tasks (6.8). The number of gating cells improves relatively slowly but the ability to plan improves very quickly, demonstrating that relatively low SA and gating coverage is necessary for successful planning.

This chapter is the last of three experimental chapters that are each devoted to methods of self-organising an aspect of the model described in Ch. 3 as biologically plausibly as possible. Chapter 4 investigated mechanisms for self-organising the afferent connectivity to SA cells. Chapter 5 investigated how the recurrent synapses between SA cells could learn to encode the causal transition structure of the environment in terms of states and actions. And finally this chapter investigated mechanisms for self-organising the afferent and efferent connectivity of the gating cells. In each chapter we have added the new mechanisms to the previous procedure, so that this chapter finally shows how each mechanism (SA self-organisation, SA map learning, and gating cell self-organisation) comes together to

self-organise the entire network used to perform planning tasks in Ch. 3.

There remain certain biological implausibilities in this series of models. In particular, each of the three self-organisation chapters has required a longer list of activity propagation, inhibition and synaptic weight update events to happen in a precise order (algorithm 6.1). There is also the issue that the process described in algorithm 6.1 relies heavily on bandstop inhibition, and although this mechanism resembles the results of inhibitory plasticity (see section **??**). Work by Vogels et al. 2011 shows that as a (simulated) neuron receives excitatory stimulation, it originally fires at a high rate. Ove time, inhibitory plasticity can produce a situation where the excitation and inhibition that the neuron receives is balanced. If this occurs, a preferred stimulus produces strong excitatory input, beating the inhibitory input and therefore producing postsynaptic firing. However, less-preferred stimuli produce very low firing, because they cannot overcome the inhibition present [97]. Such a mechanism seems likely to produce results similar to a bandstop filter. However, bandstop inhibition is not a frequently used mechanism in modelling of this type and we cannot guarantee that it occurs in the brain.

7

REPLICATION OF A DETOUR TASK

Having shown that the model can perform grid world tasks, we now demonstrate that it can replicate the characteristic performance of rats running a detour maze task. This task is still considered one of the key tasks demonstrating that cognitive maps exist [11, 48]; work in humans strongly suggests an ability to solve detour tasks without re-learning, and these results have not yet been replicated by model-free or successor-representation based models [11, 12].

The detour task was originally performed by Tolman and Honzig in the 1930's, was re-performed by Alvernhe 2011 [21] and used as a measure of model performance by Martinet 2011 [2]. Each version of this task has minor differences; this section replicates the version described by Martinet 2011 because it allows the fairest and most direct comparison between two neural network models of this task: the one proposed in this thesis and Martinet 2011. We see it as a useful litmus test of the model's basic ability to replicate map-based planning.

Section 7.1 of this chapter will investigate the model's ability to perform model-based planning in tasks where the environment is dynamic and obstacles change position during the course of the task (as in the detour task). This section will describe mechanisms that allow the model to perform these tasks successfully.

Section 7.2 will investigate the model's ability to perform model-based planning tasks when transitions between states are probabilistic, in other words when the model is uncertain about the outcome of performing a given action in a given state. This section will describe a mechanism that allows the model to successfully plan under these circumstances.

Section 7.3 will describe the detour task conceptually.

Section 7.4 will describe the simulation procedure used in Martinet et al. 2011 [2] and in this thesis to simulate the detour task.

Section 7.5 will present the results of this experiment, demonstrating that the model is able to replicate the characteristic model-based behaviour described by Tolman 1948 and Alvernhe 2011 [21].

Section 7.6 will discuss these results.

7.1 PLANNING WITH MOVING OBSTACLES

7.1.1 *Statement of Problem*

Studying how agents replan and re-evaluate when the transition structure of their environment changes is one of the key elements of the detour task. An agent that plans using a cognitive map cannot expect that map to remain perfectly accurate over time. The position of various obstacles in the world is likely to change, as is the way that agents can interact with certain obstacles. This section therefore asks: How can the agent update its cognitive map so that it is able to incorporate unexpected obstacles into its cognitive map during planning?

7.1.2 *Hypothesis*

The simplest solution to this problem is a 'nociceptive' signal or a 'suprise' signal. The former warns the agent that its actions have produced an adverse response (in this case, hitting a wall). The latter would be more extensive and tell the agent that the state transition predicted by the agent's cognitive map has not occurred, thus encompassing positive, negative and neutral changes in the environment. Dopamine may be suitable for encoding this surprise signal.

If the environment has changed, then the agent must be able to update its internal cognitive map in order to continue planning well. This process is made more difficult by the two-mode nature of the model, which must either be in the learning mode or the planning

mode, with the appropriate activity dynamics, plasticity and inhibition for that mode.

The easiest way to alter the map is to hardwire the appropriate synapses. This is a strategy seen in Martinet 2011 [2]. In this paper the authors halve the weight value of any synapse encoding an unsuccessful state transition within the cognitive map.

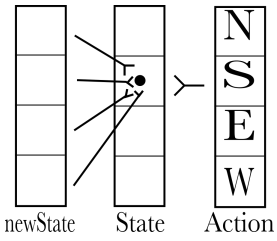
A more biologically realistic way is for the 'surprise' signal to switch the model into the 'learning' mode. The model then learns the new state transition. In the case of an unexpected obstacle, this means that the model learns that the SA combination which had formerly led to a different state now leads to the agent remaining in its current state. This effectively creates an SA cell whose efferent projections loop back to itself, rather than allowing activity to spread to another SA cell representing a different state. The former state transition has effectively been snipped out of the cognitive map (see Fig. 7.1).

Relearning is more complicated when an obstacle is removed from the environment, as the agent has no reason to attempt a state transition which it believes (according to its now-outdated cognitive map) will have no useful effect. We hypothesise that this can be solved by stochastic learning: if the model takes more than a set time to finish a task or move away from the current area, it will switch into the learning mode for a brief period.

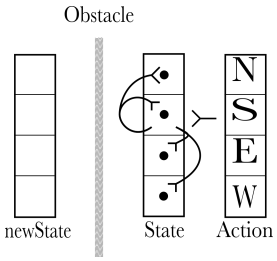
7.1.3 *Task*

The agent begins with a hardwired cognitive map encoding an 8 by 8 state open environment (Fig. 7.2a). The agent is placed at the top left of the environment and the goal is placed at the top right. A wall is added to this environment that bisects it, blocking the path that would have been optimal in the open environment and leaving only a small gap through which the agent can pass (fig 7.2b).

If the agent is able to reach the goal despite the unexpected blockage, then the agent is considered to have succeeded at the task. If the agent fails to move to the goal location after 1000 timesteps, the agent is considered to have failed the task. The model is reset to the original



(a) Before Re-Learning



(b) After Re-Learning

Figure 7.1: *Desired Effect of Re-Learning on SA Connectivity*. The transition structure of the environment has changed such that a state-action combination SA which originally moved the agent from one state into a successor state now does not, leaving the agent in its original state.

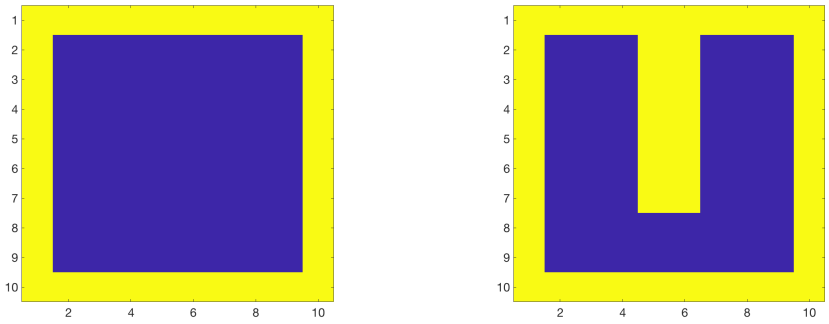
hardwired open field map after each trial, so that it has to relearn the novel obstacle each time.

7.1.4 *Network Model*

The proposed model's architecture is depicted in Figure 7.3 and contains:

- State cells
- Goal cells
- State-action (SA) cells
- Gating cells
- Action cells

During this thesis, we have shown the same network (in terms of cell types and connectivity) operating in two fundamentally different ways. Chapter 3 describes how a hardwired version of the model can plan by injecting activity in the SA layer at a location of a goal state



(a) Open Environment (b) Open Environment with Obstacle

Figure 7.2: The agent can move between blue squares, which are free states, and cannot move into yellow states, which are walls. Each state has a unique index value which is used to fire a specific state cell. The agent is positioned at the top left of the environment and the goal is placed at the top right. (a) shows the environment that the model initally explores to self-organise its cells responses and environment representation. (b) shows the environment in which the model then performs a planning task. The model therefore initially encodes an open environment (a) but encounters an unexpected obstacle during the planning task (b).

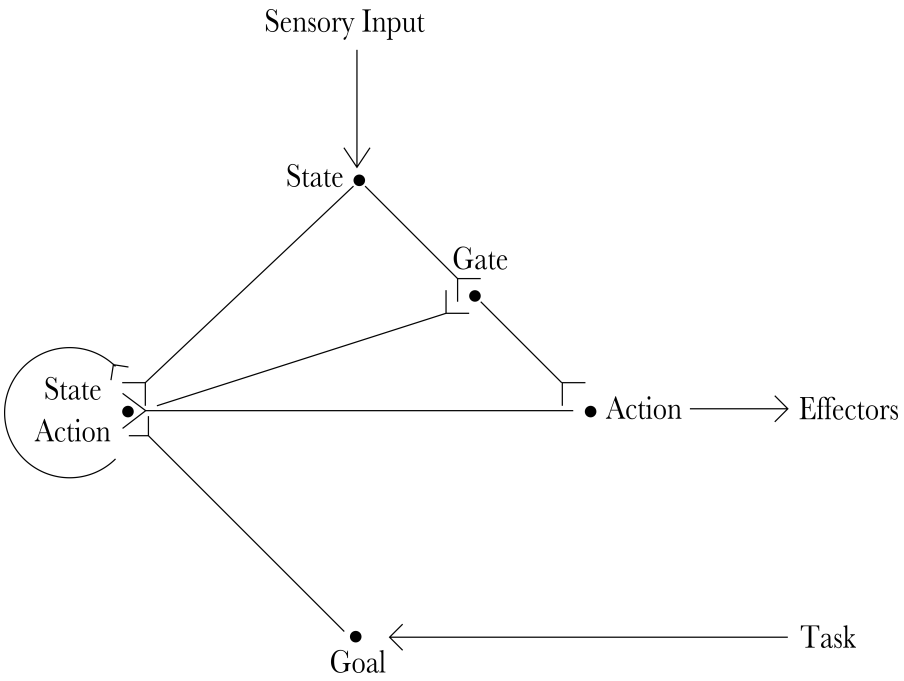


Figure 7.3: Architecture of the proposed network model to be used in simulations with dynamically changing or probabilistic environments.

and allowing that activity to propagate through the network until eventually an action is selected and propagated through the gating layer to the action cells to move the agent. Ch. 6 describes how the model can self-organise its synaptic connectivity while exploring its environment using random actions, self-organise the properties of the SA and gating cells, and learn to encode the state transitions that it experiences in the recurrent connectivity between SA cells. Both of these operations—learning and planning—are currently mutually exclusive. These two modes of operation require different patterns of activation and different forms of inhibition to operate. Therefore the model may be learning or planning during any given timestep but it cannot do both simultaneously. We will refer to these two patterns of network operations as "learning mode" (fully described by Ch. 6, and in particular algorithm 6.1) and "planning mode" (described by Ch. 3 and algorithm 3.1).

Simulation procedures A and B describe two different ways that the model's recurrent connectivity can be altered to describe a new obstacle. In simulation procedure A, we alter the synaptic weights algorithmically (as in Martinet 2011 [2]). In simuation procedure B, the model temporily switches from planning to learning mode when it encounters a failed transition, allowing the model to relearn the appropriate recurrent connectivity for that transition. Sections 7.1.4.1 and 7.1.4.2 describe these procedures in more detail, and Sections 7.1.5 and 7.1.6 show the effects experimentally.

7.1.4.1 *Simulation Procedure A - Algorithmic rewiring of recurrent synaptic weights between SA cells at unexpected obstacles*

In the simplest formulation of the agent, the model begins with a hardwired cognitive map of the environment, which is an 8x8 open field. The environment is bisected as in the task description, and the agent and goal are placed on opposite sides of the wall. At each timestep, the goal cells fire and the activation drives the propagation of a wavefront through the SA layer as described in the planning mode description above. This allows the model to choose optimal or near-optimal actions until it reaches the unexpected wall.

When the model takes an action which causes the agent to hit the wall, the SA cells encoding the intended state (which the agent expects to reach using its incorrect cognitive map) and the SA cell denoting the agent's current location and action are identified algorithmically. Any synapses between these SA cells then have their weights reduced to 0.

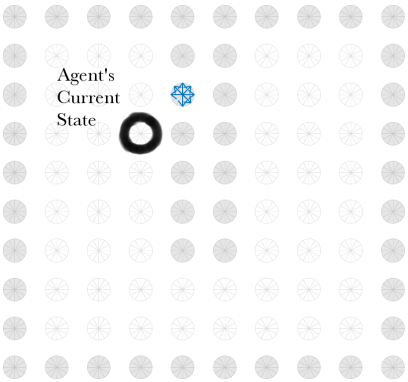
7.1.4.2 *Simulation Procedure B - Hebbian re-learning of recurrent synaptic weights between SA cells at unexpected obstacles*

Simulation B uses the same hardwired map and bisected environment as Simulation Procedure A. When the model takes an action which causes the agent to hit a wall, the model receives a 'surprise' signal that resets the activity of all cells and switches the model into 'learning mode'. The model then acts exactly as it does in Ch. 6 (see Algorithm 6.1), except that instead of moving randomly it replays the last state-action combination. This is intended to retrain the synaptic connectivity within the cognitive map. The model then returns to 'planning mode' and continues to perform the planning task.

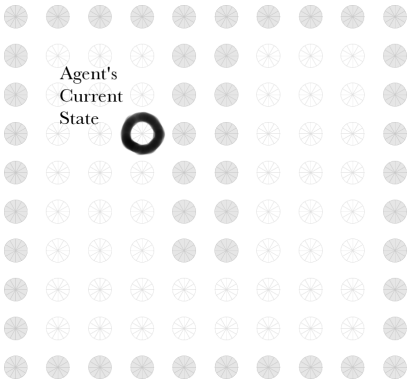
7.1.5 *Results using simulation procedure A*

Figure 7.4 shows that if the model is trained on an 8x8 open environment, but is then required to circumvent an unexpected obstacle (a wall that partially bisects the environment), the model will initially predict that the agent can move straight to the goal (Fig. 7.4a). By reducing the relevant synapses algorithmically as the agent attempts various transitions and hits the wall, we can lower the associated estimated transition probabilities accordingly, or remove them altogether.

Figure 7.6 demonstrates that if the agent encounters an unexpected obstacle, and the relevant synapses are reduced algorithmically to 0, the model can still succeed in the task with 100% probability.



(a) The recurrent synapses between SA cells encode a backwards causal model of the environment. Each SA cell therefore receives afferent synapses from all SA cells in that SA cell's *predicted successor state*. This figure shows the afferent synapses for an SA cell representing a movement northeast from the marked state (into the new wall) before alteration. We see that the SA cell receives synapses from the state immediately northeast of it, inside the new wall (marked in light grey). This connectivity (see Fig. 7.5 for reference) encodes the (now false) knowledge that movement northeast from the marked state will move the agent northeast as intended. In reality this movement will fail, leaving the agent in the marked state.



(b) Having altered the relevant synapses to 0, the same cell as that marked in (a) now receives no afferent synaptic weight from SA cells representing the predicted successor state. This signifies that the relevant transition probability is now assumed to be zero. In other words, after relearning using Simulation Procedure A, the afferent connectivity of the SA cell now correctly predicts that movement northeast from the agent's current position will fail, returning the agent to its current state.

Figure 7.4: *Obstacle avoidance through manually altering synapses*. This figure shows the effect of simulation procedure A on the connectivity of an SA cell encoding a movement northeast from the marked state. Initially, the afferent connectivity of this cell predicts that movement northeast from the marked state will shift the agent to the northeastern neighbouring state (Fig. 7.4a). However, a wall has now been added that blocks this movement. After attempting to move northeast from the marked state and failing to do so, the value of the synapses encoding this prediction are set to 0, signifying that this transition is now expected to fail. Under the effect of propabilistic propagation (see next section) a low synaptic weight has the effect of making it much less likely that the agent will choose this state-action combination during planning. In other words, by altering the synaptic weights encoding a given transition, we can increase or decrease the associated transition probability, or remove it altogether. By altering the agent's cognitive map in this way, we allow it to circumvent the obstacle and reliably reach the goal.

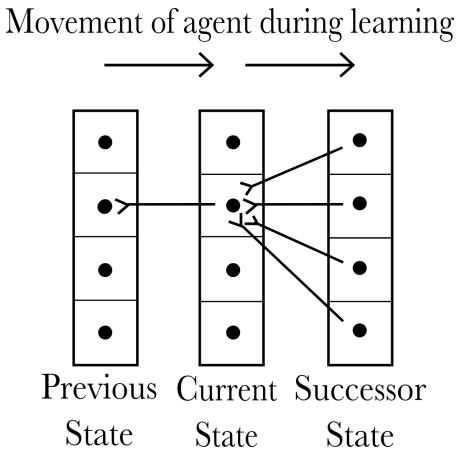


Figure 7.5: A diagram of the expected pattern of SA connectivity (for reference). The recurrent connectivity in the SA layer supports a backwards causal model, as described in Ch. 5. Each SA cell receives a set of afferent synapses from all of the SA cells encoding its predicted successor state, allowing cells in a successor state to activate the SA cell responsible for that state. Likewise, each SA cell sends an efferent synapse to any SA cells that will result in the SA cell's own encoded state.

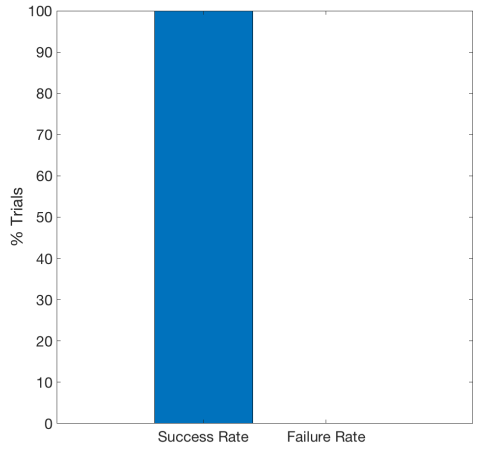


Figure 7.6: % Trial success in 100 Timesteps if the cognitive map encoded by recurrent connectivity between SA cells is algorithmically altered according to Simulation Procedure A when agent encounters an unexpected obstacle

7.1.6 *Results using simulation procedure B*

Figure 7.7 shows that the model can successfully relearn causal connections using the same Hebbian mechanisms described in Chapters 4, 5 and 6. Specifically, it shows that if the model is trained on an 8x8 open environment, but required to circumvent an unexpected obstacle (a wall that partially bisects the environment), the model will initially predict that the agent can move straight to the goal (Fig. 7.7a). As the agent attempts to do so, hitting the wall, it is informed of failed transitions via a 'surprise' signal. It therefore replays that transition in 'learning' mode and updates the recurrent connectivity between SA cells accordingly (Fig. 7.7b). The introduction of this mechanism allowed the model to achieve a high percentage of success in the navigation task (Fig. 7.8).

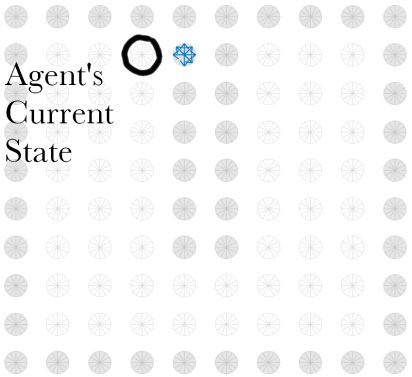
7.2 PLANNING WITH PROBABILISTIC PROPAGATION

7.2.1 *Statement of Problem*

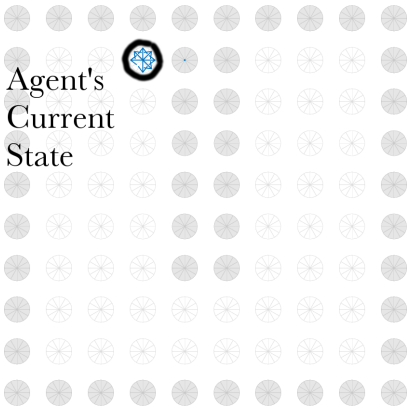
In the simulations described previously in this thesis, the propagation of activation has been considered to be essentially deterministic. Although some simulations in earlier chapters (see section 3.2) have investigated the problem of planning when propagation is noisy, those simulations essentially investigated the problem of planning reliably *despite* noise. The predictable nature of activity propagation in these simulations means that the model tends to produce very similar paths if it re-encounters the same problem (the same agent and goal positions). This does not match the results from rats performing a detour task: Fig. 7.14a shows that although rats tend to take the optimal path to a goal, they will take an alternative path in some trials. This suggested that our model's planning mechanism was overly deterministic.

Furthermore, although most of the alternative models described in Ch. 2 do not consider transition probabilities,[1] those models that *do* consider them are based on the principle of planning through decay-

1 Assuming that all transitions are deterministic in nature, not probabilistic.



(a) The recurrent synapses between SA cells encode a backwards causal model of the environment. Each SA cell therefore receives afferent synapses from all SA cells in that SA cell's *predicted successor state.* This figure shows the afferent synapses for an SA cell representing a movement east from the marked state (into the new wall) before relearning. We see that the SA cell receives synapses from the state immediately east of it, inside the new wall (marked in light grey). This connectivity (see Fig. 7.5 for reference) encodes the (now false) knowledge that movement east from the marked state will move the agent east as intended. In reality this movement will fail, leaving the agent in the marked state.



(b) We see that, after relearning, the same cell as that marked in (a) receives afferent synapses from SA cells representing the marked state. This signifies that the SA cell's predicted successor state is now the marked state, rather than the state to its east. In other words, after relearning using Simulation Procedure B, the afferent connectivity of the SA cell now correctly predicts that movement east from the agent's current position will fail, returning the agent to its current state.

Figure 7.7: *Obstacle avoidance through relearning*. This figure shows the effect of simulation procedure B on the connectivity of an SA cell encoding a movement east from the marked state. Initially, the afferent connectivity of this cell predicts that movement east from the marked state will shift the agent to the eastern neighbouring state (Fig. 7.7a). However, a wall has now been added that blocks this movement. After attempting to move east from the marked state and failing to do so, the model replays the failed movement in 'learning mode' as per Simulation Procedure B, and we see that the connectivity of the model now encodes the correctprediction. The model is therefore capable of encoding unexpected obstacles using a biologically plausible Hebbian learning procedure (Fig. 7.7b), allowingthe agent to reach its goal by circumventing such obstacles.

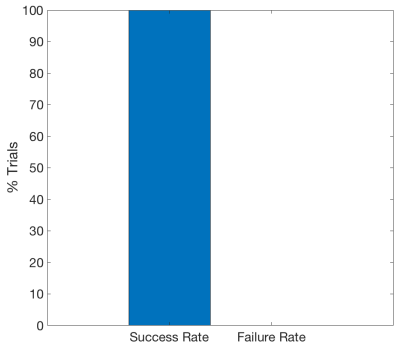


Figure 7.8: Success rates using relearning of recurrent connections between SA cells according to Simulation Procedure B during obstacle task

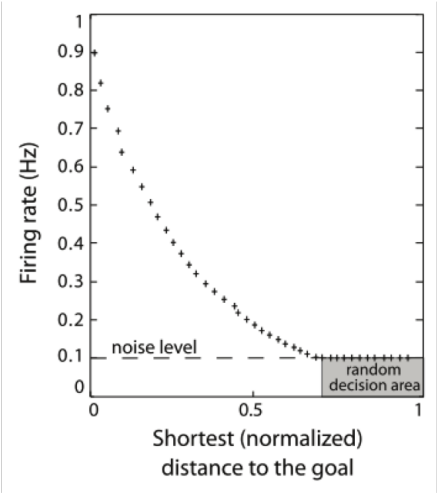


Figure 7.9: This figure shows that, under the decaying activity paradigm, cells further from the goal have reduced firing rate. More importantly, it shows that the firing rates of cells far from the goal are so low that their firing rate cannot be realistically read out by a neural mechanism. Use of the decaying activation paradigm therefore limits the maximum viable path lengths in Martinet 2011, and by extension in other models that use the decaying activation paradigm. This figure is reproduced from Martinet et al. 2011 [2] in accordance with the Creative Commons Attribution (CC BY) licence.

ing activation (see Fig. 7.9). In the paradigm of decaying activation, the passage of activation outwards from the goal produces the planning process. Activation decays as it propagates and so an optimal state-action or equivalent cell with receive a larger amount of activation compared to a less optimal cell, because the optimal state-action cell is connected to the source of activation (the goal) by fewer transitions and so receives activation that has decayed less. In this way the activation in these model becomes a crude approximation of the value-function that would be calculated by a model-based algorithmic planner (explicitly so in Friedrich et al. 2016 [83]). Decaying-activation models can account for transition probabilities quite easily by encoding them in the recurrent weights between SA cells (or their equivalents). Lowering the synaptic weight between two cells encodes that the corresponding transition is low probability and heavily reduces the amount of activity that propagates through this synapse. Essentially, low-probability transitions impart extra decay to activity passing through them, signalling their undesirability.

However, in the model that we have proposed in this thesis (see Ch. 3), the wave-propagation planning mechanism relies exclusively on the *timing* of goal-based activity propagation through cells encoding the cognitive map, rather than how much activation those cells receive. The *first* SA cell to receive activation in each state represents the optimal action to take in each state, and the level of activation is not relevant. This planning mechanism has two primary advantages: it allows planning to become faster, and it removes the requirement for goal-based activation to decay, allowing the model to hypothetically cope with problems of much greater scale. It has an important disadvantage in that—because the precise activation of SA cells becomes irrelevant—the model cannot indicate transition probabilities by reducing the level of propagated activation to indicate a reduced transition probability. Increasing or decreasing the level of activation that an SA cell receives will not change the output of the proposed model, because it does not change whether that SA cell was the first cell to receive activation.

7.2.2 *Hypothesis*

We hypothesise that a timing-based planning model must instead indicate transition probabilities by altering the timing of activity propagation, and that this can be done either by delaying propagation or by making the probability of propagation dependent on the strength of the synapse, which in turn roughly indicates the transition probability (see previous section). Although this is a primarily practical decision, the nature of synaptic transmission makes this possible, even likely, especially if we expect planning to take place over a relatively short period of time.

This is because synaptic transmission is naturally unreliable [96]. Specifically, the spike of a presynaptic neuron does not reliably produce vesicle release. The probability of vesicle release (which depends on both the number of sites and the probability of release at individual sites) is affected by synaptic plasticity and makes up part of what is generally called synaptic weight [118]. The weaker the synaptic weight, the less likely that an action potential is to produce a postsynaptic spike at any given time. Furthermore, although cells connected by many tens of synaptic contacts show reliable responses to similar spike trains, cells that are connected by fewer synaptic contacts show much more variable responses because they are more reliant on the behaviour of individual synapses.

Modelling this process is not entirely straightforward as the stochastic propagation of this sort occurs on a short timescale whereas rate-coded models usually describe a longer period of time, averaging out individual spikes to produce activation values. However, we expect the planning process to take place over a relatively short timescale in order to produce useful behaviour (we explore the issue of planning speed at more length in Chapters 8 and 10) and so we hypothesise that planning may occur using a stochastic propagation mechanism like the one we describe.

7.2.3 *Task*

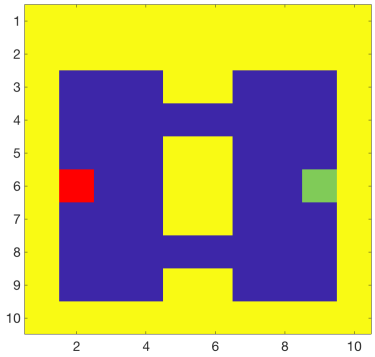


Figure 7.10: Two Gate Environment (8x8). As in Fig. 7.2, the agent can move between blue squares, which are free states, and cannot move into yellow states, which are walls. The green square represents the goal, and the red square represents the agent's starting state. The agent therefore starts equidistant from both gates, and can reach the goal from either gate in the same number of timesteps.

An 8x8 grid world is bisected by a wall into two 'rooms'. There are two points at which the agent may pass through the wall (Fig. 7.10) called 'gates'. The agent is placed in one room, equidistant from both gates, and the goal is placed in the other room, likewise equidistant. The number of trials in which the agent passed through each 'gate' is recorded. The model may perform the task in one of two conditions.

NORMAL The model has learned the environment using the mechanisms described in the previous chapter, and performs the task 'as is'. The recurrent SA to SA synapses that indicate the existence of transitions in the cognitive map are equal in weight, and all other recurrent synapses remain at zero.

REDUCED The model is the same as in the 'Normal' condition but the synapse representing the transition through the lower gate has been reduced to 10% of its original value.

7.2.4 *Method*

The network carries out planning as described in Ch. 3, with the single exception that the mechanism for propagating activity through the network has changed. In Ch. 3, SA cells are activated as follows:

$$h_i^{SA} = \sum_j w_{ij}^{GL-SA} r_j^{GL} + \sum_j w_{ij}^{SA-SA} r_j^{SA} \tag{7.1}$$

where $\sum_j w_{ij}^{GL-SA} r_j^{GL}$ is the summed input received from the goal cells and $\sum_j w_{ij}^{SA-SA} r_j^{SA}$ is the summed recurrent SA input.

In this experiment, in which we introduce probabilistic planning, SA activation is as follows:

$$h_i^{SA} = \sum_j p_{ij}^{GL-SA} r_j^{GL} + \sum_j p_{ij}^{SA-SA} r_j^{SA} \tag{7.2}$$

where p_{ij}^{GL-SA} is stochastic, being either 1 or 0 depending on the value of w_{ij} as follows:

$$\begin{aligned} P\left(p_{ij}^{GL-SA} = 1\right) &= w_{ij}^{GL-SA} \\ P\left(p_{ij}^{GL-SA} = 0\right) &= \left(1 - w_{ij}^{GL-SA}\right) \end{aligned} \tag{7.3}$$

As described in Ch. 5, the recurrent weights between SA cells are scaled such that:

$$\sum_j w_{ij}^{SA-SA} = 4 \quad \forall i \tag{7.4}$$

where $\sum_i w_{ij}^{SA-SA}$ is the sum of afferent weights received by post-synaptic cell i. In practice a cell will receive synapses from 9 cells (encoding 9 actions in its predicted successor state) so an individual synapse will usually have a value of 0.44 (4/9).

7.2.5 *Results*

Figure 7.11 shows that if the agent's encoded cognitive map indicates (by way of reduced synaptic weights) that one path to a goal has a lower transition probability than another path, then the agent is much more likely to take the latter alternative path. In this experiment, the weight of the SA to SA synapses encoding the lower gate was reduced to 10% of their original value. We can see that in the normal condition (where the weights of the SA to SA synapses encoding both gates are equal) the agent ends up with an approximately equal chance of taking either gate. However, in the reduced condition, the agent has a much lower chance of taking the gate encoded by reduced SA to SA synaptic weights. In other words, if we reduce the weight of an SA to SA synapse to indicate a lower transition probability, the model is less likely to attempt that transition and more likely to prefer a more certain alternative route. The probabilistic propagation mechanism allows the model to incorporate transition probability into the planning process.

7.2.6 *Discussion*

The biological plausibility of the probabilistic planning mechanism is debatable, and largely depends on the timescale of the planning process because at longer timescales synaptic variability will simply translate into higher or lower firing rates. The probabilistic propagation mechanism makes much more sense if planning is considered to operate close to the level of individual spikes, similar to the model proposed by Ponulak 2013 [3], but interpreted as a rate coded model.

If we assume that planning occurs on a relatively short timescale, say, 1 second, and that state-action cells fire at a rate between 5 and 50Hz, and that most plans require activity to propagate through somewhere between five and twenty synaptic connections (more on this in Ch. 8) then it seems likely that the form of planning detailed here will rely on a relatively small number of spikes. If this is the case, then the probability of synaptic transmission between SA cells can genuinely

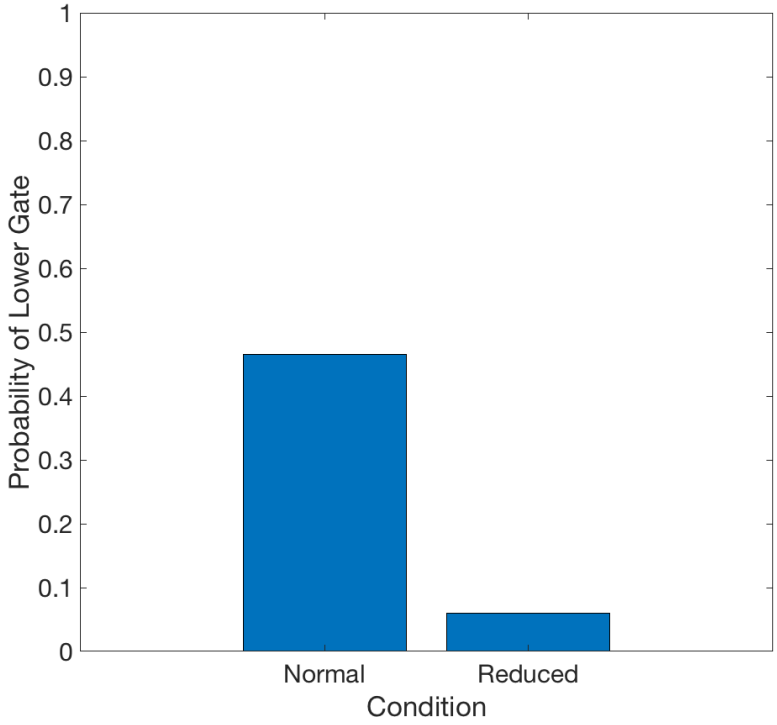


Figure 7.11: The probability that an agent will take the lower gate in the 'normal' and 'reduced' conditions, based on 100 trials in each condition. In the normal condition, the SA to SA weights encoding a transition through the lower gate (see Fig. 7.10 for the structure of the two-gate environment) are kept at their original value of 0.44. The synaptic weights for both gates are therefore equal, and so the agent takes each path with approximately equal (50%) probability. In the reduced condition, the synaptic weights associated with the lower gate are reduced to 10% of their original value. Under the probabilistic planning paradigm described in Sec. 7.2, activation has a significantly lower probability of passing through the lower gate in any given timestep due to the reduced synaptic weights encoding this transition and therefore the agent is much less likely to pass through the lower gate. Note that the likelihood of taking either path depends on the ratio between the synapses encoding transitions through each gate and not on the absolute value of these synapses.

affect the propagation of activation through the network and therefore a mechanism on the lines of that detailed in this chapter could exist and allow the network to plan in environment with probabilistic transitions.

7.3 DESCRIPTION OF THE DETOUR TASK

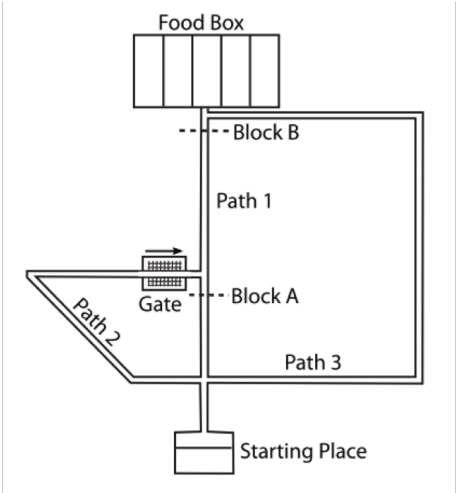


Figure 7.12: The maze used in the Tolman & Honzig detour task, in which rats navigate through a maze to a food box [2]. The maze consists of three pathways (Path 1, Path 2 and Path 3) with different lengths. A block can be introduced at point A (preventing the rat from navigating through Path 1), or point B (preventing the rat from navigating through Path 1 or 2). A gate near the second intersection prevents rats from going right to left. This figure is reproduced from Martinet et al. 2011 [2] in accordance with the Creative Commons Attribution (CC BY) licence.

The Tolman detour task (illustrated in Fig. 7.12) is a maze with three paths of varying length: Path 1 (short), Path 2 (medium) and Path 3 (long). At the end of the maze is a food source (a goal state in the model). Put briefly, rats are allowed to explore the maze and are then expected to seek the reward. Path 1 is shortest; if it is open, rats will take it. If Path 1 is blocked at point A, so that only Path 2 and Path 3 are accessible, rats will then take Path 2 the majority of the time, without extensive retraining. If Path 1 is blocked at point B, rats take Path 3. This behaviour happens quickly, without extensive exploration, suggesting that rats are relying on a previously learned

map of their environment to predict the outcomes of alternative trajectories through the environment.

7.4 DETOUR TASK SIMULATION PROCEDURE

A fully self-organising version of the model (as formulated in Ch. 6) was trained using a protocol given in Martinet 2011, which was designed to emulate the protocol originally designed by Tolman and Honzig [2]. The training process took place over 14 simulated 'days' and consisted of several different procedures. The model plans using probabilistic propagation (as described in Sec. 7.2). This was required to allow the network's environment model to encode unexpected obstacles in a more realistic manner that includes some uncertainty about a newly encountered obstacle, rather than operating on the deterministic proposition that a transition is either possible or impossible, and so to allow the model to replicate the uncertainty and mistakes made by the experimental rats. Without this mechanism, the model simply takes the best path at all times. The probabilistic propagation mechanism is not therefore required to successfully complete the task, but is required to reproduce realistic results.

A consequence of representing transition probabilities using synaptic weights, and of the fact that the agent lacks any ability to probe a state without actually entering it, is that it is possible for the agent to believe that all three paths are blocked, and so refuse to enter any of them, thus not discovering that one of them is now open. To counteract this tendency, a stochastic exploration mechanism is used whereby if the agent has not moved for 200 timesteps the agent will then begin a short period of exploration (20 timesteps). A similar mechanism was used in Martinet 2011 [2].

DAY 1: A series of 3 "forced runs" were carried out. Each forced run consisted of a sequence of actions that the agent was forced to carry out. In other words, where the agent in Ch. 6 explored randomly, learning the SA cell responses, SA cell connectivity and gating cell connectivity as it went, the model now does the

same for a preplanned set of actions. Each forced run moved the agent from its start position along one path (P1 or P2 or P3) until the agent reached the goal; the three forced runs explored P1, P2 and P3 in succession. These forced runs were carried out by the network in Learning mode (see Section 7.1.4)

The forced runs were followed by nine trials in the open maze, where the model was switched into Planning mode (see Section 7.1.4) and all of the paths were unblocked. The end of the maze was set as a goal (see Fig. 7.12) and the agent was allowed to navigate freely towards the goal. (Although these nine trials are identical and are not not strictly necessary in our formulation, we have kept the training procedure exactly as given in Martinet 2011 [2].)

DAYS 2-14: On these days the model remained in Planning mode, although synapses were altered at unexpected obstacles according to Sim. Proc. A (sec. 7.1.4.1). The model ran twelve trials every 'day' for thirteen days, with the end of the maze set as a goal. In ten of these trials, Path 1 was blocked at point A. However, the entrances to Paths 2 and 3 were also blocked, forcing the agent to move to block A. Every time the agent encountered a block, the synapses making up the cognitive map were manually altered as described in Section 7.1.4.1. This alters the cognitive map to include the block, and so alters the propagation of planning activity through the SA layer.

When the agent reached the block at point A, the entrances to Paths 2 and 3 were reopened so that the agent was free to choose either Path 2 or Path 3. The agent's choice was recorded (see Figures 7.13 and 7.14). For each day, the 10 block A trials were randomly mixed with 2 non-successive runs with paths 2 and 3 blocked, to maintain the preference for Path 1. By non-successive, we mean that the agent never experienced 2 of these runs in a row.

DAY 15: Seven trials (in Planning mode) were run with a block placed at point B. As before, paths 2 and 3 were blocked until the agent

reached point B, and then reopened. The agent therefore had a choice between choosing path 2, which was unsuitable given the block at point B, or choosing path 3. The results from both Alvernhe 2011 and Martinet 2011 showed that the agents reliably chose path 3, as did the proposed model. See Figures 7.13 and 7.14.

In summary, the protocol included three types of trials:

OPEN: In this condition, all of the paths are unblocked and the agent is allowed to navigate freely. This condition was present on Day 1.

BLOCK A: In this condition a block was placed at point A, blocking Path 1 and forcing the agent to choose between Path 2 and Path 3. This condition was present on Days 2-14.

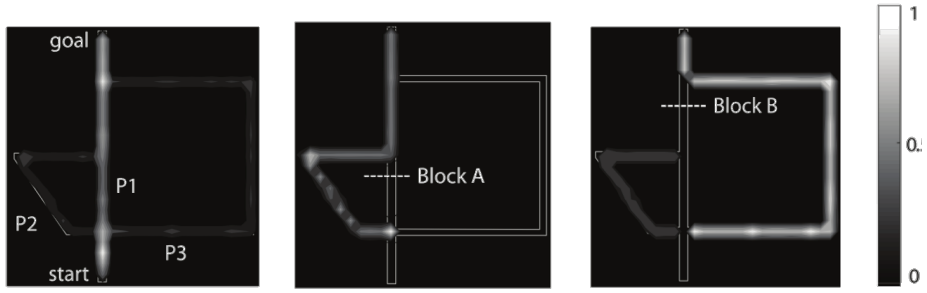
BLOCK B: In this condition a block was placed at point B, blocking Path 1 and forcing the agent to choose between Path 2 and Path 3. However, in this condition the placement of the block renders Path 2 useless, so that the agent must use its knowledge of the environment to realise that Path 3 is the only way to reach the goal. This condition was present on Day 15.

As in Martinet 2011, 40 agents were simulated, and their results collated to produce the final analyses, shown in Figures 7.13 and 7.14.

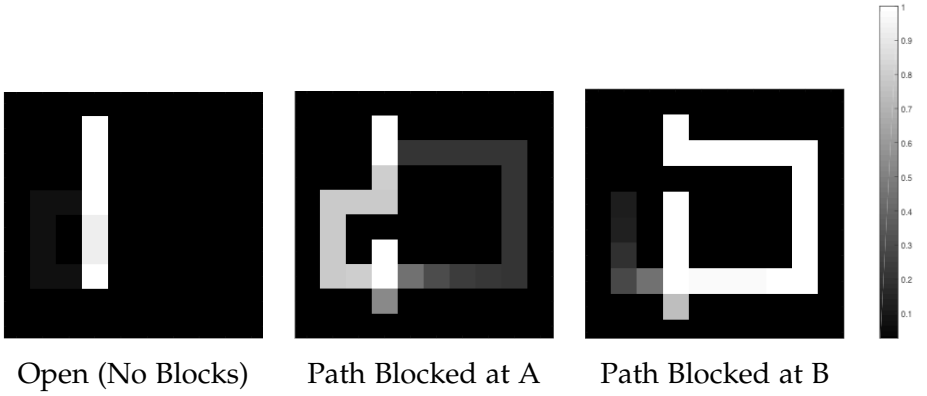
7.5 RESULTS OF THE DETOUR TASK

Figures 7.13 and 7.14 demonstrate that the proposed model is capable of reproducing the behavioural results of Alvernhe 2011 [21] and the model output of Martinet 2011 [2].

On Day 1, after exploring each of the three possible paths, the proposed model is allowed to navigate in the open maze (with no paths blocked) and shows a preference for Path 1, replicating the Path 1 preference seen in Martinet 2011. This is the shortest and most optimal route.

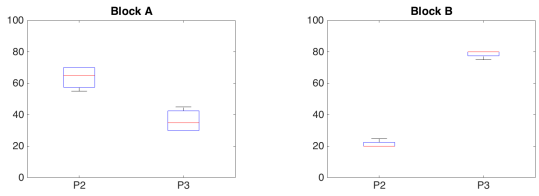


(a) Martinet 2011. These figures are reproduced from Martinet et al. 2011 [2] in accordance with the Creative Commons Attribution (CC BY) licence.



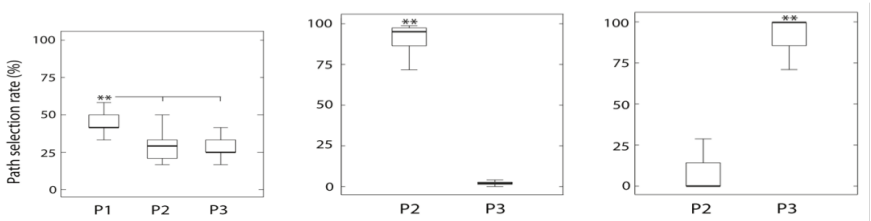
(b) The Proposed Model

Figure 7.13: Occupancy Grids for Tolman & Honzig Detour Task. The occupancy grids show the probability that a modelled agent will at some point pass through each section of the maze during the various trial types. The scale runs from 0 to 1. An occupancy of 1 means that every agent passed through that point during every trial of that type; an occupancy of zero shows that no agent ever passed through that point in any trial of that type. The top row shows results taken from Martinet 2011[2] and the bottom shows the model described in this section. We see that in the majority of Open trials, the agent takes the shortest possible route to the goal: the direct Path 1. In the majority of trials where Path 1 is blocked at Point A, the agent takes the now-shortest Path 2. Finally, in the majority of trials where Paths 1 and 2 are blocked at Point B, the agent takes the now-shortest Path 3. The qualitiative performance of both the Martinet model and the proposed model is similar.



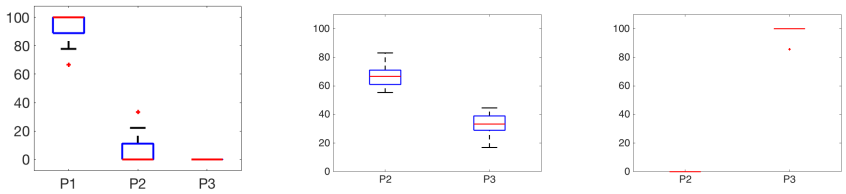
Path Blocked at A Path Blocked at B

(a) Boxplots showing the path selection rate of rats in Alvernhe 2011's detour task replication [21]. Note that the original graphs in Alvernhe 2011 have been redrawn as box plots to be more easily comparable to figures 7.14b and 7.14c. No information is available for path preference in the Open maze condition.



Open (No Blocks) Path Blocked at A Path Blocked at B

(b) Boxplots showing the path selection rate of Martinet 2011 for Paths 1, 2 and 3 in different conditions. Path 1 is only shown in the leftmost figure, because in other trials it is blocked and therefore impassable. We see that the Martinet model shows a preference for Path 2 when Path 1 is blocked at point A, and for Path 3 when Path 1 is blocked at Point B, qualitatively replicating the results of Tolman [5] and Alvernhe [21]. This figure is reproduced from Martinet et al. 2011 [2] in accordance with the Creative Commons Attribution (CC BY) licence.



Open (No Blocks) Block at A Block at B

(c) Boxplots showing the path selection rate of the proposed model for Paths 1, 2 and 3 in different conditions. Path 1 is only shown in the leftmost figure, because in other trials it is blocked and therefore impassable. The proposed model shows a preference for Path 2 when Path 1 is blocked at point A, and for Path 3 when Path 1 is blocked at Point B, qualitatively replicating the results of Tolman [5], Alvernhe [21] and Martinet [2].

Figure 7.14: Comparison of model output to previous experimental and modelling results. Each box plot shows the distribution of route choices over trials of a particular type (Open, Blocked A or Blocked B). (a) shows experimental results from rats. (b) shows the results given by Martinet 2011's model and (c) shows the results of the proposed model. The chosen path in each trial is recorded by measuring the last section of maze that the agent moves through before it reaches the goal.

On Days 2-14, when the path is blocked at point A, the model chooses the shorter and more optimal Path 2 more frequently, whilst occasionally choosing Path 3, which is longer but also valid. When the path is blocked at point B on Day 15, the model reliably chooses Path 3, the only valid option.

7.6 DISCUSSION

The behaviour of the model on the detour task replicates the essential finding of experimental and modelling literature: the agent reliably chooses shorter routes towards the goal. This result is intended to demonstrate that the model is able to replicate quantitative experimental results on one of the key model-based planning tasks. We have reused the detour task procedure described by Martinet 2011 [2], which was taken from the original Tolman and Honzig paper [119]. The model proposed in this chapter differs from that proposed by Martinet 2011 in several significant ways.

Firstly, the model described by Martinet 2011 plans using the value of activation at different points in the map. Specifically, activity is injected into the map at the goal state and allowed to propagate through the map. As it propagates, the activity decays. This creates a gradient: the cell with the highest activation in each state represents the optimal action to take in that state. Unfortunately, the propagating activity eventually decays too much for SA cells to plausibly react to it, as described in the Martinet paper and shown in Fig. 2.10. The activity gradient can therefore only extend so far through the map, and in a complex map such as a maze this activity may decay very quickly. Martinet introduces a variable-resolution mapping mechanism that can extend this range, but this mechanism requires that the environment is composed primarily of long straight-line segments connected by sharp discontinuities, a condition that is true of the detour maze task but does not extend well to other environment types. See Sec. 2.7 for more details.

In constrast, the proposed model does not require an activation gradient. In the proposed model, the optimal SA cell for each state

receives activity first, and so the first SA cell to receive activation in each state inherently represents the optimal action to take in that state. The level of activation is not relevant. This means that the activity does not need to decay, and so we do not experience the dropoff problem described in Martinet 2011 (Fig. 7.9). The planning mechanism is therefore more extensible to different map types and sizes, and also allows us to introduce a flexible hierarchical system for improving the efficiency of planning (see Ch. 8 and 9).

The proposed model and Martinet 2011 also differ in how they adapt to dynamically changing environments. Martinet 2011 manually identify cells whose synapses must be adjusted to encode a new obstacle and then hardwire the synapses between these cells to encode this information. To be specific, they set the relevant synaptic weight to 0.9 if the agent experiences a successful state transition, and halve the weight if the agent experiences a failed transition. We used a similar mechanism, described in Sec. 7.1.4.1, to obtain the detour task results that we have show in Figs. 7.13 & 7.14, but later work showed that the proposed model is also able to adjust its synapses using the Hebbian mechanisms proposed in Chapters 4, 5 and 6.

Having described both the learning and planning mechanisms of the proposed model, and tested it on a canonical behavioural task, the next chapters (8 and 9) will discuss the extension of this model to a formulation that encompasses hierarchical behaviour.

8

A HARDWIRED MODEL FOR PRODUCING HIERARCHICAL BEHAVIOUR USING SEQUENCE CELLS

Extensive evidence shows that humans represent space in a regional and hierarchical fashion. The clustering of spatial points has an effect on spatial judgements and navigational strategies [49]. In particular, Holscher et al. showed that human participants represented a complex public building regionally, in terms of floors and building sections [50], while Hirtle et al. showed that such regional representation occurs even in spaces that do not have explicit structure [49]. Noack et al. recently demonstrated that sleep is partially responsible for this regionalisation, increasing awareness of implicit and explicit semantic regions in a virtual environment [51].

Experimental work has also indicated that humans plan routes at different levels of the hierarchical representation. When giving route directions, either in advance or during travel, participants reliably produce high-level route segments in order to optimise information-to-memory-load [52]. Timpf et al. (2003) break down highway navigation into three levels of abstraction: *route planning* operates on a very high level, finding routes given certain constraints; *driving* is performed at a fine level; and *driving instructions* are given at a level partway between the two, dividing the route into segments that amalgamate a certain amount of driving activity [54]. Wiener & Mallot (2003) similarly show that humans appear to use a fine-to-coarse planning heuristic when navigating: a route plan contains fine-space information for the close surroundings and coarse space information for the rest of the route [55]. There is also evidence that route familiarity is an important element in human navigation [56].

It also seems clear, although there is a paucity of work on the subject, that the scale on which high-level route segments are generated

and used depends on the size of the task in question. People give directions of a roughly similar length for routes whose lengths differ by orders of magnitude, provided that route fragments of the appropriate scale exist, indicating that the existence of high-level actions (driving instructions) allows complex routes to be compressed into a practical length. At the same time, planning tasks performed at the same level of abstraction show planning times are partially proportional to absolute path length [57].

A true model of map-based route planning in the brain should be able to explain both high-level and low-level planning behaviour. We hypothesise that the hierarchical route representations described by Klippel et al., Timpf et al. and Wiener et al. [52, 54, 55] could be produced by using behaviourally significant sequences of state-action combinations to represent certain elements of the route more concisely and abstractly. We further hypothesise that these sequences could subsequently be implemented during planning as 'mental shortcuts'. In this case, the agent would be able to plan partially at a more abstract level; it would select sequences rather than individual state-action combinations. By representing a long route as 3 sequences rather than 500 individual state-action combinations, the agent would greatly reduce the number of choices required to plan that route. In effect, these sequences would function as high-level actions or blocks of policy that move the agent from one state to a far-off state, allowing the agent to plan across large sections of the state-action space concisely and efficiently.

This chapter will discuss how these sequences can be encoded within the network and how they can be used to plan faster and more efficiently. As such, this chapter will not discuss the process responsible for learning sequence representations. Chapter 9 is dedicated to the process of learning these sequences.

8.1 TESTING POSSIBLE SEQUENCE CELL CONNECTIVITIES

8.1.1 *Statement of Problem*

We need to develop a neural structure that can store sequences of state-action cells in a form that is useful when planning.

8.1.2 *Hypothesis*

Chapter 3 explained that the wave-propagation planning mechanism used by the proposed model relies exclusively on the *timing* of goal-based activity propagation through cells encoding the cognitive map, rather than how much activation those cells receive. The *first* SA cell to receive activation in each state represents the optimal action to take in that state, and the level of activation is not relevant. Since the model does not require time for the activity to 'settle', or for cells to integrate activity, the main factor limiting planning time is the speed of wavefront propagation. By increasing the speed at which activity propagates through the network, the planning process can be made much faster, provided that the spread of activity still produces viable plans. We hypothesise that frequently-used sequences of state-action combinations can act as shortcuts through state space, allowing activation to spread faster and further without sacrificing planning power.

Such a hierarchical planning mechanism would influence the model's choice of actions during planning. As explained above, the speed of activity propagation through the recurrent connections of the SA layer is used to determine optimal routes during planning. If frequently-used sequences of state-action combinations are used to speed up the propagation of activity, these familiar sequences may be selected in preference to an optimal route. Fig. 9.6 observes this effect, and shows that the model's behaviour in this aspect corresponds to the preference for known routes observed in experimental studies [56, 120]. The model might also display habitual behaviour in non-navigation tasks.

The use of 'shortcuts' through the SA layer might also make planning more robust to noise, since transmission of activity between

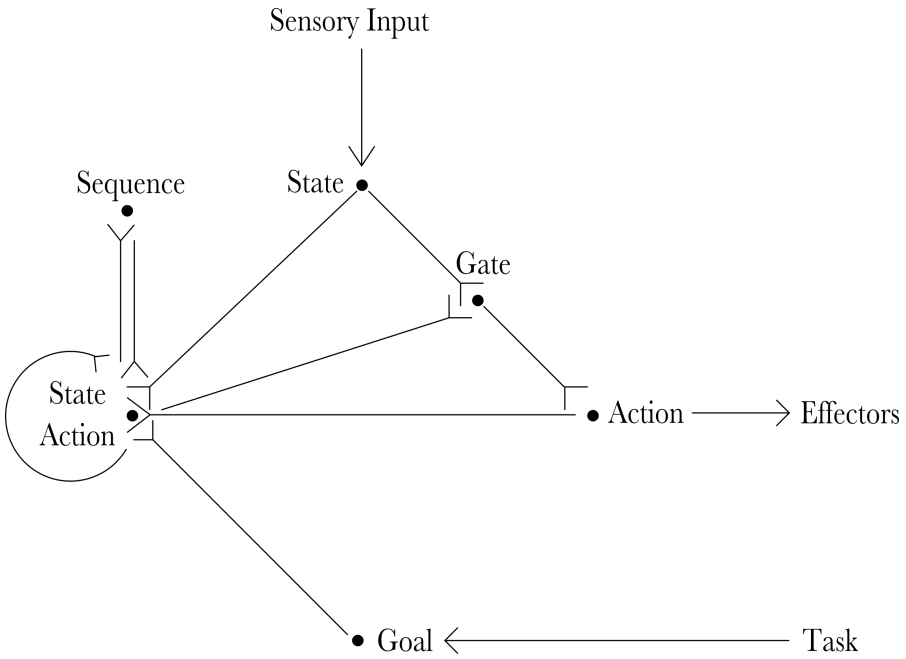


Figure 8.1: Architecture of the Proposed Model Including Sequence Cells. Unlike the basic network architecture (Fig. 3.1) this architecture has efferent and afferent connectivity between the SA layer and a layer of sequence cells.

neurons is a primary source of noise and the use of shortcuts is intended to significantly reduce the number of transmissions of activity between SA cells. However, since the activity of SA cells in active columns is kept constant (Sec. 3.2), and since planning in the model is dependent on the timing of activation propagation rather than on the precise activation value of SA cells, the effect of the hierarchical mechanism on planning in noisy conditions would likely be relatively small. Please note that the term 'column' refers to cortical minicolumns (3.1.2) unless otherwise specified.

We hypothesise that this hierarchical planning mechanism can be implemented by an extra layer of cells, called *sequence cells*, (fig 8.1) whose connectivity to the state-action layer allows them to represent frequently used sequences of actions. These sequences encode behaviourally useful sequences which provide relatively direct routes through the state space. In determining the most likely form of sequence representation, there are two primary concerns:

1. A sequence cell must represent a useful sequence of state-action combinations. To do this, it must be linked to a particular set of state-action cells that represent a behaviourally useful sequence.

2. The planning mechanism implemented by the state-action cell layer should be able to call up and use sequences during the planning process. In other words, state-action cells must be able to activate sequence cells at an appropriate time—via the synapses *from state-action cells to sequence cell*s—and sequence cells must be able to influence the activity of state action cells, via the synapses *from sequence cells to state-action cells*.

8.1.3 *Task*

The task is similar to the grid world navigation task described in Chapter 3. The agent is placed at a set position in an 8x8 open grid world shown in Fig. 3.6a. The agent can move one space at a time in eight compass directions[1] or stay still, giving nine possible actions.

A set state on the other side of the open grid world is designated as the goal and the agent is required to navigate to this state to complete the task. If the agent reaches the goal location then the task has been completed successfully. If the agent fails to reach the goal location within 1000 timesteps then the agent has failed the task.

In this experiment we hardwire the connectivity between the SA and sequence cell layers to encode a useful SA sequence, and we test several different patterns of SA-sequence connectivity as follows:

CONNECTIVITY 1 The sequence cell receives afferent synapses from the last SA cell in the sequence, and sends efferent synapses to all of the SA cells in the sequence.

CONNECTIVITY 2 The sequence cell receives afferent synapses from all of the SA cells in the sequence, and sends efferent synapses to all of the SA cells in the sequence.

1 N, NE, E, SE, S, SW, W, NW

CONNECTIVITY 3 The sequence cell receives afferent synapses from the first SA cell in the sequence, and sends efferent synapses to all of the SA cells in the sequence.

8.1.4 *Network Model*

The proposed model's architecture is depicted in Figure 8.1 and contains:

- State cells
- Goal cells
- State-action (SA) cells
- Gating cells
- Action cells
- Sequence cells

The simulation procedure is very similar to that described in Chapter 3. Algorithm 8.1 describes one timestep. The state cells fire, encoding the current location of the agent. These cells are one-hot: each state cell uniquely represents a single state and only one state cell fires at a time, denoting the current state. These cells are stimulated automatically, and are considered to be the output of unmodelled sensory processes. At the same time, the goal cells fire, encoding the desired state to be navigated to (i.e., the goal).

Activity spreads out from the goal location through the SA layer as a propagating wave. During this process (at every timestep) there is back-and-forth propagation of activation from the SA layer to the sequence layer (eq. 8.1) and back to the SA network (eq. 8.2).

Sequence cells receive activation propagated from the SA cell layer as follows:

$$h_i^{SQ} = \sum_j w_{ij}^{SA-SQ} r_j^{SA} \tag{8.1}$$

Algorithm 8.1 *Planning in the Hardwired Model with Sequence Cells.* This describes the network's operations during one step in the planning process, as an extension of alg. 3.1. The steps that have been added in this version are emphasised with italics. Steps in (brackets) only occur if there is activity in the action cell layer (signifying that an action has been selected for the agent's current state).

CELL FIRING State Cells & Goal Cells Fire (eqs. 3.3 & 3.4)

activity propagation SA Cells to Sequence Cells (8.1)

activity propagation SA Cells, Goal Cells & Sequence Cells to SA Cells (8.2)

INHIBITION SA Cells: Rescale SA Activity in All Active States (8.3)

ACTIVITY PROPAGATION State Cells & SA Cells to Gating Cells (8.4)

INHIBITION Gating Cells: Threshold (8.5)

ACTIVITY PROPAGATION Gating Cells to Action Cells (8.6)

INHIBITION Action Cells: Winner-Take-All (8.7)

(AGENT) Agent Moves to Successor State

(RESET) All Cells Reset to Zero Activation

where r_j^{SA} is the firing rate of SA cell j and w_{ij}^{SA-SQ} is the synaptic weight from that SA cell to a sequence cell i, and these sequence cells—when active—pass activation back to the SA cells according to eq. 8.2.

SA cells are stimulated by a combination of goal input, recurrent SA activity and input from sequence cells as follows:

$$h_i^{SA} = \sum_j w_{ij}^{GL-SA} r_j^{GL} + \sum_j w_{ij}^{SA-SA} r_j^{SA} + \sum_j w_{ij}^{SQ-SA} r_j^{SQ} \tag{8.2}$$

where $\sum_j w_{ij}^{GL-SA} r_j^{GL}$ is the input received from the goal cells, $\sum_j w_{ij}^{SA-SA} r_j^{SA}$ is the recurrent SA input, and $\sum_j w_{ij}^{SQ-SA} r_j^{SQ}$ is the input from the sequence cells (see below). SA cells experience divisive inhibition, rescaling the activity of SA columns so that each currently active column is rescaled to sum to 1, and each inactive column remains inactive (with a sum of 0):

$$\sum_i r_i^{SA} = \begin{cases} 1 & \sum_i h_i^{SA} > 0 \\ 0 & \sum_i h_i^{SA} = 0 \end{cases} \tag{8.3}$$

where $\sum_i h_i^{SA}$ is the sum of all SA cells in a given column.

A layer of gating cells has been hardwired such that each gating cell receives afferent synapses from one state cell and one state-action cell, and sends synapses to one action cell.[2] Activity propagates to the gating cells as follows:

$$h_i^G = \sum_j w_{ij}^{S-G} r_j^S + \sum_j w_{ij}^{SA-G} r_j^{SA} \tag{8.4}$$

The gating cells are under heavy inhibition such that they can only produce firing if they are receiving both SA and state input:

$$r_i^G = \begin{cases} h_i^G & h_i^G > t^G \\ 0 & h_i^G \leq t^G \end{cases} \tag{8.5}$$

where t^G is another thresholding constant.

Activity propagates from the gating cells to the action cells as follows:

$$h_i^A = \sum_j w_{ij}^{G-A} r_j^G \tag{8.6}$$

$$r_i^A = \begin{cases} h_i^A & h_i^A > t^A \\ 0 & h_i^A \leq t^A \end{cases} \tag{8.7}$$

where h_i^A is the activation of an action cell i, w_{ij}^{G-A} is the weight of a synapse from a gating cell j to that action cell, r_j^G is the firing rate of gating cell j and r_i^A is the final firing rate of the motor cell after thresholding inhibition using the constant t^A.

2 For each gating cell, the state cell and action cell is the same state and action that is encoded by the state-action cell. A gating cell receiving input from state-action cell S1A2 will also receive input from state cell S1 and send activity to action cell A2. Chapter 6 describes this in more detail and describes a mechanism for self-organising such cells in a biologically plausible fashion.

The thresholding inhibition (eq. 8.7) has the effect of preventing any action cell from firing unless it is strongly stimulated by a gating cell. If any action cell begins to fire despite this inhibition, winner-take-all inhibition is applied to the action cell layer to select a single action. The agent takes the action represented by the winning action cell (i.e., it moves one space in the specified direction) and updates the world. The firing rate of all cells is then reset to zero.

This chapter is focused on investigating the mechanism of planning and the role that sequence cells may play in the planning process. We therefore do not discuss how the synapses in this model self-organise: the self-organisation of the basic network is covered by Chapters 4, 5 and 6, while the following Chapter 9 will discuss the self-organisation of connectivity between the state-action layer and the sequence cell layer. The diagram in Fig. 8.1 depicts the overall architecture of the model, and Table 3.1 gives relevant parameters.

Parameter	Value
Action Cell Threshold t^A	0.2
Gating Cell Threshold t^G	0.55

Table 8.1: Table of Parameters for the hardwired model with sequence cells described in Section 3.1

8.1.5 *Results with Connectivity 1: A Sequence Cell Receives Afferent Synapses from the Last SA Cell in a Sequence, and Sends Afferent Synapses to All SA Cells in the Sequence*

The connectivity in this section is that only the last SA cell in a sequence is able to activate the corresponding sequence cell, while that sequence cell is able to activate all of the SA cells in the sequence (see Figs. 8.2 & 8.3). That sequence cell encodes a series of movements eastwards (Fig. 8.3a). The network connectivity instantiates a backwards causal model, with activity flowing from the goal backwards through the state transitions encoded by the SA recurrent synapses. This backwards propagation of activity ensures that goal signals reach the last

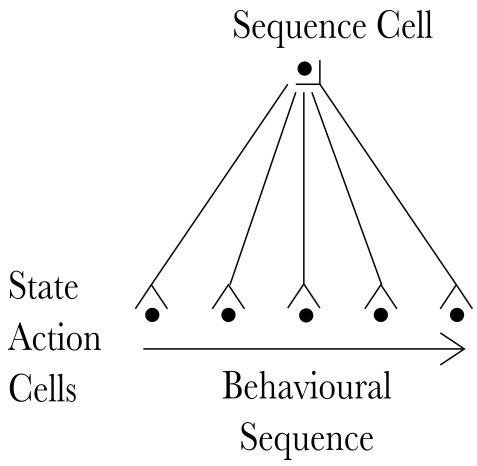
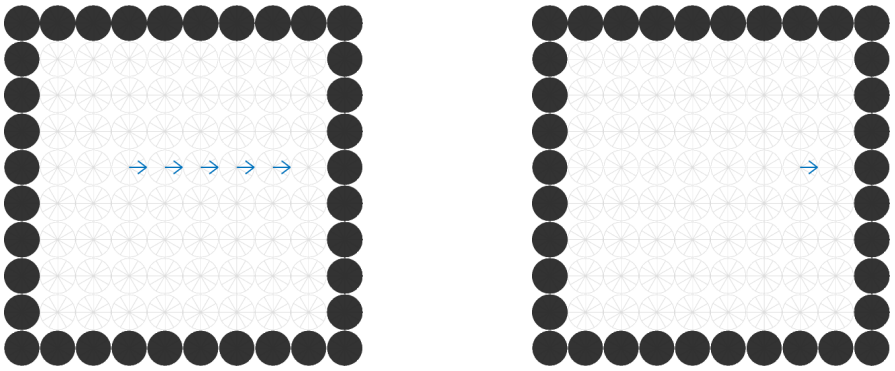


Figure 8.2: Diagram of connectivity condition 1 between a hardwired sequence cell and the SA cells representing a behavioural sequence. Only the last SA cell in the sequence is able to activate the sequence cell.

SA cell in the sequence first. This activates the sequence cell, which in turn activates all of the other SA cells in that sequence. By comparing the propagation of activity with and without this sequence cell (figs. 8.4 and 8.5 respectively) we can see that the wave of activation propagates through the SA layer faster when this sequence cell is available, allowing faster planning. We investigate this effect in more detail in Section 8.2.

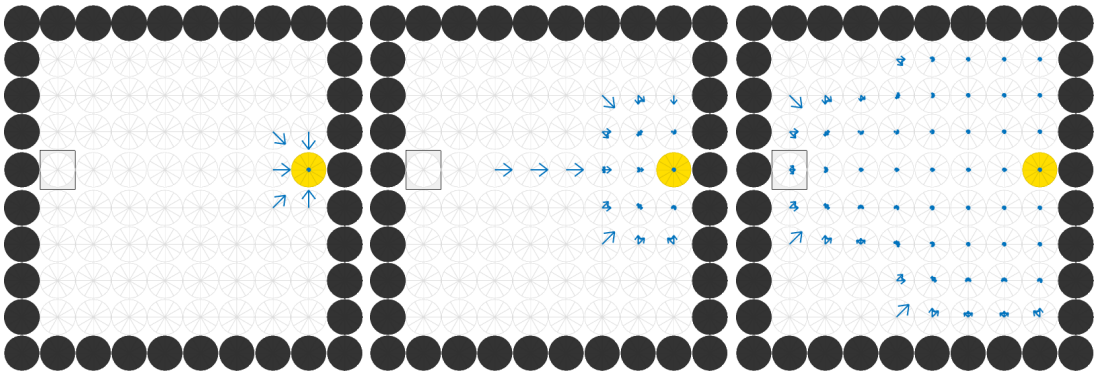
Planning success is 100% under these conditions (Fig. 8.6). If the goal and agent positions are reversed, the sequence is not useful and we see that the last SA cell in the sequence is the last activated. Since the wave of SA activity has now passed the sequence, the activation of the sequence cell does not affect the planning process and so does not interfere with it (Fig. 8.7). For this reason, the success rate is still 100% in the reversed condition (Fig. 8.8).



(a) Synapses From Sequence Cell to SA Cells

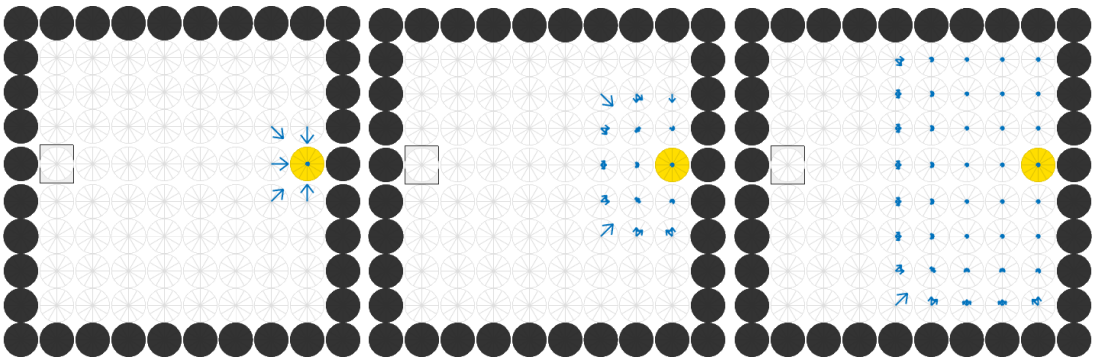
(b) Synapses From SA Cells to Sequence Cell

Figure 8.3: Connectivity between the SA layer and a sequence cell in connectivity condition 1. The sequence cell in question encodes a series of movements eastwards. (a) shows that the sequence cell sends synapses to all of the SA cells in that eastwards sequence and (b) shows that the sequence cell receives synapses from the last SA cell in a sequence. This figure uses compass plots to represent the synapses to/from SA cells in each state; Fig. 5.8c gives more information on this representation.



(a) t = 2 (b) t = 3 (c) t = 5

Figure 8.4: Propagation of Activity Through SA Layer in simulation with connectivity condition 1: The sequence cell receives synapses from the last SA cell in a sequence and sends synapses to all of the SA cells in that sequence. This figure uses compass plots to represent the firing rates of SA cells in each state; Fig. 6.1b gives more information on this representation. We can see that as activity spreads outwards from the goal, rather than spreading at a constant rate (as in Fig. 8.5) it activates a sequence cell at t = 3 and thence activates a sequence of SA cells encoding movement from the left of the environment to the right (b). In the next few timesteps, activity spreads from every point on this sequence, allowing the wavefront of activity to continue expanding from a more advanced position.



(a) t = 2 (b) t = 3 (c) t = 5

Figure 8.5: As Fig. 8.4, but without connectivity from the SA layer to a sequence cell. We can see that when sequence cells are not present, activity propagates through the SA layer at a constant rate. In Fig. 8.4c, the sequence cell acts as a 'shortcut' to allow activity to propagate faster than the base rate demonstrated in this figure. This means that in Fig. 8.4c, the wave of activity propagating from the goal state is able to reach the agent's current state by t = 5; in Fig. (c) shown here it is not able to do so.

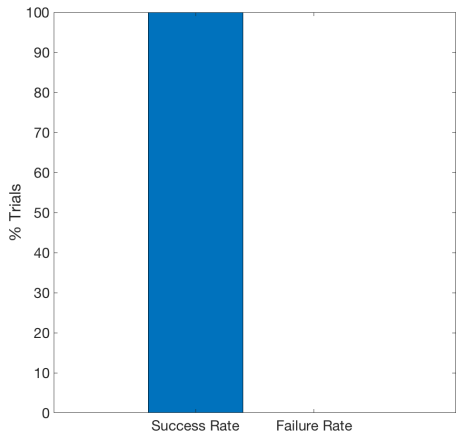
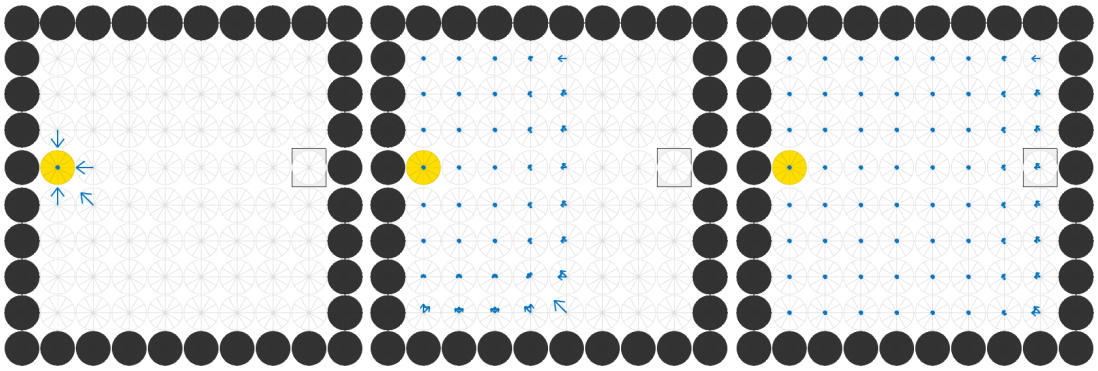


Figure 8.6: Performance in 100 planning trials with connectivity condition 1: The sequence cell receives synapses from the last SA cell in a sequence and sends synapses to all of the SA cells in that sequence. At the beginning of each trial, the agent and the goal were put on two sides of an 8x8 open environment (see Fig. 8.5). We see that in this case, the network is able to achieve a 100% success rate.



(a) t = 2 (b) t = 5 (c) T = 8

Figure 8.7: Propagation of Activity Through SA Layer with connectivity condition 1, when the starting position of the agent and the position of the goal are switched.

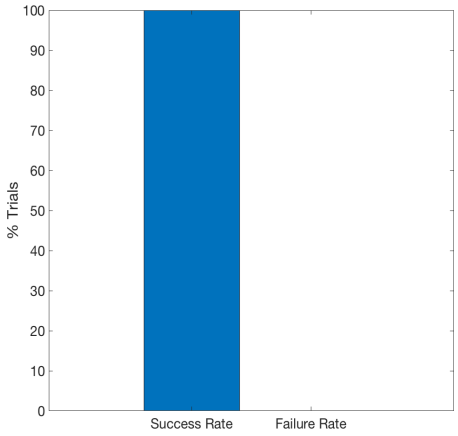


Figure 8.8: Performance in 100 planning trials with connectivity condition 1, when the starting position of the agent and the position of the goal are switched (Fig. 8.7) so that the agent is moving westwards and the eastwards sequence is now maladaptive. We see that even in this case, the network is still able to achieve a 100% success rate.

8.1.6 *Results with Connectivity 2: A Sequence Cell Receives Afferent Synapses From All of the SA Cells in a Sequence, and Sends Afferent Synapses to All SA Cells in the Sequence*

In this condition, all SA cells in a sequence are able to activate the corresponding sequence cell, and that sequence cell is able to activate all SA cells in the sequence (figs. 8.9, 8.10). If the sequence is useful, the model in this condition behaves as it does in the previous section: the wave of SA activity from the goal reaches the last SA cell in the

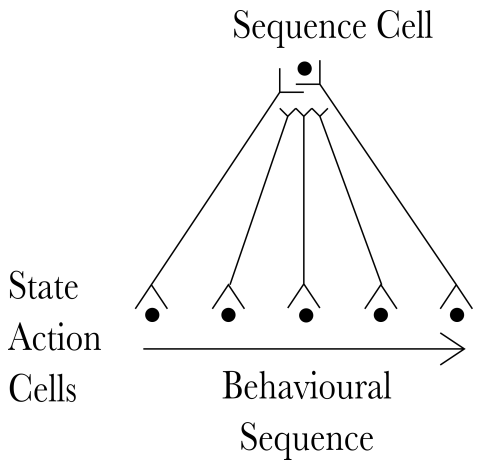
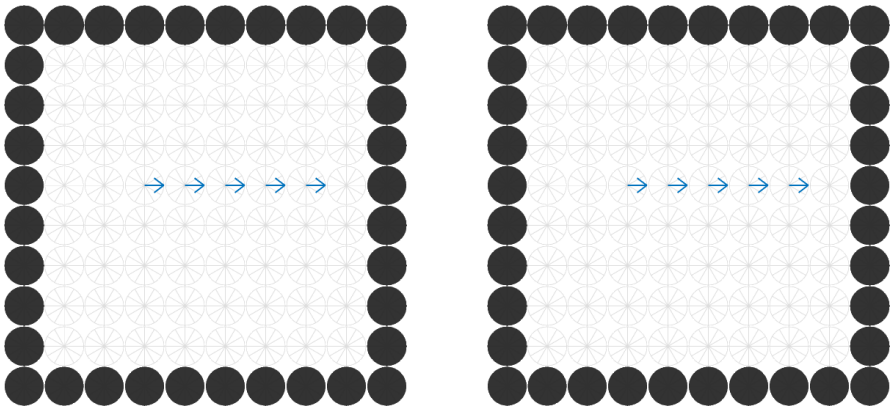


Figure 8.9: Diagram of connectivity condition 2 between a hardwired sequence cell and the SA cells representing a behavioural sequence. All SA cells in the sequence are able to activate the sequence cell.

sequence first, which propagates it along the sequence by activating all of the SA cells in that sequence (fig 8.11). For this reason the model has a 100% success rate in this condition (Fig. 8.12).

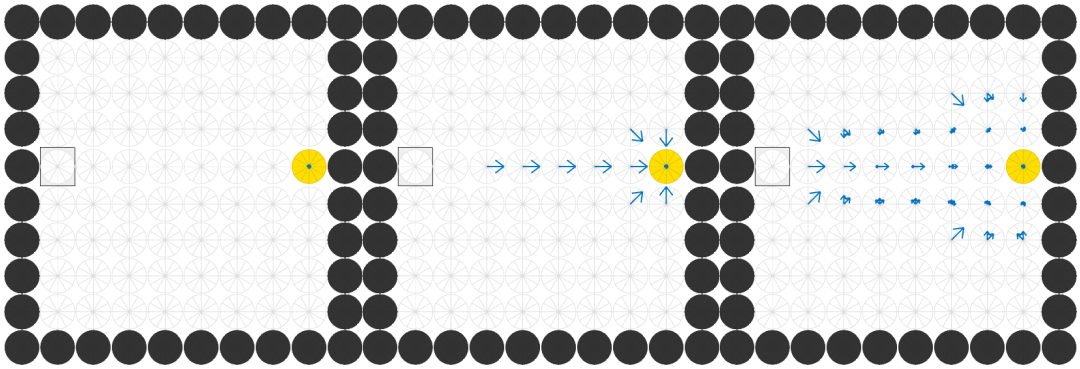
However, if the agent and goal positions are reversed, so that the sequence is not useful, the sequence will be activated as soon as the wavefront reaches the first SA cell in the sequence. The problem is that the sequence can then misdirect the agent (Fig. 8.13), even 'pushing' it away from the goal. In this case, the success rate of the model becomes 0% (Fig. 8.14).



(a) Synapses From Sequence Cell to SA Cells

(b) Synapses From SA Cell to Sequence Cells

Figure 8.10: Connectivity between the SA layer and a sequence cell in connectivity condition 2: the 'all-to-all' condition. The sequence cell receives synapses from all of the SA cells in a sequence and sends synapses to all of the SA cells in that sequence. Compass plots are used to denote the synapses to/from SA cells representing each state as in Fig. 8.3.



(a) t = 1 (b) t = 2 (c) t = 3

Figure 8.11: Propagation of activity through the SA layer with connectivity condition 2: 'all-to-all' connectivity (Fig. 8.10). Compass plots are used to denote the firing rates of SA cells representing each state as in Fig. 8.4. (a) shows activity being fed into the goal state. (b) and (c) then show that when the wave of activity reaches the last SA cell in the sequence, the sequence cell is activated and activates all SA cells in the sequence, allowing fast propagation of activity through the SA layer.

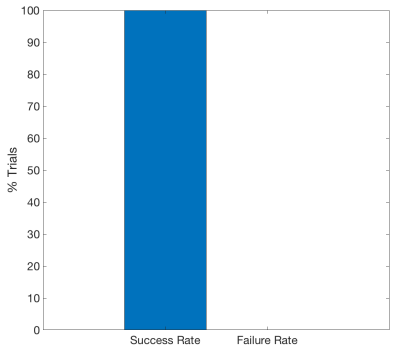
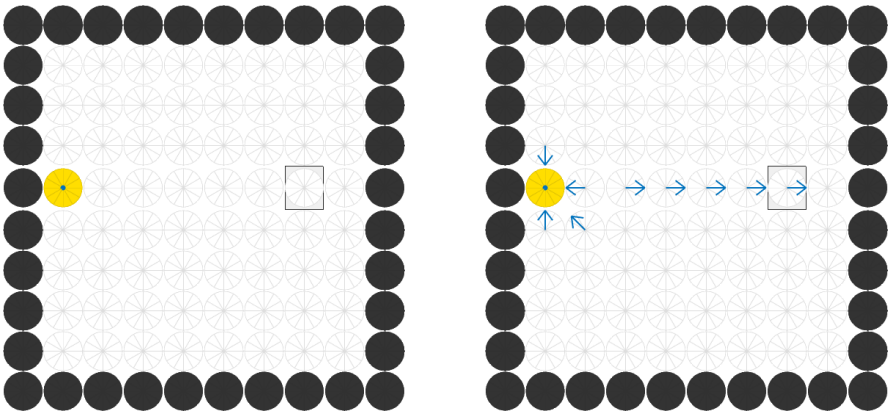


Figure 8.12: Results of 100 trials with connectivity condition 2: 'all-to-all' connectivity (Fig. 8.10). The network achieves robust performance in this condition.



(a) T = 1 (b) T = 2 (agent moves at this timestep)

Figure 8.13: Propagation of activity through the SA layer with connectivity condition 2, if the starting position of the agent and the position of the goal are reversed. We observe that because the sequence cell can be activated by any SA cell in the sequence it is now activated even though the goal-based activity is actually propagating in the opposite direction. The wrong SA cells are therefore activating, and the agent will now move in the wrong direction.

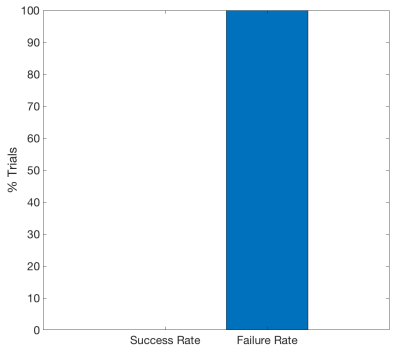


Figure 8.14: Results of 100 trials with connectivity condition 2: 'all-to-all' connectivity (Fig. 8.10) and the starting position of the agent and the position of the goal reversed. After switching the starting and goal states, we find that the network is now unable to complete the task, and produces a 0% success rate.

8.1.7 *Results with Connectivity 3: A Sequence Cell Receives Afferent Synapses from the First SA Cell in a Sequence, and Sends Afferent Synapses to All SA Cells in the Sequence*

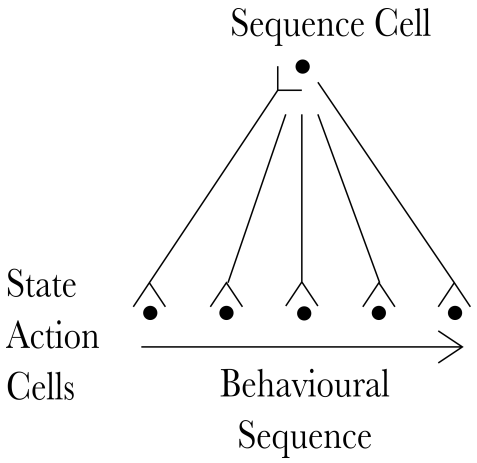
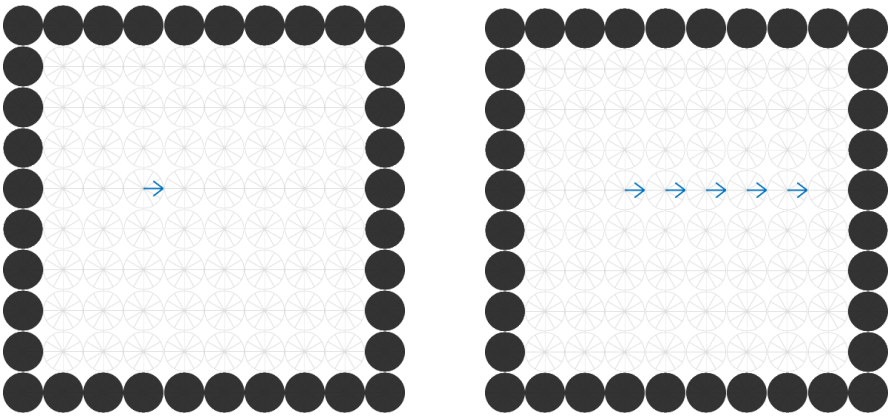


Figure 8.15: Diagram of connectivity condition 3 between a hardwired sequence cell and the SA cells representing a behavioural sequence. Only the first SA cell in the sequence is able to activate the sequence cell.

In this condition, only the first SA cell in a sequence is able to activate a sequence cell, and that sequence cell is then able to activate all SA cells in that sequence (figs. 8.15 and 8.16). Even if the sequence is useful, the sequence cell is essentially irrelevant in this condition, as it only activates once the wave of SA activity has already passed the se-

quence and so it cannot speed up propagation (Fig. 8.17). We can see this by comparing figs. 8.11 and 8.17. In Fig. 8.11, the spreading activation stimulates a sequence cell, allowing a sequence of SA cells to be activated simultaneously. The activation has therefore nearly reached the agent's current state by t = 3. In contrast, the spreading activation in Fig. 8.17 spreads at a constant rate, and takes 8 timesteps to reach the agent's current state. The model is able to plan successfully, with a success rate of 100% on this task (Fig. 8.18) but there is no gain in planning efficiency.

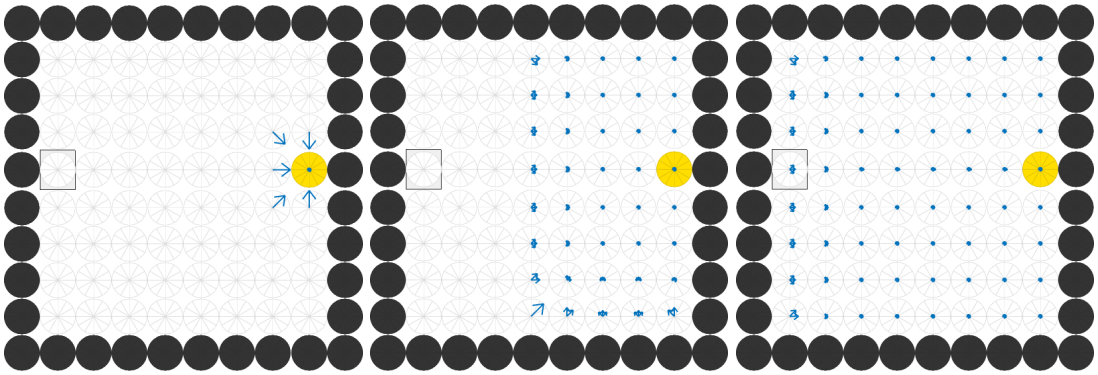
If the positions of agent and goal are reversed, so that the sequence is not useful, the sequence will still be activated when the wavefront reaches the first SA cell in the sequence and so it will be the wrong way around, misdirecting the agent (Fig. 8.19). In this case the success rate is 0% (Fig. 8.20).



(a) Connectivity from SA Cells to a Sequence Cell

(b) Connectivity from a Sequence Cell to SA Cells

Figure 8.16: Connectivity between the SA layer and a sequence cell in connectivity condition 3. The sequence cell only receives synapses from the first SA cell in a sequence and sends synapses to all of the SA cells in that sequence. Compass plots are used to denote the synapses to/from SA cells representing each state as in Fig. 8.3.



(a) T = 2 (b) T = 5 (c) T = 8

Figure 8.17: Propagation of activity through SA layer in simulation with connectivity condition 3: The sequence cell receives synapses from the first SA cell in a sequence and sends synapses to all of the SA cells in that sequence. This figure uses compass plots to represent the firing rates of SA cells in each state; Fig. 6.1b gives more information on this representation. We can see that activity spreads outwards from the goal (a) at a constant rate (b) and reaches the agent's current state at t = 8 (c).

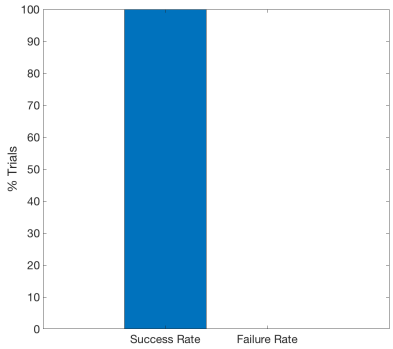
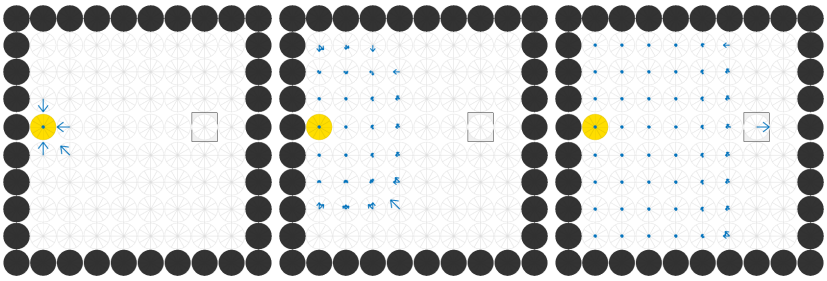


Figure 8.18: Performance in 100 planning trials with connectivity condition 3: The sequence cell receives synapses from the first SA cell in a sequence and sends synapses to all of the SA cells in that sequence. We observe that in this situation (where the agent is moving east and the SA sequence is therefore useful) the model is able to achieve a 100% success rate.



(a) T = 2 (b) T = 4 (c) T = 6

Figure 8.19: Propagation of activity through SA layer with connectivity condition 3, when the starting position of the agent and the position of the goal are reversed. We see that after activity is stimulated at the goal state (a), it appears to spread outwards at a constant rate (b) as in Fig. 8.17. However, in (c) we see that an unsuitable action has been chosen. This is because the activity has now reached the first SA cell in the learned sequence (Fig. 8.16) and therefore activated the sequence cell, which in turn has activated all of the SA cells in the learned sequence. Since that SA sequence includes the agent's current state, the agent moves according to the sequence. The sequence is unsuitable for this particular task and so the agent takes an unsuitable action.

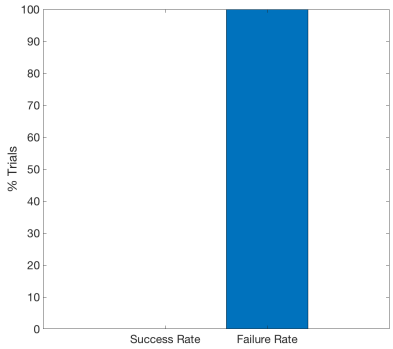


Figure 8.20: Performance in 100 planning trials with connectivity condition 3, when the starting position of the agent and the position of the goal are reversed. We observe that in this situation (where the agent is moving west and the SA sequence is therefore maladaptive) the model is unable to successfully complete any tasks. Fig. 8.19 shows that this is because the agent is being forced to move in the wrong direction by a wrongly-activated sequence cell.

8.2 TESTING THE UTILITY OF SEQUENCE CELLS

8.2.1 *Statement of Problem*

In the previous section, we tested the ability of a layer of sequence cells with various connectivities to propagate activity through the SA layer. It was found that a particular form of connectivity, that is, connectivity condition 1 as shown in Fig. 8.2, has a 100% success rate in both goal/agent combinations (figs. 8.6 and 8.8) whereas the other forms of connectivity that were tested did not. Furthermore, by comparing the way that activity propagated with and without an available sequence cell (figs. 8.4 and 8.5 respectively) it appears that activation propagates through the SA layer faster when a sequence cell is available, allowing faster planning. We now wish to investigate this claim more rigorously, and examine the extent of the utility of sequence cells in the context of planning.

8.2.2 *Hypothesis*

The previous section, 8.1, suggests that a sequence cell (representing a sequence of state-actions) should be activated when the propagating goal-centred wavefront reaches the end of that sequence[3] and should in turn activate the rest of the SA cells in that sequence immediately. This means that the wave can propagate through a sequence of n state-actions (which would normally take n timesteps) in 2 timesteps: one timestep for activity to propagate to the sequence cell, causing it to fire, and another timestep for that sequence cell to activate all of the n SA cells that it is connected to.[4]

As explained in Chap. 3, the time taken to output an action is dependent on how quickly the SA wavefront propagates from the goal to the agent's current position. Put briefly, since the optimal action for each state is encoded by the state-action cell linked to the goal location by fewest transitions (and so fewest synapses), a wave of activity

3 Bearing in mind that the activity is spreading backwards from the goal.
4 The required connectivity is seen in Fig. 8.2.

beginning at the goal and propagating at a constant speed will reach the optimal state-action cell before the rest of the state-action cells linked to that same state. The gating cells detect this activation and express the appropriate action to the action cells, which produce motor output.

As such, we hypothesise that incorporating sequence cells into the network model should enable the agent to plan a route of a given length in less time. Furthermore, because the speed at which activity propagates through a sequence does not depend on the length of the sequence, we hypothesise that the agent should receive larger gains in larger and more complex routes, where the path lengths are longer and thus the required planning time (without sequences) is longer.

8.2.3 *Task*

The task is fundamentally similar to the grid world navigation task described by Chapter 3. The agent is placed at a random position in one of four grid worlds. Two of these worlds are the small (10x10) open and maze worlds depicted by Fig. 3.6. The other two worlds are the same but scaled by 2:1, so that they are 20x20. The agent can move one space at a time in eight compass directions[5] or stay still, giving nine possible actions.

A random state is designated as the goal and the agent is required to navigate to this state to complete the task. If the agent reaches the goal location then the task has been been completed successfully. The number of timesteps required to reach the goal is recorded, as is the number of physical actions.

A set of 100 tasks is performed in each of the four environments, by two agents. The first agent has access to a set of sequence cells that encode various trajectories through the environment. The second agent has no access to sequence cells.

5 N, NE, E, SE, S, SW, W, NW

8.2.4 *Method*

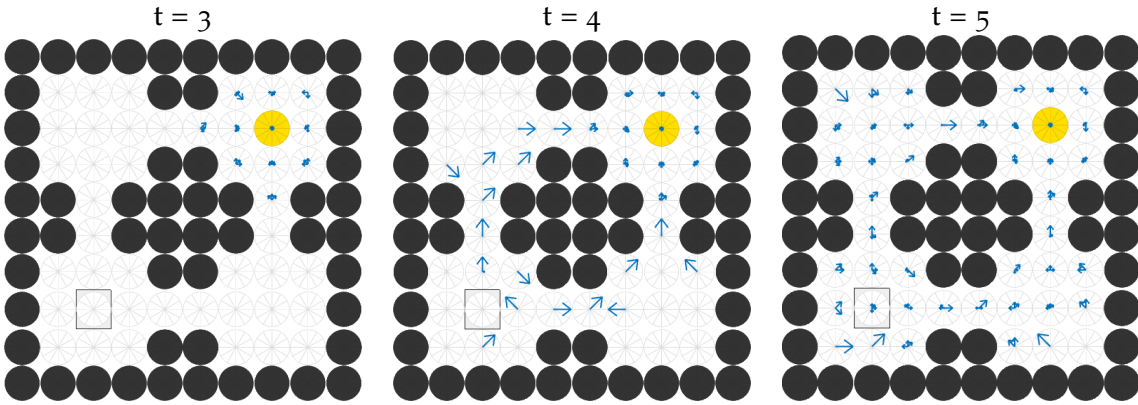
The network operates as in the previous section 8.1. The connectivity between SA cells and sequence cells is as described in Fig. 8.2 and Sec. 8.1.5: the sequence cell sends synapses to all SA cells in the sequence, but only receives synapses from the last SA cell in the sequence.

8.2.5 *Results*

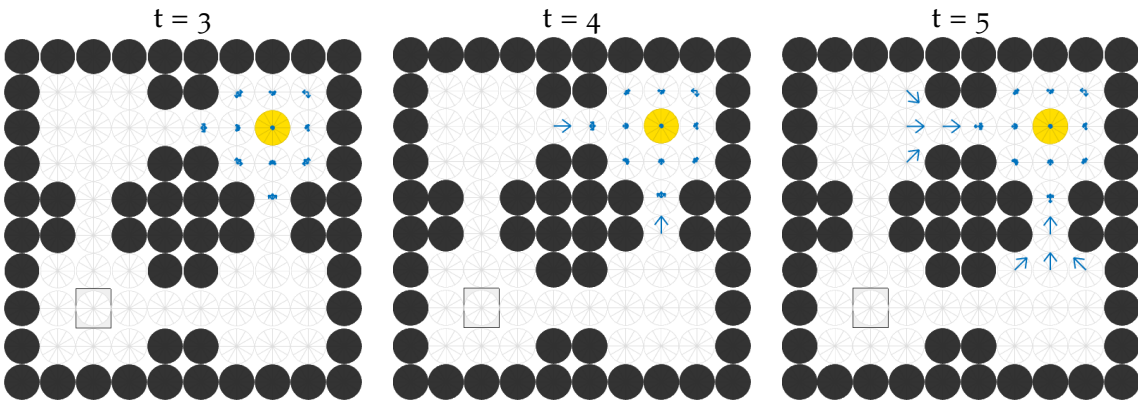
If a given sequence cell projects efferent synapses to all of the SA cells in a sequence but only receives afferent synapses from the last SA cell in the sequence (Fig. 8.2), that sequence cell will become active if the last SA cell in the sequence that it encodes becomes active. It will then immediately activate all of the other SA cells in that sequence. In effect, it allows the propagating wave to 'skip' those cells, propagating through the entire sequence at once. This allows the propagating wave to travel faster through certain areas of state-action space, an effect demonstrated in Figure 8.21, which shows the same navigation task with and without connectivity between the SA and sequence cells. In effect, the sequence cells have produced a two-level hierarchical planning mechanism, where planning happens on the level of individual state-actions but also on the level of larger state-action sequences. Self-organising mechanisms for learning this connectivity are described in Ch. 9.

Figure 8.22 demonstrates the increased speed of planning by comparing 100 planning tasks with and without using this hierarchical mechanism (the sequence cell layer). Efficiency is here defined as the amount of processing time (the number of timesteps required for activity to propagate from the goal to the agent's position) normalised by the path length (the number of steps required to reach the goal). In other words, efficiency here stands for the comparative speed of activity propagation from the goal to the SA cell representing a near-optimal action in the agent's current state.

Figure 8.22 clearly shows that planning is more efficient—the average planning time per action is lower—when a sequence cell layer is



(a) With sequence cells (t = 3 to 5)



(b) Without sequence cells (t = 3 to 5)

Figure 8.21: This figure shows the propagation of activity from the goal location through the SA layer during a simulated navigation task in the four-room environment shown in Fig. 3.6b. Results are shown for simulations with and without sequence cells. Each state is represented by a compass plot which depicts the activity of the SA cells representing that state (see Fig. 3.3b). The golden state represents the goal, and the boxed state represents the agent's current location. The agent moves once activity reaches its current state. We see that activity propagates considerably faster when sequence cells are available—compare (a) vs (b)—and that the propagating activity therefore reaches considerably more states after five timesteps.

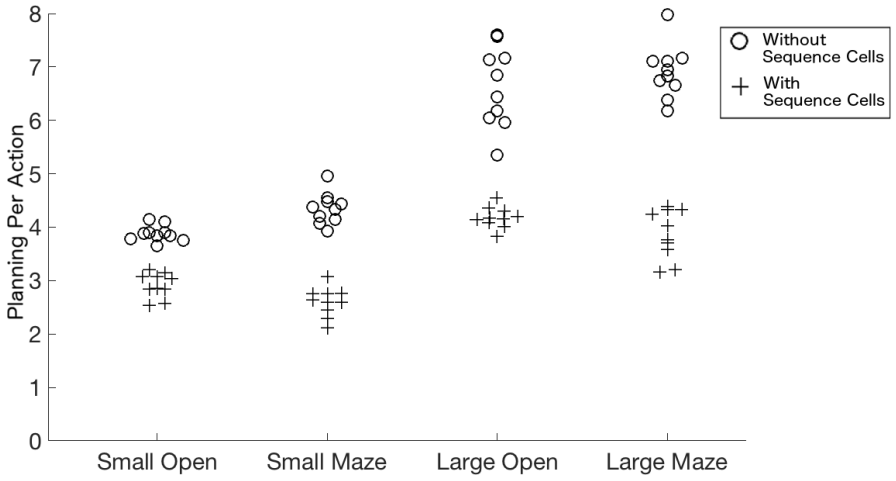


Figure 8.22: A swarm plot showing the effect of sequence cells on the efficiency of planning. 10 blocks of 10 tasks were taken with and without using sequence cells in each of four environments: a small (10x10) open environment (Fig. 3.6a), a small (10x10) maze environment (Fig. 3.6b), and large (20x20) open and maze environments with the same design. For each block of 10 tasks, we calculated the average number of timesteps required for the network to select each action and plot the result as one data point. We observe that, on average, the model requires considerably fewer timesteps to select an action when sequence cells are allowed to influence the planning process.

used (effectively producing a hierarchical planning mechanism). Furthermore, when we compare this efficiency gain between different environments, a pattern emerges: the greatest benefits occur when environments are larger.. For example, the effect of using sequence cells in the large maze environment appears to be much larger than the effect of using sequence cells in the small maze environment.

Figure 8.23 corroborates this finding by showing the specific effects of using or not using sequences on tasks with different path lengths. We see that use of the sequence cell mechanism has relatively little effect on short paths, but provides a significant (~40-60%) decrease in total planning time on path lengths of ~10 steps. These gains continue to increase as path length and environment complexity increase. Fig. 8.24 shows that the total planning time of the model seems to grow quadratically without the sequence mechanism and linearly when augmented with sequence cells.

The kind of state spaces experienced by a mammalian intelligence (especially a primate one) are very much more complicated and thus

larger than the state spaces used in this experiment, and problems in such large state spaces are difficult to solve using naive Markovian decision processes because the computation requirements in such environments are too large. The relationship between efficiency gains and environment size & complexity suggest that the efficiency gains would be much larger in real-life state spaces. As such, although we cannot currently model such large state spaces, this result gives us hope that the model would be able to plan in a realistic time period using this mechanism if we were able to test it in a real-life state space.

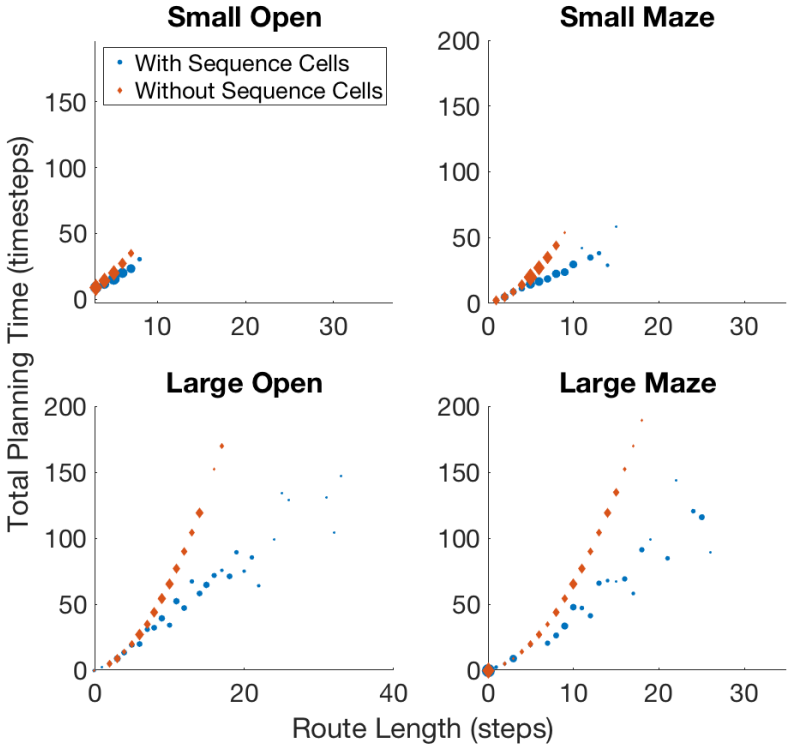


Figure 8.23: A figure showing how the total planning time varies with navigation tasks whose solutions are of different lengths. This demonstrates that use of sequences in planning produces low efficiency benefits during short tasks but increasingly large benefits as the required route length increases.

Fig. 8.25 shows that if two paths of equal length exist from the agent's starting position to its goal, then the hierarchical model incorporating sequence cells will tend to prefer familiar routes (those that have associated sequences) over other routes of equal distance. The model was trained in an 8x8 two-gate environment (Fig. 7.10) and

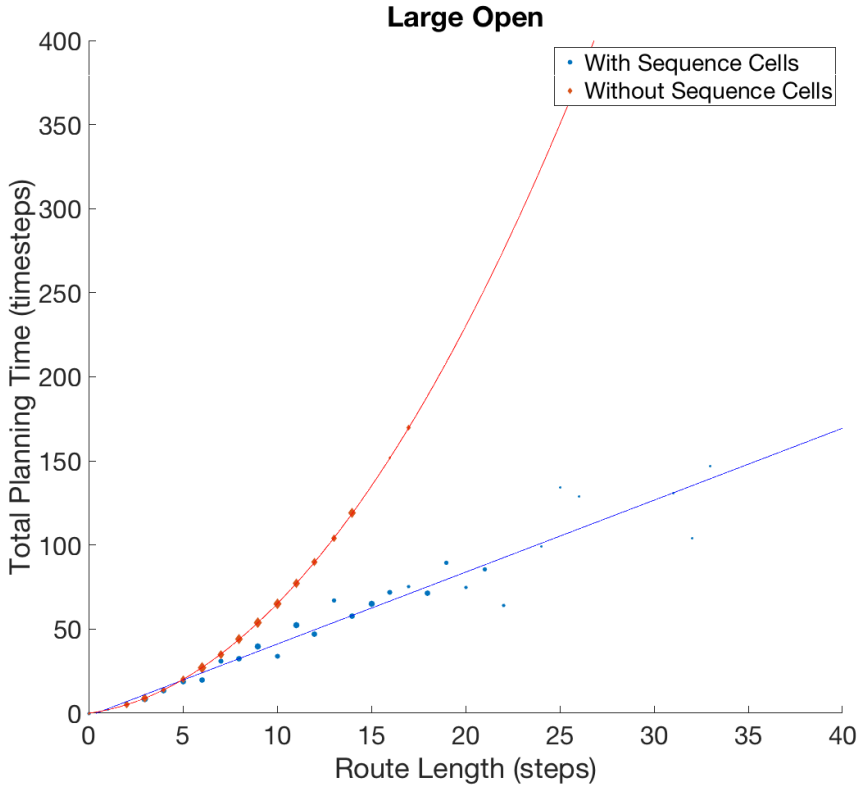
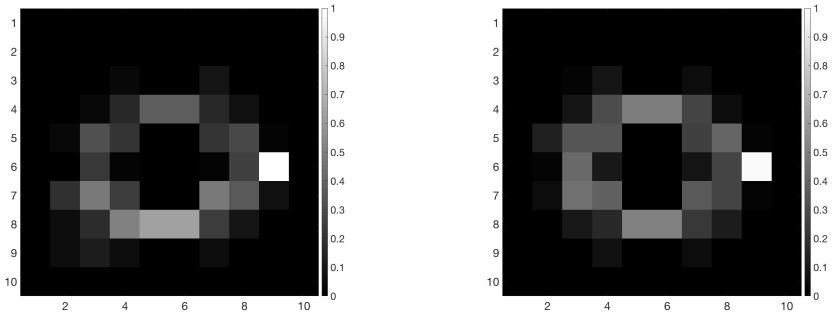


Figure 8.24: Fitting curves to the Large Open results. The basic model, without using sequences, produces planning times that increase quadratically with the route length, while the augmented model, using sequences, plans linearly with the route length. Extrapolating these fits predicts that the gains should grow very large as the size and complexity of environments increase, although we cannot currently verify this claim as the discrete state-action representation we use does not scale well beyond 20x20 states (discussed further in Sec. 10.2 and Fig. 10.1). To check the accuracy of both fits, we have calculated the Adjusted-R^2 values for each fit. The Adjusted-R^2 is a statistical measure of how close the data are to the fitted lines, adjusted for the number of terms that describe the fitted line. These values are 1 for the model without sequence cells and 0.9243 for the model with sequence cells. The value of Adjusted-R^2 must be between 0 and 1, and so these values show that both lines fit the data extremely well.

given a set of sequence cells encoding routes through the lower gate but not the upper gate. We see that the model is considerably more likely to choose the lower gate if it has access to these sequences. The agent used probabilistic propagation (sec. 7.2) for this planning task.



(a) With Sequence Cells (b) Without Sequence Cells

Figure 8.25: This figure compares the probability of an agent occupying various states in a two-gate environment (Fig. 7.10) over the course of 100 planning trials. (a) shows these occupancy probabilities when the model has access to sequence cells that encode a route through the lower gate. (b) shows the same model without access to these cells. The agent is required to move from position (2,6) to (9,6), which can be achieved in the same number of actions by passing through either gate. We see that the existence of these sequences biases the agent to use the familiar lower gate rather than the higher gate in (a). By contrast, when the agent does not have access to these sequences it uses both gates with approximately equal probability (b).

8.2.6 *Discussion*

Previous researchers have demonstrated that expert wayfinders such as the taxi drivers interviewed by Spiers et al. [121] use a common strategy to solve wayfinding tasks in known environments [122]. Although these wayfinders know a considerable proportion of the states and transitions available in their environment, in practice they mainly navigate using a small subset of paths [56, 120]. When given a wayfinding problem, the expert will plan the majority of the route along these well-trodden paths. Kuipers et al. [122] call these paths the 'skeleton' of a cognitive map: experts find a route from their start point

onto the skeleton, travel along the skeleton as far as possible, and then find a route from that location to the goal.

The sequence cells described in this chapter encode a similar skeleton structure and produce a similar preference for routes which lie on that skeleton (see Fig. 8.25). The sequences of actions that our sequence neurons encode are similar to the high-level route segments described by Klippel et al. [52] in that they denote a specific and delineated sequence of actions that efficiently represent a route and can be used to move to a goal. We demonstrate (see Fig. 8.24) that the use of sequence cells reduces the effect of path length on planning time; from quadratic to linear in our case. This corresponds with experimental work done by Ward et al. [123] who found that the planning time required to solve a 5-disc Tower of London task[6] correlated most strongly to the number of "subgoal chunks" in the task, where a "subgoal chunk" referred to a consecutive series of moves that all transfer discs to and from the same pegs, presumably in a relatively predictable manner. Ward & Allport speculate that such chunks comprise a "mental unit of planning" [123].

At present we have only investigated the role of these sequence cells in simple toy tasks but we believe that the hierarchical mechanism proposed here should be extensible to an arbitrary degree by adding further layers of sequence cells that encode sequences of sequences and so produce further speed-up, particularly for very large environments. Schmajuk et al. [124] demonstrate that the planning time of a nested hierarchical system in their region-based formulation varies logarithmically with the distance between the start and end states.

Speed-up from the use of a nested hierarchical mechanism could potentially explain the observation that people plan on different scales and levels of abstraction and are able to navigate over distances orders of magnitude apart, from short room-to-room movements within a house or workplace to long drives between cities and/or coun-

6 The 5-Disc Tower of London task consists of two configurations of 5 rings placed on three pegs. The rings can be moved between the pegs according to specific rules. The agent starts at one configuration and must perform a series of moves to produce the second configuration.

tries to multi-day sea voyages[52, 54, 55, 125]. Such distance-invariant navigation has also been observed in bats, one of the few mammals whose large-scale navigation has been extensively studied. Bats have been observed to recall the three-dimensional position of objects with an accuracy of 1-2 cm but can also navigate reliably and regularly to targets dozens or thousands of kilometres away [4]. Wild rodents have a similar range of navigational capabilities [4]. We believe that although planning-time-to-path-length correlations seem to exist for problems of the same scale [57], they do not seem to apply strongly to problems of different scales. For example, taxi drivers report being able to produce routes through London almost instantly [121] even though these routes may be several kilometres in length. This could be explained several layers of sequences operating at different scales, or by a regional hierarchy such as that used by Schmajuk et al. [124].

There are certain hierarchical effects that cannot be explained by the sequence cell hypothesis. The model does not provide an obvious way to replicate the effect of regional environment structure on perceived place relations. For example, Hirtle and Jonides 1985 [49] report that experimental participants perceive landmarks in the same spatial cluster to be closer together than they really are; participants also overestimate the distance between spatial clusters. As the model does not encode distance we cannot measure these effects directly; propagation time can be used as a proxy for the distance between points but in this case the effect of sequence cells is to *reduce* the perceived distance between clusters, which goes against Hirtle and Jonides' results.

Work by Wiener and Mallot [55] investigated the movement of participants through regions denoted by landmarks. These regions are differentiated by the use of car-, animal- and art-based landmarks. Wiener and Mallot found that participants chose routes that minimise the number of region boundaries that they passed through. They also found that participants chose routes that moved them to the target region as soon as possible. The environments we use in this thesis cannot represent such semantically-defined regions, and the networks we describe cannot identify them. Our sequence cells

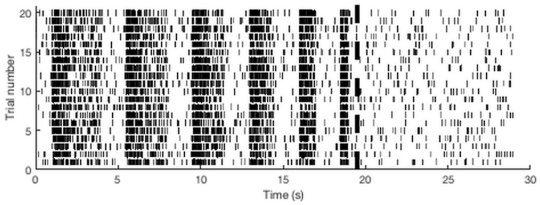
hypothesis cannot therefore directly explain these findings. If we assume that such regions generally correspond with clusters of states, however, and that sequence cells generally represent more frequently used SA sequences (see Fig. 9.3) it follows that sequence cells will usually represent bottlenecked transitions between regions. If this is so, then agents are likely to choose routes that use the fewest sequences and the model might therefore reproduce the preference for routes that pass through fewest region boundaries.

The nature of this model also prevents it from making predictions about the symbolic and linguistic representations of hierarchical element. For example, humans produce verbal route descriptions by chunking together several route elements in a sequence, but these route descriptions are biased towards the use of information that are easy to share verbally and easy to obtain visually, such as landmarks [52]. The model proposed in this thesis is not capable of recognising or incorporating such factors.

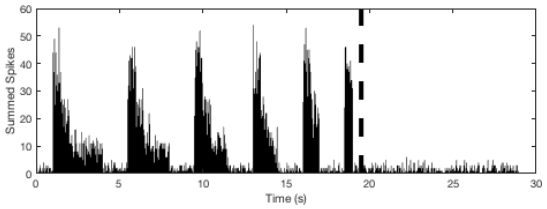
It should be emphasised that sequence-based representations and regional representations are likely to cooperate in the process of map formation and planning. Siegel et al. [125] argue that large-scale spatial cognitive maps are formed by combining landmarks and routes (of the kind that would be encoded by sequence cells) with a slowly-developing objective frame of reference in order to deduce the true structure of an environment. It is therefore quite possible that sequence-based planning plays a supplementary role to region-based planning, or vice versa, and that both are required for the development of true large-scale spatial maps.

8.2.6.1 *Sequence Cells in Pre-SMA*

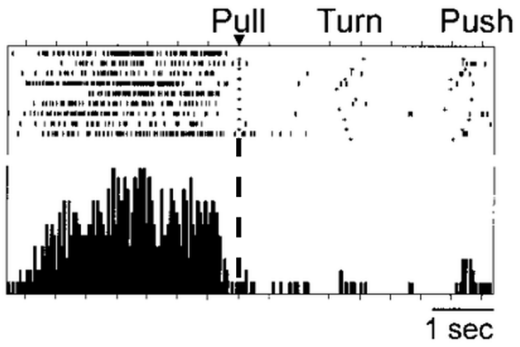
Sequence-selective cells have been found in the supplementary motor area (SMA) and pre-supplementary motor area (pre-SMA). These cells fire *before* a particular (previously learned) sequence is performed but do not fire *during* the motor performance of that sequence [126]. The pre-SMA is known to be connected to the prefrontal cortex [127], allowing interactions between sequence cells in the pre-SMA and state-action cells in PFC to occur as seen in the model.



(a) Sequence Cell Raster Plot (Generated from Firing Rate). This plot has been generated for comparison to the raster plot in Fig. 8.26c using a poisson distribution to generate spikes from the firing rate of the sequence cell. 0.5 seconds of spiking data were simulated per timestep.



(b) Histogram of Sequence Cell Firing



(c) pre-SMA Cell Recording [126]

Figure 8.26: Comparison of a simulated sequence cell to a recorded pre-SMA cell. In both the simulated and recorded data, we see that the cell has strong firing *before* but not *during* a specific sequence of actions (a dashed line marks the onset of this sequence). However, the simulated data displays periodicity, unlike the recorded data. This is discussed in the main text. The proposed model is rate-coded and so the raster plot (a) was generated from recorded firing rates using a Poisson distribution.

Figure 8.26 shows a simulated sequence cell firing during a modified version of the navigation task, and compares it to experimental recordings performed by Shima et. al [126]. As in other navigation tasks, the agent must navigate from its starting location to the goal. In this case, however, the starting location and goal are chosen manually to ensure that the only route to the goal included the sequence encoded by the sequence cell. The figure shows the recorded firing rates of this sequence cell before, during and after the sequence of actions encoded by the sequence cell.

As explained in Section 8.1 of this chapter, the network plans by propagating goal-based activation through the SA layer and recording which of the state-action cells responsible for the current state receive activation first. Sequence cells function as 'shortcuts' for the propagation of this wave. For this reason, the sequence cell will be strongly active before the agent performs the relevant sequence of actions. The propagating wave activates the sequence cell on its way to the agent's current state, so that it becomes active early and remains active until the propagating wave reaches the agent's current state. The activity then activates a gating cell and action cell as described by Sec. 8.1.4, and the agent takes an action that moves it towards the goal. The activity in the SA layer resets and the activity begins propagating again from the goal state.

This cycle of activity propagation and movement repeats until the agent reaches the first state that is involved in the sequence described by the connectivity of the sequence cell. Previously, the sequence cell was active for an extensive period of time while activity propagated further through the SA layer, beyond the sequence encoded by the sequence cells. However, SA cells responsible for the agent's current location are now within the encoded sequence and so can be stimulated *directly* by the sequence cell without required further time for activity propagation. This means that as soon as the sequence cell becomes active, so do the relevant SA cells, gating cells, and action cells. The sequence cell is therefore only active very briefly when the agent is performing the sequence which the sequence cell encodes.

To put it another way, the sequence cell is only active *after* the propagating activity reaches the sequence cell's preferred sequence *and until* that activity reaches the SA cell encoding the agent's current state. All SA cells in the sequence become active simultaneously, so if the agent's current state is in the encoded sequence, the sequence cell will only be active for a very short period of time because activity is not propagating further through the SA layer.

Figure 8.26b shows the simulated firing rate of the sequence cell, and figure 8.26a shows a spiking raster plot with the spikes generated according to the firing rates by a poisson distribution for comparison to the experimentally recorded data in Fig. 8.26c. We can see that the sequence cells in the proposed model have very similar properties to the pre-SMA cells recorded in this figure [126]. Not only do they both encode a particular sequence of actions [126], but they become active immediately before the encoded sequence is carried out and then cease to fire as the sequence is performed. Furthermore, impairing either SMA or pre-SMA inhibits the ability of monkeys to perform the learned sequences of actions when cued, even though they retain the ability to perform any of those actions individually [126].

The main difference between the simulated data and recorded data is the periodicity seen in figs. 8.26a and 8.26b. This periodicity arises from the fact that the simulated agent took several steps before the onset of the sequence, and so several 'cycles' of activity occurred as the activity spread out from the goal state to the agent's location to plan the next action. There are several possible explanations for this difference.

Firstly, the recordings took place under different task conditions. The pre-SMA recording in Fig. 8.26c was taken during a task in which monkeys were trained to perform three different movements, separated by short waiting times, in four or six different orders. The monkeys then reproduced one of these sequences several times based on visual cues. In other words, the sequences were not initiated as part of a goal-directed set of free movements, but as a cue-induced sequence. The difference in cell behaviour may therefore be partly task-based,

accessing the same representations through a slightly different mechanism.

Secondly, the model resets activity in the SA layer after each movement, and begins re-propagating activity from the goal. The activity in a true neural system is likely to be more noisy and less able to be inhibited so cleanly, especially since the true motor system is considerably more complex than that modelled here. Such complexity might blunt or hide the periodicity shown in the simulated data. A related factor may be the simplified nature of the model, which is rate-coded and based on distinct timesteps, and operates on a set of discrete states. Further work may be able to relax these constraints, and this may lead to less obvious periodicity.

9

ENCODING USEFUL BEHAVIOURAL SEQUENCES BY SELF-ORGANISING A LAYER OF SEQUENCE CELLS

The work described in the previous chapter demonstrated that sequence cells can be used to improve the efficiency of the network's planning. The planning process requires a wave of activity to travel from the SA cells representing the goal state to the SA cells representing the agent's current location. Sequence cells assist this process by acting as a 'shortcut' for the wave, allowing it to travel faster through frequently experienced areas of the state-action space. To do this, the sequence cells require a particular pattern of connectivity with and from the SA cells.

In this chapter, we show how the connectivity between the sequence and SA cells may come about through a variant of Hebbian learning—known as trace learning—which connects cells that tend to fire close together in time [114]. We also show how this process of unsupervised learning helps the sequence cells to encode the most biologically useful sequences available in the environment.

9.1 SELF-ORGANISING THE CONNECTIVITY BETWEEN SEQUENCE CELLS AND STATE-ACTION CELLS USING TRACE LEARNING

9.1.1 *Statement of Problem*

The previous chapter demonstrated that sequence cells require a specific form of connectivity to and from the SA cells in order to efficiently aid planning. We wish to explain how the model could self-organise such connectivity. Furthermore, since the number of possible trajectories that the agent may take is very large, including many that are not helpful in the majority of tasks, the sequence cells

should learn to encode the most behaviourally-useful sequences that the agent has experienced: sequences that move the agent directly between important states such as bottlenecked areas of the environment. How should the sequence cells identify and encode these sequences as the agent performs tasks?

9.1.2 *Hypothesis*

We hypothesise that the connectivity between sequence cells and SA cells comes about through a form of trace learning, similar to that seen in Ch. 5. As discussed in that chapter, the trace learning rule is a variant of Hebbian learning that ties together cells which fire close together in time. In this case a sequence cell that has been stimulated by an SA cell will remain in an active learning state for several more state-action combinations, therefore building up afferent and efferent connectivity to a sequence of SA combination cells.

The most behaviourally relevant sequences are those which provide a direct path through an important area of state-action space and so provide maximum augmentation to the network's ability. We hypothesise that such sequences are likely to be encountered more frequently than other sequences[1] when moving purposefully through the state space under the kinds of route planning mechanisms demonstrated in this thesis. Specifically, the basic route planning mechanisms described earlier in this thesis tend to move the agent along a direct route from its current state to a goal state. If sequence cells are incorporated into the network, then these cells can learn these direct routes through the state space that emerge from the basic planning mechanism.

In this way, the basic planning mechanism (which generates direct routes between states) operates in tandem with the hierarchical model extension incorporating sequence cells (which learn these useful dir-

1 Another possible indicator of significant sequences is that they pass through a bottleneck in state-action space such a door between rooms [66]. In practice this property is likely to be correlated with sequence frequency, because any task that requires moving from one collection of states to another will require the agent to pass through the bottleneck that separates them and so such bottlenecks are frequently used. We do not investigate this property separately.

ect routes). The final model formulation presented in this chapter therefore offers a biologically plausible solution to how hierarchical behaviour can self-organise in the brain through unsupervised learning.

The requirement for humans to accumulate extensive task experience to achieve a high level of automaticity supports this hypothesis, as does the fact that it seems to be necessary for learners to spend disproportionate amounts of time on aspects of a task that they find difficult if they wish to further improve their skills. To mimic this process, we decided to take the grid world navigation tasks that we had used to *test* the model in Chapter 8 and now use them to *train* the model.

9.1.3 *Task*

The task is fundamentally similar to the grid world navigation task described by Chapter 8. The agent is placed at a random position in one of four grid worlds. Two of these worlds are the small (8x8) open and maze worlds depicted by Fig. 3.6. The other two worlds are the same but scaled by ~2:1, so that they are 18x18. The agent can move one space at a time in eight compass directions[2] or stay still, giving nine possible actions.

A random state is designated as the goal and the agent is required to navigate to this state to complete the task. If the agent reaches the goal location then the task has been been completed successfully. If the agent fails to reach the goal location within 1000 timesteps then the agent has failed the task. At the same time, the agent is both learning and using sequences of actions by altering the connectivity between the sequence cell layer and the SA cell layer (see below).

Algorithm 9.1 *Learning Sequences During Planning.* This describes the network's operations during one step in the planning process, based on alg. 3.1. The steps that have been added in this version are emphasised with italics. Steps in (brackets) only occur if there is activity in the action cell layer signifying that an action has been selected for the agent's current state. All plasticity is Hebbian unless otherwise specified.

CELL FIRING State Cells & Goal Cells Fire (eqs. 3.3 & 3.4)

ACTIVITY PROPAGATION SA Cells & Goal Cells to SA Cells (8.2)

INHIBITION SA Cells: Rescale SA Activity in All Active States (8.3)

ACTIVITY PROPAGATION State Cells & SA Cells to Gating Cells 8.4)

INHIBITION Gating Cells: Threshold (8.5)

ACTIVITY PROPAGATION Gating Cells to Action Cells (8.6)

INHIBITION Action Cells: Winner-Take-All (8.7)

(AGENT) Agent Moves to Successor State

(activity propagation) State Cells, Action Cells and SA Cells to SA Cells (9.2)

(inhibition) SA Cells: Winner-Take-All (9.3)

(activity propagation) SA Cells to Sequence Cells (8.1)

(inhibition) Sequence Cells: Winner-Take-All (9.4)

(synaptic plasticity) Set Memory Trace: Sequence Cells (9.5)

(synaptic plasticity) Set Memory Trace: SA Cells (9.5)

(synaptic plasticity) Trace Learning: Sequence Cells to SA Cells (9.6)

(synaptic plasticity) Trace Learning: SA Cells to Sequence Cells (9.7)

(RESET) All Cells Reset to Zero Activation

9.1.4 *Network Model*

9.1.4.1 *The Network Architecture*

The network used in this experiment has the same structure as that used in the previous chapter (Fig. 8.1). It is assumed that basic environmental learning has already taken place, i.e., that the agent has already explored its environment and that the self-organisation of SA cell responses, recurrent SA connectivity and gating cell connectivity has already occurred as in Chapter 6. For the purposes of this experiment, only the connectivity between the sequence cell layer and the SA cell layer is considered to be plastic.

As stated previously, the connectivity between sequence cells and SA cells is learned during the performance of planning tasks. The sequence cells can therefore be activated in two circumstances: firstly, as part of the planning mechanism (as in Ch. 8), and secondly when an action is chosen for the purposes of self-organising their connectivity. For the sake of clarity, we assume in this chapter that the sequence cells are not participating in the planning mechanism (i.e., they are not being used as shortcuts during the goal-based activity propagation process. When activated for the purposes of self-organisation, sequence cells are mutually inhibiting and learn to encode the most frequently used state-action sequences over time. Algorithm 9.1 describes one timestep of this procedure.

9.1.4.2 *Route Planning—A Timestep Where No Action is Taken*

The state cells fire, encoding the current state using a one-hot encoding (eq. 3.3). At the same time, the goal cells fire, encoding the location of the goal (eq. 3.4). The SA cells are then updated by activity from the goal cells and from other SA cells according to the following equation:

$$h_i^{SA} = \sum_j w_{ij}^{GL-SA} r_j^{GL} + \sum_j w_{ij}^{SA-SA} r_j^{SA} \tag{9.1}$$

2 N, NE, E, SE, S, SW, W, NW

where $\sum_j w_{ij}^{GL-SA} r_j^{GL}$ is the input received from the goal cells, and $\sum_j w_{ij}^{SA-SA} r_j^{SA}$ is the recurrent SA input. (The SA layer does not receive activation from the state cells—this occurs only during learning for the purpose of self-organising the state-action responses and consequently the cognitive map.) The effect of this recurrent SA stimulation combined with firing from the goal cells is to create a propagating wave of activity spreading through the SA layer. The wavefront of this activity spreads a little further every timestep (see Fig. 3.7).

After the SA cells have been updated, they propagate activity to the gating cells (eq. 8.5). The gating cells are under heavy inhibition such that they can only produce firing if they are receiving both SA and state input (eq. 8.5). Activity propagates from the gating cells to the action cells (eq. 8.6). If no gating cell fires, then no action cell will fire; in this case no action will be taken at that moment and the wave will continue to propagate, allowing SA activation to spread further through the layer. However, if any action cell becomes active, then the set of events described in the rest of this simulation produce (below) will take place. These events are bracketed in algorithm 9.1.

9.1.4.3 *Learning Hierarchy of Action Sequences—If An Action is Taken*

The agent takes an action, and updates the world.

The SA layer receives a combination of sensory and motor inputs (as well as input from the recurrent synapses w_{ij}^{SA-SA}) that reflect the state-action combination chosen by the planning mechanism as follows:

$$h_i^{SA} = \sum_j w_{ij}^{S-SA} r_j^S + \sum_j w_{ij}^{A-SA} r_j^A + \sum_j w_{ij}^{SA-SA} r_j^{SA} \tag{9.2}$$

where h_i^{SA} is the activation of a state-action cell, $\sum_j w_{ij}^{S-SA} r_j^S$ is input from state cell j weighted by the incoming synapse, $\sum_j w_{ij}^{A-SA} r_j^A$ is the equivalent input from action cell j and $\sum_j w_{ij}^{SA-SA} r_j^{SA}$ is the recurrent SA input. This combination of inputs strongly activates the specific SA cell that represents the correct combination of state and action. WTA competition is then applied as follows:

$$r_i^{SA} = \begin{cases} h_i^{SA} & h_i^{SA} = \max_i \left\{ h_i^{SA} \right\} \\ 0 & h_i^{SA} < \max_i \left\{ h_i^{SA} \right\} \end{cases} \tag{9.3}$$

where $\max_i \left\{ h_i^{SA} \right\}$ is the activation value of the most active cell in the SA layer. This leaves only one SA cell active: the SA cell representing the state-action combination that the agent is currently experiencing.

The activity from this SA cell feeds into the sequence cell layer (eq. 8.1). WTA competition is then applied to the sequence cells as follows:

$$r_i^{SQ} = \begin{cases} h_i^{SQ} & h_i^{SQ} = \max_i \left\{ h_i^{SQ} \right\} \\ 0 & h_i^{SA} < \max_i \left\{ h_i^{SQ} \right\} \end{cases} \tag{9.4}$$

where $\max_i \left\{ h_i^{SQ} \right\}$ is the activation value of the most active cell in the sequence cell layer.

A trace rule is used to alter the connectivity between the SA layer and the sequence cell layer (in both directions). The trace rule is similar to those described in Ch. 5 (eq. 5.13) but uses a more complex trace value, which is calculated at this point in the timestep as follows:

$$\bar{r}_i{}^t = r_i^t(1-\eta) + \bar{r}_i^{t-1}\eta \tag{9.5}$$

where r_i^t is the current firing rate of a sequence or SA cell i at time t, $\bar{r}_i{}^t$ is the memory trace value of r_i at time t, $\bar{r}_i^{t-1}$ is the memory trace value of r_i at time $t-1$, and $\eta \in [0,1]$ is a constant that determines how much the current trace value reflects past firing rates as opposed to present firing rates. This trace value varies from that used in Ch. 5, which only considered cell firing one timestep in the past. In Ch. 5 only one timestep of information is necessary to encode a transition from one state to another. By contrast, in this chapter we are interested in forming sequence cells that will learn to associate a sequence of SA cells over a period of several timesteps. A slowly fading memory trace is therefore required, where the parameter η effectively determines the decay rate. Note that both forms of trace rule

are purely local, relying only on the activities of the pre- and post-synaptic neurons. As such, these learning rules can be considered relatively biologically plausible.

The effect of trace learning is to connect cells that fire over a period of time, where a high η value will tend to tie together cells over a longer period than a low η value (Fig. 9.4).

The synapses between SA cells and sequence cells are then updated according to a trace rule as follows:

$$\Delta w_{ij}^{SQ-SA} = k^{SQ-SA} \bar{r}_j^{SQ} \bar{r}_i^{SA} \tag{9.6}$$

$$\Delta w_{ij}^{SA-SQ} = k^{SA-SQ} \bar{r}_j^{SA} \bar{r}_i^{SQ} \tag{9.7}$$

where $\bar{r}_i^{SQ}$ and $\bar{r}_j^{SQ}$ are the trace values of sequence cells i and j respectively, $\bar{r}_i^{SA}$ and $\bar{r}_j^{SA}$ are the trace values of SA cells i and j respectively, and k is a constant denoting learning rate. Note that k^{SA-SQ} is considerably larger than k^{SQ-SA}. This is necessary because it means that the last SA cell to fire in a sequence will have an extremely strong weight to the sequence cell. This brings about the characteristic sequence cell connectivity condition 1 described in Ch. 8, where the sequence cell receives strong inputs from the last SA cell in the sequence (and therefore the first to receive activation from the propagating wavefront, which propagates from the goal) whereas all of the SA cells in the sequence receive relatively similar activity from the sequence cell (Fig. 8.2). This is further effected by the rescaling mechanism, which ensures that:

$$\sum_j w_{ij}^{SA-SQ} = 1 \quad \forall i \tag{9.8}$$

and any synaptic weights that are too low are reduced to zero:

$$w_{ij}^{SA-SQ} = \begin{cases} w_{ij} & w_{ij} \geq t^{SA-SQ} \\ 0 & w_{ij} < t^{SA-SQ} \end{cases} \tag{9.9}$$

where t^{SA-SQ} is a threshold constant.

w_{ij}^{SQ-SA} are rescaled so that

$$\sum_i w_{ij}^{SQ-SA} = 1 \quad \forall j \tag{9.10}$$

Note that the rescaling is performed with reference to the sequence cells in both in both eq. 9.8 and 9.10. This is because each sequence cell is intended to represent a single sequence, and so the amount of afferent connectivity that each sequence cell receives, as well as the amount of efferent connectivity, must be regulated. By contrast, a given SA cell may be part of many sequences or none.

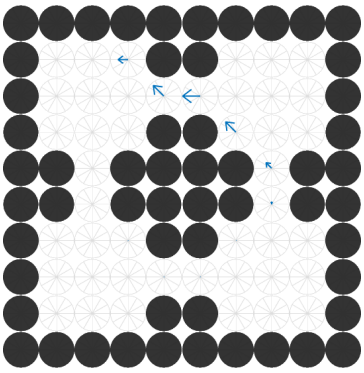
Finally, the activity of all cells is reset. Parameters are given in Table 9.1.

Parameter	Value
Trace Constant Eta η	0.5
Sequence Cells to SA Cells Learning Rate (k^{SQ-SA})	11
SA Cells to Sequence Cells Learning Rate (k^{SA-SQ})	10^{26}
SA Cells to Sequence Cells Threshold (t^{SA-SQ})	0.1

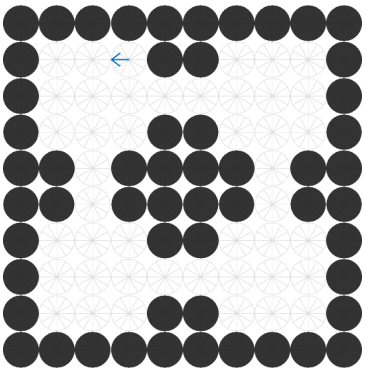
Table 9.1: Table of parameters for the learning of sequence cells as described in Section 9.1.

9.1.5 *Results*

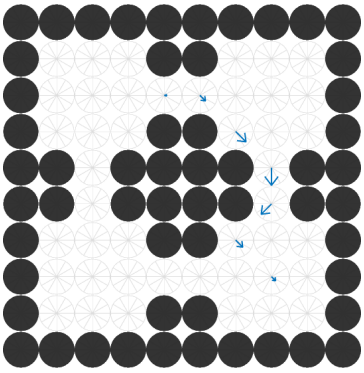
Fig. 9.1 shows some examples of the sequence connectivity that self-organises using this method. We see that these learned sequences are composed of a set of state-action combinations that describe a reasonably logical journey from one area of the map to the other. The learned sequences have a single 'entry' point at the beginning of a sequence and a single 'exit' point at the end of this sequence. This contrasts with an approach that might produce multiple entry points and multiple exit points (see for example [13, 75]). The weight structure is consistent with that described in the previous chapter, with sequence cells receiving synapses only from the last SA cell in a sequence and sending synapses to all SA cells in that sequence.



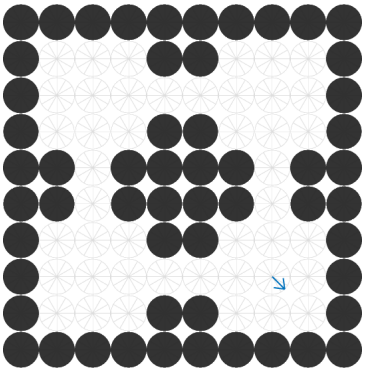
(a) Weights from Sequence Cell 111 to SA Layer



(b) Weights to Sequence Cell 111 from SA Layer



(c) Weights from Sequence Cell 71 to SA Layer



(d) Weights to Sequence Cell 71 from SA Layer

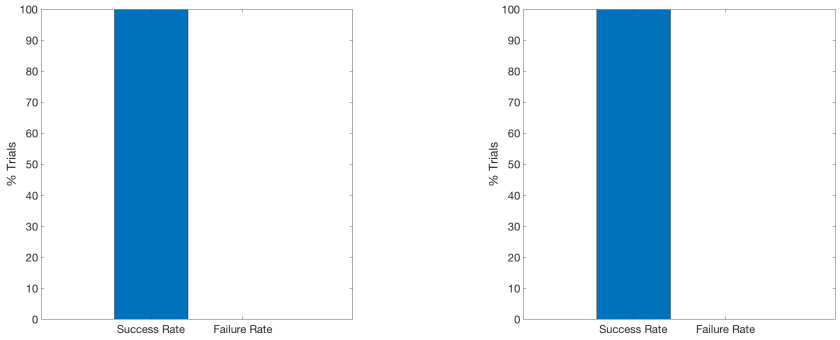
Figure 9.1: Examples of two learned sequences. The magnitude of each arrow represents the weight of the synapse from the sequence cell *to* that SA cell (left column) or the weight of the synapse to the sequence cell *from* that SA cell (right column). Synapses have been normalised so that 1 is the maximum synapse. We see that the weight structure is consistent with that described in the previous chapter, with sequence cells receiving synapses only from the last SA cell in a sequence (b, d) and sending synapses to all SA cells in that sequence (a, c). See Fig. 3.3b for legend.

Fig. 9.2 shows the results of a set of planning experiments using sequences learned in the manner described in this chapter. These tasks were conducted as standard planning tasks: the agent was placed in the 8x8 open environment and the 8x8 4-room environment (Fig. 3.6). In each trial a random location within this grid is designated as the agent's goal and another random location is designated as the agent's starting point. The agent can move one space at a time in eight compass directions[3] or stay still, giving nine possible actions. The task is for the simulated agent to navigate to the goal location. If the agent reaches the goal location then the task has been been completed successfully. If the agent fails to reach the goal location within 100 timesteps then the agent has failed the task. Fig. 9.2 shows that under these conditions the model is able to achieve 100% success rates in both the 8x8 open environment and the 8x8 4-room environment.

Fig. 9.3 shows that states that are occupied more often have more associated sequences. To obtain this figure we count the number of sequences associated with each state by counting the number of synapses that each SA cell receives and then associating them with the state which that SA cell represents. The results strongly suggest that the model tends to learn sequences that are 'useful', in other words sequences that occur frequently in the solutions to planning tasks that take place in this environment. We also see that, at least in the relatively small 8x8 environment, most states are associated with at least one sequence, suggesting that the learned sequences cover the environment reasonably well and so may be useful even in less commonly used areas of the state space.

Fig. 9.4 tests the effect of varying the η parameter (eq. 9.5) on the sequences that are learned. Although η should affect the length of these sequences (by altering the speed at which the memory trace of previous activity decays and hence how many state-action cells are associated by a given sequence cell) it does not appear to do so in practice. Fig. 9.4 shows that the distribution of learned sequences seems to vary very little over different trials with altered η values.

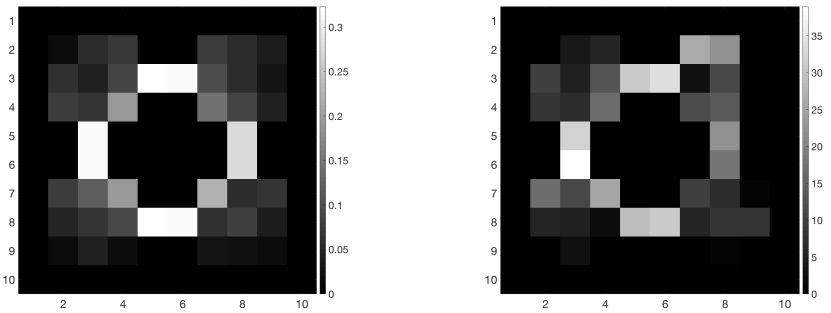
3 N, NE, E, SE, S, SW, W, NW



(a) 8x8 Open Environment

(b) 8x8 4 Room Environment

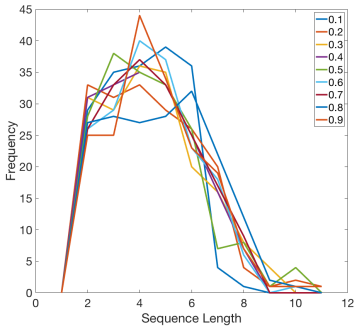
Figure 9.2: Success Rates in Planning Task When Using Learned Sequences. This figure shows the percentage success rates of the network in 100 trials.



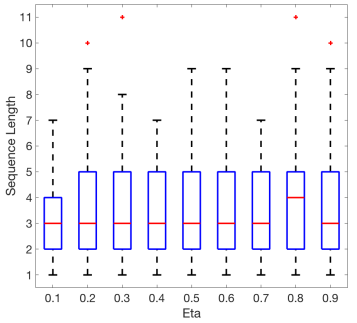
(a) The frequency with which the agent occupies each state during 300 planning tasks. The brightness of a point denotes the probability of an agent occupying that state during a planning tasks. Bright states have a higher probability of being occupied.

(b) This figure shows the number of sequences associated with each state. Brighter states are associated with more sequences.

Figure 9.3: *The occupation of different states during the sequence-learning period.* (a) shows the probability of an agent occupying any given state in this 4-room environment (Fig. 3.6) while (b) shows the number of sequences associated with each state after 300 planning tasks in this environment. By comparing the two, we see that states that are occupied more often have more associated sequences. This strongly suggests that the model tends to learn sequences that are 'useful', in other words sequences that occur frequently in the solutions to planning tasks that take place in this environment.



(a) This figure plots the frequency distribution of sequence lengths as a line plot. Each line represents a value of the parameter η, given in the legend.



(b) This figure plots the frequency distribution of sequence lengths as a blot plot.

Figure 9.4: This figure shows the effect of altering η on the length of the sequences learned by the model. The length of a sequence is calculated as the number of SA cells that receive afferent synaptic connectivity from a given sequence cell with weight greater than 0.0001. (a) shows that altering the value of η has very little effect on the distribution of learned sequence lengths: effectively a Gaussian distribution with a long tail. (b) shows that altering the value of η does not affect the mean or upper/lower quartiles of sequence length unless η is very low, and that the range of learned sequences does not vary significantly either.

9.1.6 *Discussion*

In the previous chapter we demonstrated that the network can use a layer of sequence cells to assist planning, reducing the time it takes for activity to propagate through the SA layer and so reducing the amount of planning time required to reach decisions. There remained, however, the question of how the agent should produce the required connectivity between the sequence cells and SA layer, thus learning useful state-action sequences.

In this chapter we have demonstrated that the network is able to successfully learn state-action sequences which cover the commonly used area of the state-action space. Because the learned state-action sequences tend to be those that occur most frequently during the performance of navigation tasks (Fig. 9.3) they are in the main appropriate for the structure of the environment (e.g., moving from one room to another in a four-room maze). In theory, the length of these sequences can be partially controlled using the trace learning parameter η, although this seems to have little effect in practice (see Fig. 9.4).

The learning procedure described in this chapter requires that trace learning takes place at specific times in each timestep, when the SA cells are representing the appropriate information about combinations of state and action. We hypothesise that the firing of the action cells once planning is complete and the next action has been decided, and the flood of sensory and motor feedback produced by the performance of that action, strongly activate the SA cell representing the current state and action, allowing sequence cells to connect to that SA cell.

The question then arises of why the sequence cells only form synapses to the SA layer at this point in the timestep. It is possible that SA cells fire very strongly at this point, boosted by sensory and motor feedback, while the level of SA activity during the planning process is much weaker. Equally, it is possible that plasticity is gated in some fashion such that the synapses are only plastic at this point in the timestep.

9.1.6.1 *Replication of Human Planning Times*

As humans spend more time in an environment, they will learn a cognitive map but will also begin to learn useful sequences of actions for performing tasks in that environment. In wayfinding experiments, this means that as humans perform more navigation tasks in an environment,[4] they should show faster decision times at choice points. This prediction is difficult to validate, because very few papers on human navigation give planning times and none describe how planning time changes with experience. However, there is circumstantial evidence which supports this account.

In particular, Howard 2014 [57] found that the decision-making time of humans doing a wayfinding task in a virtual reality replication of the London borough of Soho was proportional to the absolute path length between the current choice point and the goal. The correlation was strongly significant, and of medium strength (correlation constant = 0.363; $p < 0.01$). Although the proposed model reproduces this correlation between planning time and path length, in its base state without sequence cells the proposed model gives a much stronger correlation (Fig. 9.5a, correlation constant = 0.88) than is shown in the Howard 2014 data. Introducing the hierarchical mechanism (utilising sequence cells) decreases planning time and introduces more variation, giving results (Fig. 9.5b, correlation constant = 0.69) that are closer to Howard 2014's data. As environments become larger, and the available repertoire of learned sequences becomes larger and more complex, we believe that the effect of the hierarchical mechanism on planning time would be even stronger, and so potentially produce planning-time vs. path length correlations close to those found by Howard et al. 2014.

Figure 9.5 shows that when probabilistic propagation[5] is used (see Sec. 7.2) the proposed model has a very high initial[6] planning time to path length correlation in its base state (Fig. 9.5a) but the hierarch-

4 Particularly if they have already memorised the layout of the environment, as in Howard (2014) [57].

5 All other figures in this section use the propagation mechanism described in Sec. 8.1.4 and 8.2.4.

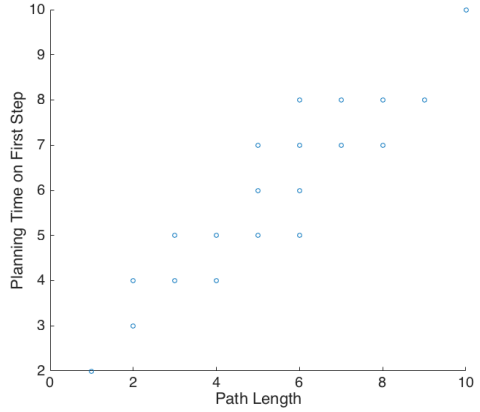
6 Measuring the planning time for the first action only, as opposed to the *total* planning time measured in Fig. 8.23. This is done to match data found in Howard 2014 [57].

ical mechanism decreases planning time and introduces more variation (Fig. 9.5b). In these simulations, the model was presented with 100 tasks (random combinations of goal and starting location). For each task, the planning time (in timesteps) was measured, as was the length of the final route to the goal. This allowed us to check the initial planning time required to produce a route.

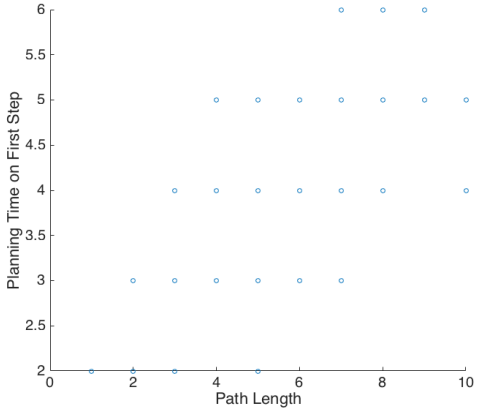
9.1.6.2 *Replicating the Preference for Familiar Routes Under Stress*

Brunye 2017 [120] showed that humans navigating through a large virtual environment rely more on familiar routes when under high time pressure. Specifically, they are more likely to re-use fragments of previous routes when they are given a low time limit for successfully reaching the goal (movement speed cannot be altered), showing that participants are relying on remembered route fragments rather than calculating optimal routes. A potential explanation for this effect is that using remembered route fragments allows for faster planning.

In the previous chapter, we demonstrated that the use of the hierarchical mechanism employing sequence cells produced a tendency towards familiar routes and stereotyped behaviour (Fig. 8.25). We now demonstrate that the proposed model partially reproduces the findings of Brunye et al. using two related simulations (Fig. 9.6). In the first simulation, we demonstrate that using the hierarchical mechanism (sequence cells) increases success rates under conditions of high time pressure and so that real life agents would be incentivised to use such a mechanism when under high time pressure. In the second simulation, we demonstrate that using the hierarchical mechanism causes the model to spend more time using familiar routes. The logical deduction from these two simulations is therefore consistent with the experimental effect observed by Brunye et. al: agents following this model would use the hierarchical mechanism more when under heavy time pressure and this would have the effect of increasing their tendency to reuse familiar routes. In other words, the proposed model explains why participants use the hierarchical mechanism more frequently under conditions of high time pressure, as well as why that leads to the increased reuse of route fragments.



(a) The model run without sequence cells. Pearson correlation coefficient = 0.88



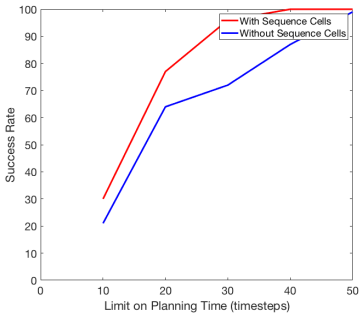
(b) The model run with access to sequence cells encoding a variety of trajectories. Pearson correlation coefficient = 0.69

Figure 9.5: Planning Times for First Step Against Path Length. The proposed model was run using probabilistic propagation for 100 trials in a 10x10 state environment, with and without the hierarchical planning mechanism utilising sequence cells. The hierarchical planning mechanism had previously learned a set of frequently-used sequences using the method described in Sec. 9.1.4. We then plotted the number of timesteps required to produce the initial action against the distance of the goal. The Pearson correlation coefficient is a measure of the strength and direction of the linear relationship between the planning time for the first step and the length of the path. If those two variables are postively correlated to some extent, their correlation will be a positive number between 0 and 1. The magnitude of this number denotes the strength of the correlation. If the model is run without sequence cells (a) initial planning time and path length are strongly correlated, whereas if the model has access to sequence cells the correlation is still present but is considerably lower.

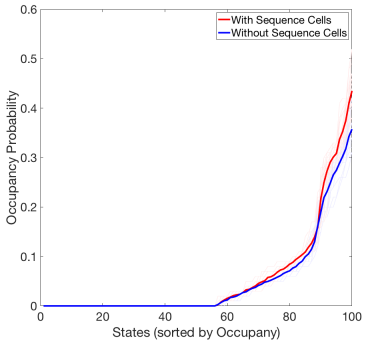
The first simulation, investigating whether use of the hierarchical mechanism increases success rates when under heavy time pressure, is run as follows. The model is run for a number of navigation trials, with new goal and starting locations for each trial. To mimic the time pressure that the participants in Brunye's experiments were put under, these trials are terminated after a certain number of timesteps; if the model has reached the goal by this point then it has succeeded at the task and if not then it has failed. An earlier cutoff therefore corresponds to a higher time pressure. In total, the model was run for 100 trials in each time pressure condition and we measured the corresponding percentage success rate. We then repeated this procedure but removing connectivity between the SA cells and the sequence cells. The results of this experiment are shown in Figure 9.6a and confirm that using the hierarchical mechanism increases success rates under conditions of high time pressure.

Having demonstrated that agents following this model would be incentivised to use the hierarchical mechanism when under heavy time pressure, we now demonstrate that using this mechanism increases the degree of overlap between task solutions (see Fig. 9.6b). The agent is again run for 100 simulated trials with connectivity between the SA and sequence cells, and 100 trials without that connectivity. Goals and start locations are chosen randomly for each new trial.

In this experiment, we calculate the degree of overlap between task solutions by measuring the amount of time the agent spends in each state. We expect that, if the agent is reusing familiar routes, the agent will spend more time in fewer states than if the agent is planning optimal routes for each task. We therefore measure the *occupancy* of each state, i.e., the probability that the agent will pass through a given state in a trial. Figure 9.6b shows that more states show higher occupancy when the hierarchical mechanism is active than when it is not.



(a) Success rate against time pressure, with and without the hierarchical mechanism incorporating sequence cells.



(b) The occupancy of states during a 4-room maze planning task with and without sequence cells. Light blue and red lines show the probability of occupying each state during 20 blocks of 100 trials in each condition. We see that in the hierarchical condition, the 10% most occupied states are occupied more often than the 10% most occupied states in the non-hierarchical condition (these are not necessarily the same states). The effect is small, likely due to the small size of 8x8 environment which gives the agent little choice of routes to the goal, but statistically significant (two-sample t-test, $p<0.001$). This suggests that the hierarchical mechanism produces more stereotyped behaviour, causing the model to inhabit the same states more often.

Figure 9.6: (a) shows the percentage success rate—the likelihood that an agent will reach the goal—if processing is terminated after a certain number of timesteps. The model was run for 100 trials in a 10x10 state maze, with and without the hierarchical mechanism incorporating sequence cells. Fig. 9.6b shows the distribution of the probabilities of occupying different states, with and without the hierarchical mechanism. As found in Brunye 2017 [120], the use of a hierarchical mechanism leads to faster (Fig. 9.6a) but more stereotyped behaviour (Fig. 9.6b).

10

DISCUSSION

Early in this thesis (Ch. 2) we gave a list of aims for the work that we have described: to produce a model that formed and used a cognitive map, to remain consistent with known neurobiology as far as possible, to address flaws in previous research and to investigate the use of hierarchical planning mechanisms. Having introduced the relevant concepts (Ch. 1), and reviewed previous work in the field (Ch. 2), we then produced a series of simulations designed to address these aims (Chs. 3 to 9). In this chapter, we will discuss this work in the context of each of our original aims.

For the sake of simplicity, the models reviewed in Ch. 2 will be referred to simply as "Cuperlier 2007", "Hasselmo 2005" etc. for the rest of the chapter. Full citations are given in Section 2.2.

The remainder of this introduction will explain the approach that we are taking to discuss the work proposed in this thesis. Section 10.1 will then recap the aims of this work. Sections 10.2 to 10.4 will address each of these aims in turn, discussing how previous work has addressed these aims, how the model proposed in this thesis has built on that work, and how further modelling and experimental research could extend and test the theories that we propose.

The relationship between modelling, experimental work and empirical knowledge is complex. Not only are there different types of modelling with different goals (one framework proposes that *descriptive modelling* describes what a system does, *mechanistic modelling* describes how a system works, and *interpretive modelling* explores why the system's properties are desirable [101]) but the level of empiricism that should be applied to a piece of modelling work is debatable. Theoretically, a model should produce clear, falsifiable predictions on which it can be judged. A model whose predictions are not falsified is retained, a model whose predictions are falsified is scrapped or altered. This rigorous approach is important: it prevents modelling from becoming disconnected to the work that is being modelled and forces modellers to produce work that is clear and of use to research-

ers. In the context of a doctoral thesis, such work is an original contribution to the field.

However, the application of too much rigour too soon does not fit the actual process of producing models. It can take many iterations of a model, and much work in the field as a whole, to produce a key paper that can conclusively test (and potentially falsify) a paradigm and the associated set of models. As such, the papers reviewed in Ch. 2 as well as the model that is presented in this thesis should be considered as works that progress towards this end goal. They can replicate certain experimental results and produce predictions, but only to a certain extent. Some aspects of these models are clearly false.

So how can we judge them? One approach is simply to accept that modelling is in practice creative and fallible. It can be argued that any exploration of a system is automatically an extension of knowledge and will, to some extent, be of use to further modelling efforts. Even a model that is manifestly false may demonstrate mechanisms that form the core of a later, empirically valid model. Even model-free reinforcement learning, an incredibly powerful family of models that has been the subject of intensive research for more than a decade, at present fails to solve a variety of tasks. These include contingency change tasks [11], tasks which require credit assignment over long periods [35], tasks which require robust transfer of skills from one subset of tasks to another [42], and more. Is this because the model-free reinforcement learning hypothesis is fundamentally flawed, or simply because it has not yet been elaborated sufficiently? Or does it form one part of a larger solution? However, an all-embracing view of modelling lacks rigour, and if taken to an extreme it denies that good and bad models exist at all.

If we accept that models are to some extent works in progress towards a truly empirically useful model, there seem to be two primary characteristics that can be judged: firstly, on the desired model that is envisioned at the end of the procedure, and secondly on the progress that is made towards that ideal model.

For each aim, we will therefore ask: to what extent has this work addressed that aim, in comparison to other models? And how does this work produce falsifiable predictions or could do so in the future?

10.1 THE AIMS OF THE PROPOSED MODEL

The core aim of this project has been to investigate the brain mechanisms underpinning cognitive-map based behaviour. To do so, neural networks were developed with the following properties:

1. The proposed network should learn a causal model of the simulated agent's environment purely through unsupervised sensory exploration. The network should then be able to use this causal model to guide intelligent sequential behaviour such as navigation towards a specified state or location. Once this environmental model is learned, the proposed network should be able to produce a novel route from any randomly chosen state to any other randomly chosen state.

2. The architecture and operation of the proposed network should remain broadly consistent with known neurobiology. In particular, it should rely on local Hebbian learning to model synaptic plasticity between neurons.[1] It should also self-organise all of its connectivity from sensory and motor inputs as the simulated agent explores its environment, rather than rely on pre-existing hardwired connectivity or the manual alteration of specified synapses using meta-knowledge of decoded cell representations.[2]

3. The proposed network should achieve aims 1 and 2 while addressing outstanding problems raised by earlier neurobiological models of these functions.[3]

1 The proposed model is rate-coded, and so Hebbian learning was used as a time-averaged approximation of Spike-timing-dependent plasticity (STDP).

2 For example, Friedrich 2016 calculate target synaptic weights from observed state probabilities and update weights to minimise the difference between the actual and desired weights.

3 Discussed further in the review of each model (Ch. 2).

4. Extensions of the network should learn a hierarchy of motor/behaviour primitives, similar to that described by work on hierarchical cognitive maps in humans [49–51], which would allow the model to reproduce important elements of hierarchical map-based planning behaviour [52, 54, 55]. All of the neural mechanisms for implementing this behaviour should self-organise in a biologically plausible manner according to aim 2.

In Sec. 10.2, we look at the model from the perspective of Aim 1: the mechanistic problem of learning and using a cognitive map in a neural system. We consider how previous work has investigated this problem, and identify some open questions and areas of contention that remain within this topic. In particular, we consider the various formulations of cognitive map that have been proposed by previous models, the mechanisms that have been proposed for planning using these different maps, and the mechanisms that have been proposed for supporting and extending the planning process. We described the mechanisms of the model by this thesis in terms of these questions, and show where they provide additional hypotheses and how these hypotheses could potentially be tested.

In Sec. 10.3, we look at the model from the perspective of Aim 2: producing neural models that remain consistent with the known neurobiology. This necessitates a series of compromises: our guidelines for producing a biologically plausible model have to be stringent enough to be meaningful but not so stringent that they make meaningful progress towards a working model impossible. We have therefore drawn up a set of guidelines that we have tried to adhere to in the process of producing the model proposed by this thesis, and a further set of conditions that we have relaxed for the course of this work. We consider previous work from this perspective, considering in particular the relationship of between the learning and planning modes that exist in current models, the mechanisms that have been proposed to self-organise the neural substrate that encodes the cognitive map and the mechanisms that have been proposed to encode a cognitive map within that substrate. We then describe the self-organisation mechanisms that we have proposed in this thesis in the context of previous

work, and discuss how these elements could be tested in experimental work. Note that both Sec. 10.2 and Sec. 10.3 consider the proposed model in the context of previous work (from different perspectives) and so Aim 3 is effectively covered within these sections.

In Sec. 10.4, we look at the problem of hierarchical map representations and planning. We define and motivate this problem, discussing the regional representation of space and the production of high-level route descriptions. We discuss the variable-resolution map mechanism proposed by Martinet 2011 [2], which proposes the existence of an extra, simplified cognitive map that merges together states according to a proprioceptive signal and uses this to extend the range of their model's planning mechanism. We then propose an alternative mechanism that learns frequently-used sequences of state-action combinations, similar to high-level route descriptions, and uses them to speed up planning. We compare these two approaches and discuss how they could be further investigated, both in terms of modelling work and in terms of experimental work.

10.2 AIM 1: LEARN AND USE A CAUSAL COGNITIVE MAP

10.2.1 *Definition and Motivation*

After considerable study over the last few years, the production of a cognitive map and the use of it for consequent/concurrent model-based planning seems to be a necessary element for the production of some behaviours (see Ch. 1). Such maps appear to be particularly important in the context of revaluation and contingency-change tasks [11, 12].

Given the apparent importance of this mechanism, one of the primary aims of this work was to investigate this process in the context of a (biologically plausible) neural network and there are two fundamental elements that therefore had to be replicated: the ability to learn a causal model of the simulated agent's environment, and the ability to use this model to generate behaviour. The other aims of this work—to remain broadly consistent with known neurobiology, to ad-

dress outstanding questions raised by previous models of this mechanism and to extend this mechanism to include hierarchical mechanisms—rest on the prerequisite that the model is ultimately able to produce the desired behaviour, or at least represents a meaningful step towards such a model.

10.2.2 *Progress Made By Previous Work*

All of the models reviewed in Ch. 2 were chosen because they were able to successfully form a cognitive map of a simple environment and use that map to produce paths between randomly chosen states.

10.2.2.1 *Form of the Cognitive Map*

The cognitive maps that these models form vary—they are encoded in different types of cells and using different recurrent synaptic connectivity. In this section we will examine these cognitive map representations from a mechanistic point of view, exploring how they differ and what predictions they consequently produce. In Sec. 10.3 we will explore these representations from a modelling fidelity point of view, exploring how closely they fit the constraints of known neurobiology and how they could potentially be self-organised using synaptic plasticity.

STATE (PLACE) CELLS One form of cognitive map reviewed in Ch. 2 encodes the topology of the environment using state (place) cells. These place cells are connected together by recurrent synaptic connectivity in such a way that each state is connected to its neighbouring states. This is probably the most simple form of cognitive map that we reviewed, and Matsumoto 2011 shows that it is possible to use this map to plan (using a mechanism described below). However, this form of map does not encode any information on the actions that are required to move the agent from one state to another. Planning in this context can therefore only mean assigning value to different states, marking some as more desirable and some as less desirable. This form of cognitive map therefore implicitly predicts that

some other, unmodelled, mechanism must exist that knows how to move the agent from one state to another. In other words, this form of cognitive map does not reproduce an extremely important part of the planning mechanism.

STATE-ACTION CELLS Another form of cognitive map, used by Friedrich 2016, is based on state-action cells. These cells encode a combination of a particular state with a particular action taken in that state. They are connected together with recurrent synaptic connectivity that encodes a causal relationship. If a cell encodes a state-action combination that produces a transition to a particular successor state, that cell will be synaptically connected to other cells encoding the successor state. This means that a state-action map can encode not only how different states are connected but also what actions an agent should take to move between these states.

TRANSITION CELLS A variation on the state-action map is the transition map used by Cuperlier 2007, which uses several different kinds of transition cells to encode the cognitive map. Some of these cells encode a unique transition between two states, without encoding the action that produces that transition, and these cells effectively work in the same way as state cells. Other cells encode a combination of a given transition and a motor action that produces that transition and so end up resembling state-action cells. A problem with this form of map representation is that it does not seem likely to generalise well to nondeterministic environments. As described in Sec. 2.3 (eqs. 2.1 to 2.5), in certain environments, a single state-action combination may produce many state transitions. In a state-action model, these transitions may all be represented using recurrent synaptic weights. By contrast, in a transition cell model each transition requires its own cell. As the ratio of transitions to state-action combinations increases, the ratio of required transition cells to required state-action cells increases.

MINICOLUMNS More complex cognitive maps (Hasselmo 2005, Martinet 2011) still fundamentally use a state-action encoding for the cognitive map but incorporate these cells within complex minicolumn structures designed to facilitate various elements of the planning process. These structures are hard to describe in detail (see Ch. 2) but the cells within them tend to correspond to state cells, (motor) action cells and state-action cells. Functionally, minicolumns allow for much more detailed circuitry to be easily modelled and iterated across the cognitive map representation. This makes minicolumn models more powerful than models using other representations but also more complex and less clear.

The form of an agent's cognitive map will inevitably affect the mechanisms that are required to plan using this map. From this perspective, the most important distinction between the various forms of maps is between those maps that incorporate actions implicitly (a pure state cell map) and those that incorporate actions explicitly within the cognitive map (all others). A pure state cell map inherently predicts that planning takes place at the state level, and that some other mechanism exists which encodes the causal knowledge of how to move from one state to another and that outputs actions to do so. In other word, a pure state model predicts that planning and acting are two fundamentally separate processes, which output a set of desired transitions and a set of actions respectively. By contrast, a state-action representation predicts that these two processes are inherently interlinked. This claim could be explored experimentally by looking for cases (if they exist) where people appear to be able to plan a solution to a problem (i.e., to give a set of transitions between states) but are not able to give a set of actions which would perform those transitions. This would demonstrate that it is possible to disrupt the action-production process without disrupting the planning process. The existence or nonexistence of such cases might support a state-based or state-action based account.

Another open question is how these representations should scale. Of the models cited, all of them except for Matsumoto 2011 and

Friedrich 2016 are discrete: they assume that states and actions are separable from one another. This has two consequences. The first is that in many cases the associated cognitive maps encode an artificial separation between different states and different actions. Either they are described purely in the context of tasks with discrete state spaces (such as the grid word tasks used throughout this thesis, where you are either in one grid square or another grid square), or the models designate certain points in a continuous state space as 'states' and ignores all of the points in between. The same is true of actions, where a limited set of discrete actions is made available for the agent to choose between. The second consequence of using a discrete state space—at least in the reviewed models—is that they do not handle scaling well. As the size of the environment increases, either the number of discrete states must increase or the ability of the map to encode the state space must decrease. For this reason, the models reviewed in Ch. 2 all have small state spaces.

The alternative is a continuous representation, where each input neuron represents some form of function in the joint space of states and/or actions (e.g., a place cell with a gaussian receptive field). In this case states and/or actions can be represented as a unique combination of input neuron firing. Two models deal with such continuous representations: Matsumoto 2011 and Friedrich 2016, but these models are also the most limited in terms of explaining the mechanisms around the formation and use of these maps: Matsumoto 2011 does not address the encoding of causal relationships between states, Friedrich does not address the formation of the cognitive map, and neither of them model the process of reading out the results of the planning process using a neural mechanisms. It seems likely that the complexity of modelling continuous representations is responsible for the more limited scope of these models.

10.2.2.2 *Planning Mechanisms*

THE PROPAGATION OF DECAYING ACTIVATION The planning mechanisms used by these models are all to some extent based on the

principle of propagating activity from a goal representation through the synaptically encoded cognitive map.

The basic paradigm is that of propagation with decaying activation. In this paradigm, the cells encoding the cognitive map are stimulated at one location, or several locations, encoding the goal state that the agent is required to navigate to using its cognitive map. This activity propagates imperfectly through the cells encoding the cognitive map, such that cells close to the goal receive high activity and fire at a high rate, while cells further from the goal receive much less activity and fire at a much lower rate (Fig. 2.10). In effect, a gradient of activity is created with the cells that represent the goal state(s) have the highest firing rate and the state cells or state-action cells that are far away in the map (separated by many transitions) having very low or no firing.

If the map is made up of state cells, as in Matsumoto 2011, this gradient effectively encodes a value-function: states represented by high-firing cells are highly valuable, states represented by low-firing cells are not valuable.

If the map is made up of state-action cells, as in Friedrich 2016, this gradient effectively encodes a Q-function: state-action combinations represented by high-firing cells are valuable, state-action combinations represented by low-firing cells are not valuable.

The decaying-activation paradigm allows successful planning. In particular, Martinet 2011 show that a decaying-activation model is able to reproduce the characteristic behaviour of rats performing a Tolman detour task [2, 21]. It also reproduces the latent learning phenomenon, in which rats which have previously been allowed to explore a maze in the absence of explicit external reward are able to successfully navigate to a reward in that maze much faster than rats which lack this experience [5].

EXTENDED PLANNING MECHANISMS Some of the reviewed models[4] use the decaying-activation paradigm roughly as described, but others extend it in various ways. Matsumoto 2011 uses the decaying activation to form synapses between the place cells that make up the

4 Cuperlier 2007, Hasselmo 2005 and Friedrich 2016.

cognitive map and a "goal-place" cell in another layer. The firing rate of these place cells encodes their value, and the strength of the synapses to these states corresponds to this firing rate. These synapses therefore encodes the value of different states when navigating to a specific goal. Essentially, this mechanism caches the value function produced by the decaying-activation mechanism, presumably allowing the task to be done more quickly in future.

Erdem & Hasselmo 2012 uses a system of "linear probes" to allow the model to sample the value function produced by the decaying-activation function. This system samples the value gradient (produced by the decaying-activation mechanism) in straight lines away from the agent, as shown in Fig. 2.8. The agent then moves in the direction of highest value. If a shortcut exists such that a higher-value of the map can be reached by moving in a straight line through unexplored space, this will be discovered by the linear probe, allowing the agent to use this novel shortcut. Finally, Martinet 2011 implement a variable resolution method, which extends the distance that activity can propagate before it fully decays. This will be discussed further in Sec. 10.4.

Although a decaying-activation planner can reproduce basic planning behaviour, as well as replicate seminal tasks such as the Tolman detour task, the behaviour of a decaying-activation planner does not appear to match experimental observations in certain ways. The first and most important of these is that if the goal is sufficiently distant, and a sufficiently large number of actions are required to reach it, then the decaying activity will decay to zero before it reaches the agent's current state. Since it is (the gradient of) this activity that defines either a value function or a q-function for each state, if no activity reaches a state or set of states then the agent has no value information available about those states. The agent will therefore not be able to produce a useful action. A decaying-activation planner will therefore fail to produce appropriate actions if the distance to the goal is too great. This does not seem to be consistent with evidence that animals can plan and navigate in large-scale environments [4].

A second issue with planning using decaying activation is that it may be vulnerable to noise. If the gradient of decaying activity spreads over a large number of states, then the relative firing rates of cells representing different states (or different actions within the same state) are likely to be similar. The firing rates of these cells are used to indicate the relative values of the associated states or state-actions; if they are very similar then small changes due to noise may make a big difference to their relative values and cause very different actions to be selected. Perhaps significantly, the models reviewed in Ch. 3 do not investigate the effect of noise on activity propagation.[5]

Some of the reviewed models also suggest that a decaying-activation mechanism may take a relatively long time to output actions. Cuperlier 2007 reads out an action only when the activity in the network is considered 'stable', which presumably requires either a set period of stabilisation or a mechanism that judges the stability of the activity dynamics. Hasselmo 2005 waits for a specific period of time before reading out an action and thereby risks reading out too early (before activity has reached the right part of the map) or too late (when the situation has changed or time has been wasted).

Erdem & Hasselmo 2012 raise another problem: that any theory of planning using the propagation of activity through a map predicts that an agent will only be able to navigate *to* states that it has previously experienced, *through* states that it has previously experienced. Experimental evidence exists that seems to show that if animals which have sufficient experience in an environment are provided with a novel shortcut to the goal that uses novel states, they will preferentially use this shortcut without prior exploration [5]. It should be noted, however, that the extensive sensory apparatus of rats make the distinction between novel states and experienced states somewhat more blurry than in the models we review.

As described above Erdem & Hasselmo 2012 present an extension that may account for shortcut-using behaviour: planning using a combination of decaying activation and linear probes. At present, this is

5 Erdem & Hasselmo 2012 conduct experiments with noise, but only in other parts of the model.

the only model we review that is able to use novel shortcuts during planning with a cognitive map. However, the linear probe mechanism as given in that model has significant disadvantages. Firstly, it predicts extremely maladaptive behaviour when navigating around obstacles. The linear probe mechanism currently has no means to detect when a probe passes through an obstacle, and so the agent will choose a path that is blocked by an obstacle if there is a high-value region on the other side (see Fig. 2.8). In other words, for this planning mechanism to work there must be a system which constantly regulates the length and direction of linear probes to avoid such incidents. It is difficult to imagine how this mechanism would work as a neural mechanism, particularly with the limited information available to the agent.

10.2.3 *Progress Made By This Thesis*

In the previous section we highlighted several problems that have not yet been fully solved by previous models. In other words, areas where a mechanism has not yet been found to solve a problem with forming a cognitive map or planning with the cognitive map, or where that mechanism has important limitations. The most important of these areas is the use of a decaying-activation planning mechanism, which limits the size of the problem that can be solved.

The model described in this thesis had to replicate two primary elements of model-based planning: the production of a cognitive map and the use of this map to produce behaviour. In the process of doing so, we attempted to address the limitations of the decaying-activation theory by using an alternative planning mechanism.

Chapter 3 demonstrated that a particular network architecture was capable of producing solutions to grid world tasks, so long as the model was provided with a valid cognitive map that described the state transitions that were possible in each grid world. This map took the form of recurrent connectivity within a layer of state-action (SA) cells. The rest of the network supported the planning process by ensuring:

- That activity entered the cognitive map at the right point(s)
- That this activity propagated through the SA cell layer in a manner determined by the form of the cognitive map (as encoded in the recurrent connectivity between these cells).
- That the output of the planning process was read at the correct time and from the correct part of the SA cell layer so as to produce an optimal action for the agent's current state.

Chapter 7 extended the network described by Ch. 3 to show:

- That the same network architecture was capable of altering its cognitive map in response to changes in the environment.
- That this architecture could encode the approximate probability of various state transitions after taking a given action in a given state, and could plan so as to take these probabilities into account.
- That the model could replicate the behaviour of rats in a Tolman detour task, which is still considered to be solvable only by using a cognitive map approach [11].

In conjunction with this work, which is described at greater length below, Chapters 4, 5 and 6 demonstrate that this network architecture, including the cognitive map itself, the cellular substrate required to support this map, and the mechanism for correctly reading out the results of the planning process, can self-organise itself purely from sensory and motor inputs as the simulated agent explores its environment.

The work described in this thesis therefore fulfils the two basic requirements necessary to achieve Aim 1: the proposed network architecture is able to learn a cognitive map and to use it to produce model-based behaviour. However, several limitations apply.

DISCRETE STATE REPRESENTATIONS Firstly, the network architecture described in this thesis was only able to learn and plan within discrete (grid world) environments. A version of the model was tested

in continuous environments, and initial experiments suggested that the model was able to self-organise SA cell responses in continuous environments, as well as to learn at least a partial SA map in the recurrent connectivity between these cells, but due to time constraints this work has not yet progressed far enough to produce rigorous results for publication. We have not yet investigated planning through a continuous state space.

A consequence of exclusively modelling learning and planning in discrete grid worlds is that the number of discrete states required to encode an environment increases rapidly with the size and complexity of that environment. Figure 10.1 shows how the required size of the network increases as the environment expands. The maximum environment size we were able to simulate was therefore 20x20, with 9 available actions. This limited the size of the environments that we could simulate directly, and forced us to extrapolate the model's behaviour to larger environments. However, the trend shown in Fig. 10.1 assumes that the network should fully encode an environment. This is likely to be unnecessary, as the model can navigate successfully after learning a very small proportion of the available state-action combinations in an environment (Fig. 6.8). In practice, it may be that a very small proportion of the states and state transitions in an environment can encode almost all of the useful information about that environment. This hypothesis is supported by the success of deep reinforcement learning networks (Ch. 1) which show that networks can use backpropagation to develop very efficient representations of large state spaces, as well as by Fig. 4.9 which shows that the model can successfully learn a subset of states when the number of SA cells is smaller than the number of state-action transitions experienced.

PLANNING USING A PROPAGATING WAVEFRONT An advance of the work in this thesis has been to give a full account of how a propagating wavefront mechanism would work as the basis for planning using a cognitive map. The reason for doing so is that such a mechanism is more efficient, more robust and more powerful than the decaying-activation mechanism used by previous models.

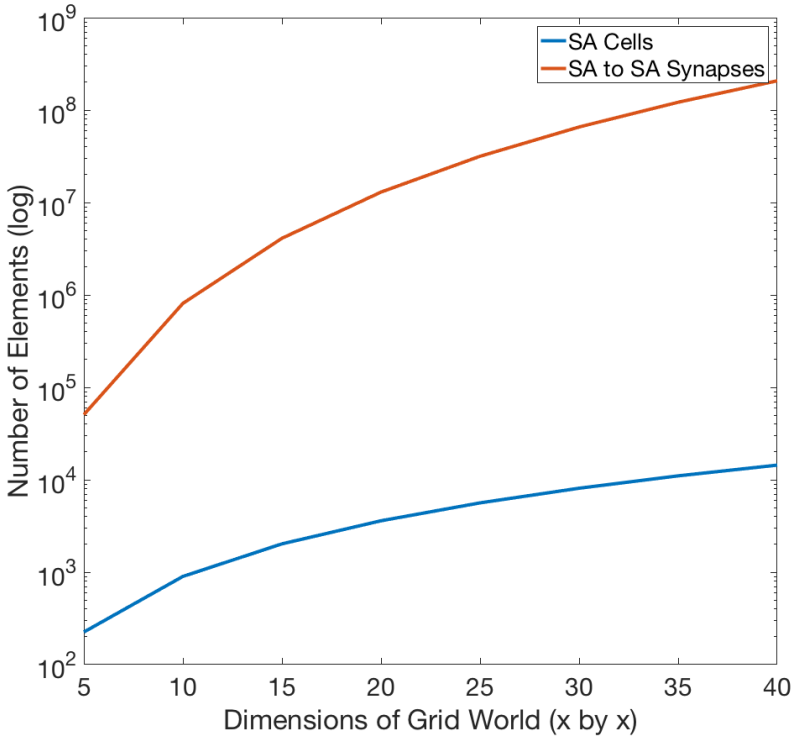


Figure 10.1: The Size of Network Required to Learn Environments of Different Sizes. The x-axis shows the dimensions of the grid world (x by x) such that the total number of states in an environment is x^2. For example, a 5x5 grid world contains 25 unique states. The y-axis shows a log function of the number of SA cells (blue) and recurrent synaptic weights between SA cells (red) that are required to encode an open-field environment with the specified dimensions. We see that as the size of an open-field grid world environment increases, the number of cells and synapses required to fully encode the environment increase rapidly.

As described in Ch. 3, a propagating wavefront mechanism fundamentally relies on the timing of goal-based activity propagation through the cognitive map, rather than how much activity different cells receive. Rather than hypothesising that activity decays as it propagates (which leads to the aforementioned problem that this activity may eventually decay to nothing) we hypothesise that activity propagates along the shortest path to the goal fastest. In other words, we hypothesise that optimal actions are driven by the timing of activation rather than the level of activation. The *first* SA cell to receive activation in each state is considered the optimal action to take in each state, and its level of activation is not relevant.

This mechanism owes inspiration to work by Ponulak & Hopfield 2013 [3]. Their paper described a model in which a wave of activity propagates through a 2D layer of topologically organised pure state cells (neighbouring state cells were recurrently connected) and showed that this wave carried information about the direction of the goal. Specifically, if the goal is east of a state cell, the wavefront will 'hit' the state cell from that direction. Ponulak showed that an anti-STDP mechanism could record this information in the recurrent synaptic connectivity of the layer by strengthening the synapses between cells to indicate the direction from which they were 'hit' by the wave.

The mechanism that we have proposed retains the insight that the propagation of a wavefront can carry information about the direction of a goal, but uses this information in a rather different way. Rather than use a propagating wavefront to adjust the synapses between pure state cells and thus produce a synaptic vector field, we have proposed that a propagating wavefront can identify and output the most valuable state-action combination for a given state.

The proposed mechanism is therefore able to improve on the mechanism described by Ponulak & Hopfield in several key ways. Firstly, because the proposed planning mechanism is able to work in a state-action map, it is able to output explicit actions that move the agent towards the goal (see below). Secondly, because the proposed planning mechanism is able to perform planning without requiring a period of synaptic plasticity, it seems likely that an agent using this mech-

anism would be able to plan more quickly, and would not have to 'undo' the new synaptic weights if there is a change in the goal state or the transition structure of the environment. Thirdly, because the proposed planning mechanism is able to plan without altering the synaptic weights that encode the cognitive map, we can store information in these weights. We show in Chapter 7 that this ability allows the model to store nondeterministic transition probabilities, and to use this information during planning. Finally, we show in Chapter 8 that the proposed planning mechanism is able to interface with a hierarchical behaviour mechanism.

The proposed propagating wavefront mechanism will produce a suggested action as soon as the propagating goal-based activity reaches the agent's current state, and it is therefore the fastest way of producing an action using propagating activity. It is, after all, impossible to read the value of decaying activation at the agent's current state until the neurons representing actions at that state become active. And the gating mechanism that we describe provides a way of ensuring that the first active neuron in the current state is detected and the appropriate action read out. By contrast, Cuperlier 2007 reads out an action only when the activity in the network is considered 'stable', which presumably requires a significant further period of stabilisation; Hasselmo 2005 waits for a specific period of time before reading out an action and thereby risks reading out too early (before activity has reached the right part of the map) or too late (when the situation has changed or time has been wasted).

Mechanistic efficiency does not, of course, mean that the theory is true, and thoughts on testing the contrasting predictions made by decaying-activation theory and the propagating wavefront theory based on timing are given in the next section (10.2.4). However, the prediction made by the propagating wavefront mechanism that planning times will be dependent on the distance of the goal can potentially by used to compare it with other theories.

The propagating wavefront mechanism also seems likely to be more robust to noise. The decaying-activation method relies on comparing the precise value of activation over different states or state-actions,

and if noise alters these values a decaying-activation mechanism may therefore produce maladaptive results. By contrast, the precise effect of noise on a propagating wavefront model is likely to depend on how it interacts with inhibitory mechanisms (Sec. 3.2 demonstrates the effect of noise on the proposed model) but the effect of noise is likely to be much lower. Again, we consider this further in Sec. 10.2.4.

Finally, the propagating wavefront mechanism is able to perform longer and more complex tasks than a decaying-activation mechanism. The level of activity in cells is not relevant, and the activity is not required to decay, so the level of activity can be kept high and the propagation of activity can continue indefinitely.

EXTENDING THE PROPAGATING WAVEFRONT MECHANISM As stated previously, we intended to give a full explanation of how planning with propagating wavefronts could work. To do this, we needed to show not only how the actual propagation would occur, but also to explore the mechanisms that would be needed to support it. In particular, the use of a propagating wavefront mechanism adds new complexity to the problem of how to read out actions from the planning mechanism, and how to plan when the transition structure of the environment is not deterministic.

In a decaying-activation paradigm, action readout must solve two problems: it must read out only from the agent's current state, and it must read out the correct action for that state. So long as the correct action is read from the correct state, the timing of action readout is not terribly important provided that it does not happen too early, i.e., before the goal signal reaches the state/SA cells encoding the current state. In a propagating wavefront model, however, there is a further element: the timing of action readout cannot be too early, and also cannot be too late, or the wrong action will be read out. We show that a layer of gating cells under heavy inhibition, similar to that used by Hasselmo 2005 and Martinet 2011, can be a viable solution in this case.

Another important problem when using a propagating wavefront mechanism is that it is less obvious how to perform planning when

the transitions between states are not deterministic. In the decaying-activation paradigm, it is possible to modulate the rate of decay to suggest different transition probabilities. If a particular transition is highly likely to occur following a particular action taken in a particular state, activity propagating along that transition can decay less, while activity propagating along another, less certain, transition can decay more. However, this mechanism cannot work in a propagating wavefront model because the level of activation (and thus the rate of its decay) conveys no information to the network.

Instead, the relevant factor in a propagating wavefront model is the time that an activation is received, and therefore it seems appropriate to indicate transition probabilities by modulating the speed of activation propagation. Specifically, we propose the probabilistic propagation mechanism (Fig. 7.2). The mechanism theorises that a given synapse (encoding a given transition) has a particular probability of passing on, or failing to pass on, activation at any given timestep. This probability is proportional to the weight of the synapse in question. The resultant prediction is that planning in more uncertain environments is likely to take longer than planning in an equivalent environment when transitions are deterministic.

10.2.4 *Predictions and Falsification*

The model that we have proposed predicts that certain tasks can be solved by model-based planning mechanisms, implemented by a specific set of neural mechanisms. By studying the model, we can elaborate on this prediction, highlighting areas in which this model can be compared to other models and to experimental data, and therefore in which aspects of the model can be improved or discarded.

10.2.4.1 *Theory-Based Predictions*

To begin with, the model produces theory-based predictions. The model described in this thesis is essentially a form of model-based reinforcement learning that has been implemented in a broadly biologically plausible neural network. It learns a Markov Decision Pro-

cess and then solves it to produce a set of actions that will move the agent from one randomly chosen state to another. This means that the model, like the models reviewed in Ch. 2, inherits many predictions and falsifications from Markovian problem solving in general and from model-based reinforcement learning more specifically.

For example, the Markovian decision process as usually formulated assumes that the transition from one state s to a successor state s' after taking action a, even if that transition is stochastic, depends purely on the value of s and a at the current time. Because of this, Markovian approaches struggle to replicate planning in tasks that require looping through a set of states, and struggle to replicate planning in tasks where different states appear to be identical to the sensory system (perceptual aliasing). In more complex environments the state must be deduced rather than defined: for example, deducing an agent's velocity from visual data.

The model therefore wrongly predicts failure in such tasks and environments, because its Markovian formulation is maladaptive in these areas. At present, the model uses a competitive network to select its current state; this structure makes it resistant to sensory noise and allows it to 'fill in' a partially observed state, but does not include an explicit and persistent probabilistic representation of state. This means that it cannot include uncertainty as part of its state, nor can it deduce its state over time as it accumulates evidence. However, there are a range of techniques that in practice allow reinforcement learning to solve tasks with these kinds of characteristics which are usually known as partially observable Markov Decision Processes (POMDPs). Such tasks are frequently considered as a Bayesian estimation problem [85]. It may be possible to incorporate techniques used to solve POMDPs into a future version of the model—for example, by including a state-estimation mechanism prior to the state cells used in the model. This mechanism could then activate the state cells to represent the most likely state, leaving the rest of the model essentially unchanged.

As stated previously, the work proposed in this thesis specifically studies model-based planning as applied to grid world and detour

tasks. The debate on whether or not such an approach is truly responsible for producing the behaviour that has been observed is still ongoing. There appear to be very few behaviours that can be replicated with a model-based reinforcement learning approach but not with a model-free approach, especially if the model-free planner is augmented using a successor representation. The most clear-cut example appears to be contingency change tasks, and the Tolman detour task [11, 12] in particular, as well as tasks in which an animal correctly predicts that a newly-revealed shortcut will lead it to its goal faster than the indirect path that it has followed until now.

However, even these tasks are subject to debate. Russek et al. 2016 point out that in experimental (animal) work, constraints on task procedure frequently apply that might allow animals to perform these tasks in a model-free fashion even though such a performance is impossible in simulation [11]. Furthermore, it is difficult to entirely account for an animal's sensory and predictive capabilities. On the opposite side of the debate, it may be that model-based approaches are discovered to be capable of solving many problems that are currently impossible with either reinforcement learning method, and that model-free approaches are not able to replicate this behaviour.

10.2.4.2 *Neural Predictions*

Because the model implements model-based reinforcement learning using neural networks, the neural mechanisms make neural predictions beyond the theory-based predictions above. On the planning side, we have hypothesised that model-based reinforcement learning is implemented in the brain by a neural network that encodes an environment. Furthermore, this implementation is carried out by activity which propagates through the cognitive map representation and indicates the optimal action for the current state, which is read out and produces action.

One way that this model, or a more advanced version of this model, could be falsified is if further inquiry into the mechanism of model-based planning falsifies the idea that planning results from the propagation of activity through some form of map representation. This would

effectively falsify this whole area of modelling—all of the models in Ch. 2 effectively rely on some form of activity propagation. The only apparent exception is Erdem & Hasselmo 2012 (sec. 2.5), which theoretically plans by sending forward probes through the environment. However, in the majority of cases, Erdem & Hasselmo 2012 still rely on the propagation of activity through the map to make the forward probe mechanism work (sec. 2.5 discusses this in more detail). An alternative planning hypothesis, not based on propagation, is that task solutions are calculated by some sort of heuristic. For example, there are models of spatial planning that suppose that planning is done by triangulating the direction of the goal and then moving in that direction, without planning an explicit path through the environment [128]. However, this does not seem to fit the available evidence given by planning times: work on street navigation [57] and the Tower of London task [123, 129] suggest that planning times are proportional to the length of solution, which suggests that the planning process involves trying to produce a solution rather than merely a direction of travel through state space.

Another way that the backward (goal-based) propagation mechanism could be (partially) falsified is if it were demonstrated that map-based planning activity propagates forwards from the agent to various goal locations. The evidence conflicts on this point. The findings of Kurth-Nelson et al. 2016 indicate that retrieval of state transition sequences occurs in a backwards direction [9]. Kurth-Nelson et al. found that participants who had been trained to perform a simple grid world task produced spontaneous whole-brain magnetoencephalographic (MEG) activity[6] encoding reverse paths through the environment. To quote: "If a neural representation of state k was active at time t, then a representation of one or both of the states with transitions to k was active around t + 40ms." These four-state sequences were legal moves according to the transition structure of the environment.

In contrast, the findings of Johnson et al. 2007 show that sequences of states travel forward from a rat's position to investigate different

6 Decoded using a trained regression model.

potential paths when deciding which branch of a T-maze to search for food [26]. Finally, the verbal reports of London taxicab drivers indicate that an important mechanism for wayfinding is to identify the compass direction of the goal and then plan a short distance of actions forwards from their current state that will take them in the correct direction [121]. We have not been able to find robust map-based network models which plan using a forward mechanism (see ch. 2) and other modellers have criticised the idea of planning based on forward sweeps through a synaptic map [27].

It is also entirely possible that forward and backward propagation coexist in the brain, either as cooperating parts of a planning mechanism or as epiphenomena reflecting other processes. For example, it is possible that the backward spread of activation is used for planning whilst the forward preplay [26] that has been observed in the hippocampus is merely an epiphenomenon, or vice versa. It is equally possible that they both cooperate, and perform different roles. For example, the backwards spread of activity may reflect planning behaviour whilst the forward spread of activity may reactivate spatial memories in an order that makes sense for the animal. Or that backwards propagation might be used when planning towards a specific known goal location while forwards propagation might be used when considering alternative potential rewards in the environment. One potential way to cut through this is to perform experiments that suppress forwards or backwards preplay in animals during the performance of different tasks, in a similar manner to experiments that selectively suppress hippocampal replay [130, 131].

The model predicts, unlike other models, that the timing of the activation propagation is important to the planning process and that the relative activation of cells is largely irrelevant. It is unclear exactly how this could be falsified. On the other hand, the decaying activation hypothesis could potentially be falsified by investigating the sensitivity of neurons to different firing rates—implementing a decaying activation planning paradigm either requires neurons to be able to distinguish very similar firing rates (because the local gradient is usually very small) or is only able to plan over a very limited

number of consecutive actions before the activity decays to noise level (Fig. 2.10). If it is not possible for neurons to detect fine differences in firing rates then alternative planning hypotheses, including the idea that the timing of propagation denotes the optimality of different states and actions, become more credible.

In this light, it is perhaps important to bear in mind that neurons exhibit substantial trial-to-trial variability that seems to be linked to intrinsic factors such as synapse unreliability and synaptic background noise [96, 132]. Even if a presynaptic cell is driven repeatedly with identical stimuli, there is trial-to-trial variability in the postsynaptic response [132]. This variability primarily affects the membrane voltage and consequently the production of individual action potentials, but it suggests that it would be difficult for a neural system to reliably set up and maintain a large-scale decaying-activation gradient in which small variations in relative firing rates carry crucial information. Furthermore, patch-clamp experiments by London et al. 2010 [133] found that small variations in firing rate (a perturbation consisting of a single extra spike in one neuron) produced approximately 28 additional spikes in that neuron's postsynaptic targets. This again suggests that it would be difficult to reliably set up a large-scale fine-grained activation gradient, although it also suggests that small differences in relative firing rates could be plausibly amplified and read out.

Another way to falsify the propagating wavefront hypothesis is to investigate whether the planning process takes the same time regardless of the number of steps involved. If this is so, then it suggests that the timing of neuron activation does not depend on the task, and so that the timing of different neurons does not affect the planning process. As we argued earlier in this section, available experimental evidence seems to indicate that planning time is at least partly proportional to the number of the number of steps to the solution of the task [57, 123, 129].

A related test of the propagating wavefront hypothesis is to investigate how the overall level of activity in the system changes during the planning process, and how the overall level of activity changes

when the number of remaining actions to the goal changes. A number of fMRI experiments have found that the level of dorsolateral prefrontal cortex activation in a Tower of London task seems to positively correlate with difficulty (the number of actions required to reach a solution) [129]. The time-course of this activation may provide information about the mechanism being used. For example, in a decaying-activation paradigm, the amount of activity in the layer may rise sharply at the beginning of the planning process as the highly active parts of the map near the goal become active. As that activity propagates, it will decay. The amount of activity in the map will therefore plateau, as far-out elements of the map have very low activation and will therefore have a relatively small effect on the total activation in the layer.

There are several problems with this approach. Firstly, it is difficult to separate planning from the processes that precede it, follow it and support it. Secondly, the overall level of activity in the layer measuring against time may be more influenced by the neural implementation than the planning mechanism being used. It is not yet clear what activity/time relationship we would expect to see for the various planning methods that are possible, nor how this relationship might vary with the task and the environment. Thirdly, there is no guarantee that tasks in which this kind of analysis is possible are actually being solved in a model-based fashion. As we discussed earlier in this section, the number of tasks that can be solved using a model-based approach but not a model-free approach are small. The Tower of London task seems better suited to investigate the relationship between planning time and remaining actions than most, as the planning period can be separated from the task performance period as in Ward 1997 [123] and the task requires participants to move from a given start position to a given end position through a state space.[7] For this reason we have cited some Tower of London results in this section and elsewhere. However, although subjects are usually given

7 The Tower of London task presents subjects with two configurations of coloured balls placed on three pegs. One of these configurations is the 'Start' configuration and the other is the 'Goal' configuration. The subjects' task is to convert the start configuration into the goal configuration in the minimum number of moves.

some practice problems to solve, it is highly that participants have obtained a good cognitive map of the state transition space when they perform the task. The transitions between states are rule-based and so it is possible that a rule-based mechanism is being used to solve tasks.

10.2.5 *Further Work*

The model proposed in this thesis shows how model-based planning can be performed using a cognitive map encoded the recurrent connectivity between state-action cells that are formed into minicolumns encoding particular states. It introduces a propagating wavefront mechanism, and demonstrates a set of mechanisms that can support this mechanism: inhibition between state columns, a probabilistic propagation mechanism to allow propagating wavefront in nondeterministic environments, and a layer of gating cells for reading off the output of the planning process.

As described above, the proposed model addresses some limitations of earlier models whilst also having limitations of its own. There are therefore two main forms of further work that need to be done. Further *modelling* work is required to discover whether the current limitations of the model that we propose are inherent in the mechanisms that we are using or whether they can be improved. Further *experimental* work is also required, to test the predictions of this model with of other models where they conflict, and thus to determine which theories more accurately represent the planning mechanisms in human and animal brains.

Obviously, both the model-based and the model-free formulations of reinforcement learning are extremely active research topics at present. There is also a field investigating how these paradigms might compete and cooperate in the production of behaviour [1, 28, 43, 44]. Further research will make it clearer what tasks are and are not solvable with each of these approaches, and how they might need to be combined. With reference to this model more specifically, future modelling work will need to test whether the model can be made to work

with a continuous representation that can encode large, complex state spaces similar to those described by Friedrich 2016. Not only will this allow experiments to compare the model's behaviour on tasks of different scales (as described in Sec. 10.4) but it will also demonstrate that the proposed mechanisms can work with realistic state spaces, or whether a different encoding or planning mechanism is required.

On the experimental side, work is needed to discover whether model-based planning is truly based on a mechanism that searches forwards or backwards through an encoded map. The findings of Johnson et al. [26] and Kurth-Nelson et al. [9] seem to indicate the existence of such a mechanism, but do not prove it. Furthermore, experimental work is needed to distinguish between the various planning mechanisms that we have discussed: the decaying-activation paradigm, the propagating wavefront paradigm, and the various extensions proposed by Martinet 2011, Matsumoto 2011 and Erdem & Hasselmo 2012.

We have discussed a few approaches that could be used to interrogate these mechanisms. Further research on how planning times vary with the number of (remaining) actions in a task could show us whether neural activation needs to settle, as argued in several of the decaying-activation accounts, or whether the planning process outputs an action according to the amount of time that it takes goal-based activity to propagate to the agent's position, as in the proposed propagating wavefront mechanism. Studying how the level of activation in prefrontal areas varies during planning and with different numbers of remaining actions could show us whether planning is being carried out by a decaying activation mechanism (in which case the cumulative amount of activity is likely to stop rising after a certain amount of time as the propagating activity decays to unreadability) or whether activity is maintained at high levels as in our proposed model, in which case cumulative activity is likely to continue rising linearly or quadratically until an action is output. Studying what happens if participants are asked to give an action after different delay periods might also allow us to distinguish between these different accounts. If a participant is unexpectedly asked to give an action early

or late in the planning process, it might show how sensitive the period of action read-out is.

One of the important differences between the decaying-activation and propagating wavefront accounts is that —unless it is supplemented by a hierarchical mechanism similar to that described by Martinet 2011—the decaying-activation mechanism will be unable to plan over more than a certain number of actions before the decaying activation decays too far. By contrast, the propagating wavefront mechanism predicts that the number of remaining actions is not a limit provided that sufficient time is given to plan. These predictions can to some extent be investigated experimentally.

Finally, further recording work needs to be done to test what kind of neural encoding is used for cognitive maps. Place cell representations are well known [22] but—as we argued previously—do not represent the causal relationships between states. The rapid remapping of place cells in new environments also makes it unlikely that long-term task knowledge is stored in the recurrent connectivity between them, as this would become completely maladaptive every time the agent entered a different environment and would presumably have to be destroyed and relearned on redoing the original task [27]. Cells encoding various combinations of state and action have been found in the prefrontal cortex [88, 92–94] and their properties, as well as the connectivity between them, need to be further investigated, along with the possibility of the transition cells proposed by Cuperlier 2007. At the same time, work can be done to see whether the transition structure of the environment and the knowledge of how to bring about those transitions are separate, as suggested by Cuperlier 2007, Matsumoto 2011 and Erdem & Hasselmo 2012.

10.3 AIM 2: REMAIN CONSISTENT WITH KNOWN NEUROBIOLOGY

10.3.1 *Definition and Motivation*

As described in Section 10.2, the primary purpose of this model was to show how cognitive-map based planning could occur in the brain,

and how such maps could occur in the first place. In order to guide the modelling process, we adopted the secondary aim of trying to keep the model as close to the neurobiology as possible whilst still modelling the basic learning and planning processes. This necessitated a series of compromises: our guidelines for producing a biologically plausible model had to be stringent enough to be meaningful but not so stringent that they made meaningful progress towards a working model impossible.

We came up with several core guidelines:

- The model should be neural. It should be comprised of individual neurons that communicated by passing activity through the synapses that connected them.
- Given that the network's behaviour relies on the connectivity of the neurons that comprise it, as far as possible it should be demonstrated that this connectivity can be brought about by biologically plausible learning processes from the sensory and motor feedback that the network would receive as it explored its environment.
- Given that the network should be self-organising, the synaptic plasticity responsible for this self-organisation should be Hebbian as far as possible, and should rely on local inputs. In other words, synaptic plasticity should rely purely on the activities of the presynaptic and postsynaptic cells. A consequence of this was that the model should not rely on the direct backpropagation of error from a non-local error signal to adjust neural weights.[8]

At the same time, we relaxed several conditions that would have made the network more biologically plausible:

8 The question of whether or not the brain learns using the backpropagation of error, as in a deep learning network, is still debated. It was originally argued that synapses in the brain would not be able to determine how to change in the strength of their synaptic weights in order to decrease the error made by the network as a whole [134, 135]. Recently, researchers have begun to put forward alternative learning rules that produce results similar to the backpropagation of error in some circumstances [136, 137], but these methods are complex, still under investigation and require very particular synaptic connectivities and neural dynamics [136]. The question of if, how and when such methods apply is beyond the scope of this research and so we have chosen to take a conservative approach.

- Given that the model is neural, we decided not to enforce that it should model the spiking dynamics of neurons in the brain. (Put another way, we decided to use a rate-coded model.) This was partly in order to enforce the constraint of least detail when modelling. A simple rate-coded model that can capture the appropriate behaviour and neural responses is in some ways a better model than a more complex spiking model that does not produce further important behaviours or predictions. Assuming that spiking behaviour *is* important to the formation and use of cognitive maps, we nevertheless decided that we would achieve more progress towards the ideal model of these processes by producing an initial model without spiking dynamics and then producing a later, more detailed model that explicitly models spiking dynamics.

- We decided not to model the network using continuous time. This allowed us to simplify the modelling process by removing the need for events to have specific temporal characteristics (beginning, end, duration) and making it easier to model how events took place within a timestep (A happens, and then B, and then C).

- We decided to use simple neural codes as far as possible. In particular, we have mostly used one-hot encodings—a layer of cells where each neuron represents only one state or action, and only one neuron is active at a time. This is discussed at more length in Secs. 3.3 and 4.4, which shows how the network can learn and plan when its inputs are more complex. We have also made heavy use of winner-take-all competition during the learning of the map and the self-organisation of the network. The use of winner-take-all competition and the resulting grandmother cell representations for SA cells and gating cells made it considerably easier to develop the model and understand its characteristics. Although it makes the model less accurate in the short-term, lacking the sophisticated neural representations that are predicted by both deep reinforcement learning theories

(sec. 1.5.4) and various neural coding theories [107], it allowed us to make considerable progress towards achieving Aim 1 and in particular allowed us to investigate sequence-based planning later in this thesis. Model-based planning is a complex process and so this compromise is true of other work in this area (see Ch. 2).

- As part of this simplification, we decided to model inhibition implicitly, by passing neural activation through various functions, rather than by explicitly modelling inhibitory interneurons and inhibitory synapses.

10.3.2 *Progress Made By Previous Work*

Ch 2 showcases a variety of previous models that have also investigated the neural processes by which an agent forms and uses a cognitive map. As described in Sec. 10.2, these models are able to perform a degree of planning using a cognitive map and all model the process of forming that map to some extent.

10.3.2.1 *The Relationship Between Learning and Planning in Previous Modelling*

There are several questions that are left open by this previous work. Firstly, there is the relationship between the learning and planning process that occur in these networks. All of the models cited have separate learning and planning phases. These phases have different dynamics and plasticity. From a modelling perspective, this is expected, since learning a cognitive map using the mechanisms described in these models requires the sequential activation of states or state-actions or columns along the agent's path of movement. At each timestep the agent must self-organise (or retrieve) a representation for its current state or state-action, and link this representation to the previous representation. By contrast, the planning mechanisms described by these models all require some form of spreading propagation through the cells and synapses encoding the cognitive map. The

difference between these 'learning' and 'planning' mechanisms suggests that the cells involved behave in quite different ways when each mechanism is active, and further suggests that these mechanisms may be mutually exclusive.

As such, all of the models described in Ch. 2 have separate learning and planning phases. These phases occur in slightly different ways: in some models there is a long exploration phase followed by a series of planning tasks; in other models exploration alternates with planning, or occurs stochastically during planning. Regardless, the processes described for storing the results of this information within the neural network are not the same as the processes for planning using this information. The models therefore switch between learning or planning mode, and they do not model how this switch could occur.

10.3.2.2 *Neural Substrate of the Cognitive Map*

A requirement for modelling the process of planning using a cognitive map is modelling the cellular substrate that encodes this map. The models described in Ch. 2 show, consistently with the known neurobiology, that usable cognitive maps can be encoded using state/place cells, transition cells, or state-action cells. However, with the exception of Cuperlier 2007 and Martinet 2011, these models do not attempt to show how the proposed cellular representations could self-organise.

Cuperlier 2007 demonstrated a neural network model that formed a cognitive map made out of transition cells, which encode a deterministic transition between two states. The recurrent activity between these transition cells encodes the transition structure of the environment (i.e., the cognitive map) and the model is able to plan by propagating activity through this transition cell layer.

Cuperlier 2007 predicted that, given a certain kind of initial hardwired connectivity, cells receiving both previous-state and current-state input could self-organise to represent a unique combination of these inputs and therefore come to represent a particular transition. This hardwiring was necessary to ensure that each potential transition cell only received current-state input from one current-state cell, therefore simplifying the competition. The model does not predict

how this initial connectivity would come about, but the consequent self-organisation process that they describe is able to happen using local Hebbian learning.

Martinet 2011 demonstrated a neural network model that formed a cognitive map out of complex state minicolumns, which contain complex, stereotyped and hardwired connections between state (place) cells, action cells and state-action cells. This model is able to plan using entirely neural processes but although Martinet 2011 technically provide an explanation for how a minicolumn comes to represent a given state, this explanation is incomplete and does not involve Hebbian learning. It is assumed that the cell types within each minicolumn already exist (in other words, Martinet et al. do not model how the state cells, action cells and state-action cells would self-organise) and that the connectivity within each minicolumn is hardwired.

During exploration, an externally-controlled two-phase process takes place. In the first phase, the model automatically assigns place cells to map locations. The place cell assignment process continues until at least six place cells are active for every state that the model has visited, as verified by an external algorithm. When this happens, the model switches to the second phase, where it assigns mini-columns to states. This is done by algorithmically checking whether any column currently responds strongly to that state; if none are responding, then the authors connect a random column to that state. That column is then connected to all place cells that are firing above a certain threshold. Although the place cell representations and the connectivity between columns and place cells are technically learned as the agent explores, the processes that are used to do so are essentially non-neural.

It should be possible to test for the presence of transition cell representations (as predicted by Cuperlier) and/or state-action cell representations (as predicted by Martinet) using experimental data obtained. State-action cell responses have been recorded in the prefrontal cortex by several researchers [88, 91–94] but frequently these cells have more complicated and task-dependent response patterns. Cuperlier predict that transition cells in the hippocampus might be re-

sponsible for the existence of directional place cells but state that "the biological proof of transition cells might be hard to achieve as it would be difficult to observe the difference between (directional) place cells and transition cells". Matsumoto 2011 and Erdem & Hasselmo 2012 predict planning taking place using place cells, and of course there has been extensive demonstrations of these cells in experimental data. These models do not predict how these representations would self-organise, although there is other work on this topic [24, 25].

10.3.2.3 *Encoding of the Cognitive Map in Recurrent Connectivity*

All of the reviewed networks encode the cognitive map as a set of synapses that connect the cellular substrate (state/place cell, state-action cells or neocortical minicolumns). These synapses can be assigned algorithmically or they can be self-organised using a variety of methods.

In Cuperlier 2007, Martinet 2011 and Friedrich 2016, the connectivity is not self-organised using biological processes. Instead, as the agent explores its environment, the appropriate cells are connected together algorithmically. In the case of Cuperlier 2007, which uses transition cells (see above) this means setting the synapses between the current and previous transition cell to an arbitrary positive value after a state transition. In Martinet 2011 and Friedrich 2016, it means doing the same with state-action cells.

Erdem & Hasselmo 2012 model a mechanism which is slightly more consistent with the known neurobiology. In this model, the topology cells (which represent the topology of the environment) are essentially place cells, but are connected together using a learning rule which does not incorporate any firing information related to the topology cells that are being connected together. Instead, the equation used implicitly combines activation, competition and recency cell activity to connect topology cells together as a function of a "recency signal". This implicit modelling seems to presuppose that recency cell activity would activate the appropriate topology cells in certain ways, that certain forms of competition could be applied to the topo-

logy cells such that their firing came to encode the desired function of their inputs, and that Hebbian learning could then connect the still-firing topology cells together in such a way as to obtain the desired connectivity. Essentially, the learning rule implicitly describes an extended sequence of mechanisms that are not explicitly modelled. Since these mechanisms are not explicitly simulated it cannot be assumed without reservation that they would operate as hypothesised and without unforeseen side-effects. The possibility that—if these mechanisms were implemented explicitly—Erdem & Hasselmo 2012 might be unable to learn a cognitive map is a limitation in this work.

Hasselmo 2005 uses a Hebbian local learning rule that implements a "memory buffer" which holds activity from the input state, thus allowing self-organisation of the synapses between the minicolumns that encode the cognitive map. The actual self-organisation is highly complex (see Figs. 2.5 & 2.6) but appears to be Hebbian.

Finally, Matsumoto 2011, which encodes the cognitive map between place cells with large, overlapping fields, is able to encode the relationship between states using a simple Hebbian rule. Each location in the state space is signified by the combined activation of multiple state cells. As the agent moves through the environment, it will move from one receptive field to another, and as it moves between the two receptive fields both place cells will be firing simultaneously. A simple Hebbian learning rule is therefore able to connect neighbouring place cells.

10.3.3 *Progress Made By This Thesis*

In the previous section we highlighted several areas that represent flaws in existing models (from a neurobiological point of view). In other words, areas where a mechanism has not yet been found that fits within the guidelines we have specified. These areas include: how learning and planning modes interact in the same network, and how neural mechanism could produce the shift between these modes; what the cellular substrate of the cognitive map should be, and how it

should form, and how the recurrent connectivity necessary to encoding the synaptic map is created.

The model proposed by this thesis adheres to the guidelines that we produced reasonably well. It forms and uses a cognitive map entirely through neural mechanisms, assuming only that reliable sensory and motor feedback are available from external systems. The processing required for both the learning and planning mechanisms is performed entirely through the synaptic communication between neurons. The model therefore attempts to address the open questions described above.

10.3.3.1 *The Relationship Between Learning and Planning in Previous Modelling*

The relationship between learning and planning in this model is similar to the previous models. Similarly to those models, the proposed model has separate learning and planning modes, which involve different neural dynamics, different plasticity and different patterns of activation. These modes can be arranged in various ways: for example, Sec. 7.1 shows that a 'surprise' signal can be used to trigger a switch from the planning to the learning modes when the agent encounters an unexpected obstacle. Sec. 7.4 describes a mechanism whereby the network will switch from planning to learning mode when too much time has elapsed in the same position. However, we still have no mechanism that could manage these two modes in a plausible fashion.

In other words, the relationship between learning and planning in these models is still an open question. It is possible that learning and planning are not in fact separate neural processes. Another explanation is that there is an unmodelled neural system that is responsible for shifting the relevant neural circuits between these two phases. For now, it is an open question how the learning and planning phases relate in a neural fashion.

10.3.3.2 *Neural Substrate of the Cognitive Map*

We now turn to the issue of self-organising the cognitive map. Figure 10.2 shows the proposed network architecture (without the hierarchical mechanism described in sec. 10.4).

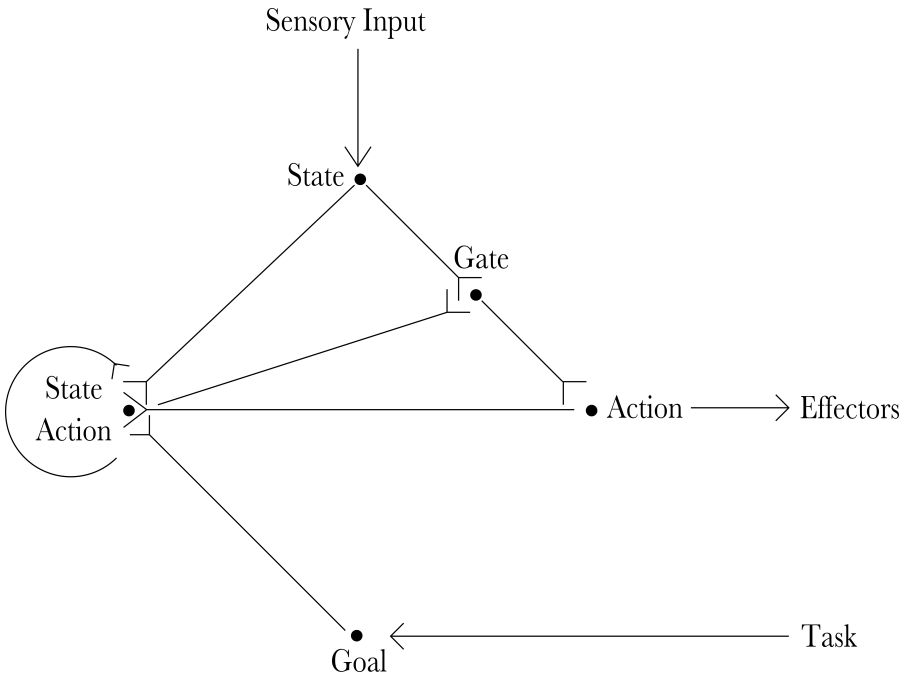


Figure 10.2: Architecture of the Proposed Network Model (Without Hierarchical Mechanism)

The primary connectivity necessary for these processes are as follows:

- State cells to SA cells: self-organised using local Hebbian learning (ch. 4).
- Action cells to SA cells: self-organised using local Hebbian learning (ch. 4),
- SA cells to SA cells: self-organised using local Hebbian learning with memory trace (ch. 5).
- State cells to Gating cells: self-organised using local Hebbian learning (ch. 6).
- SA cells to Gating cells: self-organised using local Hebbian learning (ch. 6).

- Gating cells to action cells: self-organised using local Hebbian learning (ch. 6).
- Goal to SA cells: this is not self-organised at present. Instead, goals are presented in the form of state cell representations and the synapses between the goal cells and the SA cells are identical to the synapses between the state cells and the SA cells.

Cells with analogous properties to SA cells [88, 91–94], action cells [58] and gating cells [100] exist in the prefrontal cortex, premotor cortex and basal ganglia respectively. In Ch. 4 we show that the self-organisation of SA cells can be done using several mechanisms: diluted connectivity, columnar winner-takes-all, or bandstop inhibition. These mechanisms can also be combined in various ways. All of them essentially fit within the guidelines of plausibility that we have stated: they rely on neural processes (notwithstanding the use of implicit inhibition) and they model the self-organisation of SA cells using a Hebbian learning rule that relies only on local information.

10.3.3.3 *Encoding of the Cognitive Map in Recurrent Connectivity*

As stated in Sec. 10.3.2, all of the reviewed networks encode the cognitive map as a set of synapses that connect the cellular substrate (state/place cell, state-action cells or neocortical minicolumns). In this thesis we have also used such a formulation, proposing that the cognitive map is encoded between state-action cells.

We propose that the recurrent connectivity between these state-action cells is established by a local Hebbian learning rule using a memory trace. This trace is 'set' during one timestep, when the state-action for that timestep has been learned or retrieved and is therefore active. Also during each timestep, when the state column has been formed, a local learning rule uses this memory trace to form a synaptic connection between the currently-active state column and the state-action cell active in the previous timestep.

10.3.3.4 *Planning*

In Sec. 10.2, we proposed several mechanisms for supporting propagating wavefront planning. In particular, we proposed the idea of probabilistic propagation, a mechanism that allowed a propagating wavefront model to operate in a nondeterministic world (Sec. 7.2). To our knowledge, we are the first to investigate this problem. There are therefore no other paradigms that we can compare this theory to. The two modes of interrogating this theory are therefore: firstly, to ask how plausible it is in terms of the known neurophysiology, and secondly, to ask what predictions it makes and how well they match what is known. In Sec. 10.2 we made the prediction that planning times would be longer in an environment where the transitions were uncertain vs. an equivalent but deterministic environment.

In terms of plausibility, as discussed in Sec. 7.2, the idea of probabilistic propagation appears to be plausible if planning is considered to operate over short periods of time. This is one instance where using a rate-coded neural network appears to be counterproductive. But if we consider synaptic transmission as naturally unreliable, such that the likelihood of a postsynaptic spike is linked to the synaptic weight, which is in itself linked to the probability of neurotransmitter vesicle release at the presynaptic terminal, then it seems likely that the strength of the synaptic weight affects the chance of a given spike being 'passed on'. On a small timescale, it therefore seems possible that the weights of recurrent synapses would affect the speed of activity propagation.

10.3.4 *Predictions and Falsifications*

10.3.4.1 *Neural Substrate of the Cognitive Map*

In the above section, we described several proposed mechanisms for self-organising the neural substrate (transition cells, state-action cells, etc.). The proposed model gives several mechanisms for producing state-action cells. Cuperlier 2007 also provides explanations for how the cellular substrate of the cognitive map could come about. Specific-

ally, Cuperlier 2007 proposed a cognitive map made out of transition cells and predicted that, given a certain kind of initial hardwired connectivity, cells receiving both previous-state and current-state input could self-organise to represent a unique combination of these inputs and therefore come to represent a particular transition.

One method for distinguishing and investigating these predictions is to look at the initial connectivity that is required and how this affects the resulting cellular substrate. Cuperlier et al. describe an initial hardwired connectivity within the cellular substrate made up of 'neighbourhoods', such that a set ('neighbourhood') of potential transition cells are linked to the same current-state cell. The cells in a neighbourhood then learn to associate themselves with different previous-state cells, thus coming to represent different transitions to the same state. The columnar winner-take-all approach that we proposed in Ch. 4 has a similar starting point. Specifically, that potential state-action cells are linked together in a minicolumn, such that inhibition between and within minicolumns is possible. These columns become associated with a given state, and the cells within them become associated with particular state-action combinations that can occur in that state.

Both of these theories suppose that cells associated with the same (current) state are initially connected together. In other words, cells associated with the current state are not chosen randomly, but are chosen from a small pool of pre-existing cells. This is in contrast to the competitive-learning-with-diluted-connectivity approach, which posits that potential SA cells receive diluted afferent connectivity from the state cells and the action cells, and that any state-action cell which happens to receive the appropriate connectivity to encode the current state-action may compete to do so. In the bandstop mechanism, all state-action cells can compete to represent the current state-action provided that they do not already encode a different state-action. In other words, the 'neighbourhood' approach and the columnar winner-takes-all approach both propose that the cellular substrate of the cognitive map self-organises in a structured manner, whereas

the diluted connectivity approach and the bandstop approach as formulated by Ch. 4) do not.

This difference between cellular self-organisation mechanisms potentially provides a mechanism for making and interrogating predictions about these mechanisms. Minicolumns as described in the literature have a literal columnar structure in the brain [95]. The columnar winner-takes-all theory therefore specifies that all state-actions associated with a given state should be encoded by cells that are close to each other in space. This is something that can potentially be interrogated by recording studies. In this respect it is a useful theory.

Cuperlier's neighbourhoods theory is more ambiguous: it proposes neighbourhood structures but does not draw conclusions about how these structures would appear in the brain. Minicolumn structures are primarily a neocortical phenomenon [95, 138], whereas Cuperlier predicts that these neighbourhoods of transition cells would appear in the hippocampus. It is therefore less clear how they would manifest. Nevertheless, this may prove a useful avenue for interrogating these models.

10.3.4.2 *Encoding of the Cognitive Map in Recurrent Connectivity*

The proposed model predicts that the cognitive map (the actual state transitions available in the environment) is encoded in the recurrent synaptic weights between state-action cells. We proposed (in Ch. 5) a 'trace learning' mechanism that produces such recurrent connectivity using a 'memory trace'. Essentially, this mechanism proposes that neurons which fire consecutively are able to connect together based on a memory of their previous firing. Hasselmo 2005 predicts the existence of a somewhat similar mechanism, the 'memory buffer', which stores activation from the previous timestep to be used for learning in the next timestep. Erdem & Hasselmo 2012 do not give an explicitly neural learning rule but calculate synaptic weights based on a 'recency signal'.

Essentially, all of these models predict that some mechanism exists which can keep track of previous neural firing and use this to form recurrent connectivity. This presupposition can be termed the 'memory'

theory of cognitive map formation. There is some evidence that this is possible—Rogerson et al. 2014 review a set of molecular mechanisms that allow neuron to associate events that are separated in time during electrophysiological experiments [114]. The most important of these mechanisms is the CREB protein, which attaches to a firing neuron and increases that neuron's excitability, thus predisposing it to fire again when a new input is received and so to associate these two temporally separate inputs.

The memory theory of cognitive map encoding contrasts with that of Matsumoto 2011, which posits that the cognitive map is encoded between state (place) cells with large, overlapping receptive fields and therefore that the cognitive map can be encoded using a simple Hebbian rule (Sec. 10.3.2). Since the rest of the reviewed models do not model the formation of the cognitive map in neural at all (Cuperlier 2007, Martinet 2011 and Friedrich 2016), these two theories appear to be the primary explanation for the formation of this connectivity. Falsifying one or the other of these explanations will require more evidence on the type of cells that encode the cognitive map (discrete vs. continuous, state vs. state-action).

10.3.5 *Further Work*

In this thesis, we have proposed several novel mechanisms for the formation and use of cognitive maps, whilst remaining as far as possible within the guidelines set out in Sec. 10.3.1. An important avenue of further work is to see how many of these mechanisms can still be used if these guidelines are tightened. For example, further modelling work needs to be applied to the probabilistic propagation mechanism that we have proposed. We presented an argument of favour of the plausibility of this idea in Sec. 10.3.3 but a detailed spiking network model would be necessary whether or not this mechanism could realistically work as we describe. Similarly, we have argued that a combination of winner-takes-all and bandstop inhibition operates during the self-organisation of the SA and gating cell layers, and a more detailed model could investigate these processes specifically.

If it becomes clear that some of these mechanisms cannot be implemented in more biologically realistic models, further future work would be necessary in order to discover whether these mechanisms can be improved or replaced. If an alternative mechanism with a similar function cannot be discovered, and the proposed model cannot be implemented without these mechanisms, it suggests that alternative neural explanations of model-based planning may need to be adopted.

As described in Sec. 10.3.2, all of the models that we have reviewed have two separate modes of operation: one for learning a cognitive map, the other for using it to plan. This is necessary because the patterns of activation required, the inhibition and propagation required to produce them, and the level of plasticity in the model all vary between these two modes, and the learning/planning separation remains an unsolved problem. There are two potential routes to a solution. The first is to investigate oscillatory or episodic neural mechanisms that could alter the properties of the relevant neural circuits to move them between these modes. Experimental data demonstrates that switching between exploration and exploitation radically changes prefrontal cortex dynamics, making activity less predictive of choice [139]. It is also thought that neural rhythms play a role in the periodic modulation of synaptic transmission and plasticity and many excitatory and inhibitory neurons fire preferentially at different phases during a theta cycle [140].

An alternative solution is to find learning and/or planning mechanisms that are not incompatible with each other. For example, in the case for probabilistic propagation we argue that the propagating wavefront mechanism may operate at a very fast timescale, and in consequence that the number of spikes involved is likely to be low. It may then be the case that there is not enough sustained activity to produce meaningful synaptic plasticity and so there is no need to worry that these patterns of activation would affect the connectivity of the cognitive map, whereas the agent receives strong state and action feedback when it enters or leaves a state and so learns the ap-

propriate elements of the map. In this way the learning and planning mechanisms may be made more compatible.

These two solutions to the learning/planning problem may coexist. For example, it is possible that the recurrent plasticity in the SA layer may in fact be left on during planning. The primary element of the planning stage (in the proposed model) is that activity propagates through the SA layer in a pattern that is defined by the cognitive map, allowing the model to output actions that move it towards the goal. If the cognitive map is defined by recurrent connectivity between SA cells, it follows that the activity propagating through the SA layer in the planning mode is likely to do so according to the pre-existing recurrent connectivity. In other words, one SA cell will drive another SA cell according to the preexisting recurrent connectivity between them. It is therefore possible that plasticity could remain active in the SA layer even when the agent is planning. This seems to be more likely to occur if the model encodes a forward map, because the pattern of activation in the SA layer during learning would then be qualitatively similar to the pattern of activation during planning. Specifically, during the learning phase one SA cell would become active and then its successor cell(s) would become active, and during the planning phase if an SA cell becomes active, all of its successor SA cell(s) will become active just afterwards.

10.4 AIM 4: LEARN AND USE A HIERARCHICAL PLANNING MECHANISM

10.4.1 *Definition and Motivation*

There is strong evidence that humans and animals represent space in a hierarchical fashion and this seems to play an important role in planning (ch. 1 & 8). There is also evidence that humans and certain animals (rats and bats) are able to plan over very different spatial and temporal scales, ranging from a few centimetres to thousands of kilometres [4].

Our initial findings, as well as experimental evidence, suggested that planning time correlates reasonably closely with the size of the task. Howard 2014 found that during a spatial navigation task the planning time for participants was correlated to the distance to the goal [57], and Ward & Allport 1997 found that when participants were required to prepare a solution to a 5-disc Tower of London task[9] the preparation time was correlated to the number of movements required to reach the goal [123]. This correlation seems to be incompatible with the idea that humans and animals plan over vastly different spatial scales. If planning time is correlated to the number of moves required to solve the 5-disc task (between 2 and 12) then how could humans and animals plan over thousands of kilometres and millions of movements in a reasonable period of time?

As an example, the 5-disc task used by Ward & Allport 1997 [123] used 2-12 actions and recorded mean planning times on the order of seconds. Meanwhile, the navigational task used by Howard et al. 2014 [57] asked participants to navigate a complex street map to a goal several hundred metres away and again recorded planning times on the order of seconds. In both cases, planning times were correlated with the number of actions / the length of the route. And yet, even though these tasks occur on very different scales, the planning times appear very similar. This comparison is not rigorous, for reasons that we will explore further in Sec. 10.4.4, but it potentially shows that planning can occur at different scales and levels of detail, and thus planning time can correlate with complexity within a task without correlating between tasks.

Our hypothesis is that a hierarchical mechanism might be responsible for the apparent lack of effect that task scale has on planning time. Essentially, if a 'move' can be of arbitrary size and complexity, such that movement over a few centimetres can consist of ~10 'moves' but movement over a hundred metres (10, 000 times that distance) would also consist of ~10 'moves', then it makes sense that on any

9 The 5-Disc Tower of London task consists of two configurations of 5 rings placed on three pegs. The rings can be moved between the pegs according to specific rules. The agent starts at one configuration and must perform a series of moves to produce the second configuration.

given scale planning time depends on the number of moves but that if the scale changes by orders of magnitude then the 'move' increases correspondingly in size so that the planning time remains within a limited (and feasible) range. This corresponds with the findings of Ward & Allport 1997 [123], who found that although participants' planning time correlated with the number of moves in the solution, it correlates more strongly to the number of "subgoal chunks" in the task. A "subgoal chunk" referred to a consecutive series of moves that all transfer discs to and from the same pegs, presumably in a relatively predictable manner. Ward & Allport speculate that such chunks comprise a "mental unit of planning" [123].

10.4.2 *Progress Made by Previous Work*

With the exception of Martinet 2011, the models reviewed in Ch. 2 do not consider this issue of scaling. As observed above, these models use a decaying-activation mechanism and so are unlikely to produce appropriate scaling because the activity is likely to decay too much in larger maps. Martinet 2011 introduces an extra map at a lower resolution, which uses proprioceptive feedback to essentially merge states that are considered to be functional aliases of each other. This mechanism extends the range over which their model can plan before the decaying activation disappears completely. This could potentially explain the relationship between planning time and distance described above.

Martinet 2011 only model this mechanism in a very particular circumstance: a Tolman detour maze 7.12, which is formed exclusively from straight line corridors without wider sections. Since the low-resolution map is based on a proprioceptive signal, this allows the low-resolution map to merge states that lie on a straight line. This works because these states are functionally identical from a task perspective. Martinet 2011 do not show this mechanism planning in any other environment.

Martinet et al. state that the role of the low-resolution map in their model is to allow activation to spread further through the high-

resolution map: essentially, activation seems to spread over the low-resolution transitions before being injected into the high-resolution map. It is not entirely clear how this works in practice, and there are potential issues. For example, is the result of this mechanism a large (global) gradient spread over the entire map, or a set of regions (a cluster of high-resolution states corresponding to one low-resolution state) that each have their own independent gradient? If the result is a global gradient, this does not seem to solve the fundamental problem of decaying activation—that over a sufficiently large map, either the gradient at every point becomes so shallow that it is effectively unreadable, or the gradient must decay to near zero.

If the gradients are local (regional), then various issues may arise. For example, what do the boundaries between regions look like? Do the gradients in neighbouring regions interfere with each other, and if not, why not? How are regional gradients set up so that they point towards the goal? Do local attractors (isolated 'hills' of high activity) arise that can trap an agent within a region?[10] Further exploration of the hierarchical dynamics in Martinet 2011, along the lines of the connectivity experiments we performed in Sec. 8.1 would be very interesting for this reason. There are also open questions about the resolution at which this alternative map would operate, and about how many maps of different resolutions Martinet's model could form and sustain. In particular, a proprioceptive signal seems unlikely to be useful in forming maps on scales above a few metres.

10.4.3 *Progress Made By This Thesis*

In this thesis we propose an alternative hierarchical mechanism that works on the principle of learned sequences. Rather than merge similar states into a low-resolution map, the proposed model learns frequently used sequences and uses these to speed up the planning process.

10 We experienced similar issues with a very early decaying-activation version of the hierarchical model presented in this thesis.

In Chapter 8, we investigated what kind of connectivity would be required to allow this mechanism to work (Sec. 8.1), and showed that the mechanism can successfully be used to plan and to augment planning (Figs. 8.6 and 8.8). We followed this by showing that the use of the hierarchical mechanism allows for faster planning in large state spaces, particularly on tasks with longer solutions that require more actions (Fig. 8.23 & 8.24). In Chapter 9 we explained how the required connectivity could self-organise whilst remaining within the biologically plausible guidelines described in Sec. 10.3. We then briefly investigated the kind of sequences that are learned, and showed that they seem to be primarily located within the areas of state-space that are most commonly encountered by the agent (Fig. 9.3).

The mechanism that we describe may provide a better explanation of the tendency of people to rely on familiar routes through large spaces and when under time pressure [56, 120, 121] than the variable-resolution approach proposed by Martinet 2011. The low-resolution map, once formed, covers of all encountered state space and is then used to perform optimal planning. In other words, the low-resolution map approach does not specifically encourage the repetition of previously useful behaviour. By contrast, the sequence-based approach we propose explicitly produces a hierarchy based on stereotyped sequences of motor primitives and so is likely to produce highly stereotyped behaviour in familiar circumstances (Fig. 8.25).

A sequence-based approach based on a propagating wavefront mechanism also provides more definite predictions about planning time. The effect of learned sequences on planning time is distinct, measurable, and depends on the type and number of sequences that are learned (Fig. 8.23). Although the model is not currently able to provide predictions for very large state spaces, because of its discrete state-action map encoding (Sec. 10.2.3) the mechanisms we describe should inherently produce testable predictions on this basis. At present, the hierarchical models produce planning times roughly equal to the number of moves required to reach the goal, where each encoded sequence is counted as a move. Planning time therefore increases with the number of actions required to reach the goal, but this increase

is modulated by the presence of learned sequences (Fig. 8.23). This seems to correlate with the observations of Ward & Allport that planning times in a Tower of London task correlate with the number of high-level actions involved in the task solution [123].

The hierarchical mechanism that we describe also makes the relationship between the hierarchical levels very clear. The variable-resolution mechanism proposed by Martinet 2011 uses the network's low-resolution map to supplement activity in the high-resolution map, a relatively complex and subtle interaction. By contrast, the sequence-based mechanism makes clear and testable predictions about when and why sequence cells should fire or not fire. This allows us to compare their behaviour to existing recording studies such as the comparison to pre-SMA sequence cells made in Fig. 8.26.

The mechanism by which the sequence cells are activated should also ensure that the learned sequences are not utilised unless they are adaptive (Sec. 8.1). By contrast, a decaying-activation mechanism which uses a low-resolution map to inject activity into faraway states, as Martinet 2011 seems to be at risk of producing local maxima of activity. Essentially, it may be possible for the local gradient to be highest at the source of injected activity, such that the activity is attracted to a state or set of states that is being stimulated by the low-resolution map and then remaining there rather than move towards the true goal. We experienced similar difficulties in early experiments that used a similar hierarchical decaying-activation mechanism; this was one of the reasons for choosing to use a propagating wavefront planning mechanism instead.

10.4.4 *Predictions and Falsification*

This model predicts that if an animal performs many tasks in an environment and forms sequence cells, it should begin to perform tasks faster. Specifically, the correlation between planning time and path length (or number of moves) should decrease, as the animal moves from planning without sequence cells to planning with sequence cells (Fig. 8.23).

Furthermore, the model predicts that sequence cells exist in the brain, in an area that is directly involved with planning or is connected to one. Recording results from Shima & Tanji show that cells exist in the pre-SMA that seem to fit the expected properties of sequence cells [126]. These cells are active for a sequence of movements, but they are not active any particular one of those movements outside the context of that sequence. Furthermore, they are not active for other sequences that use the same movements in a different order. Experiments in Ch. 8 seem to show that the firing patterns of a sequence cell in the proposed model are similar to those recorded by Shima & Tanji (Fig. 8.26c).

As we mentioned previously, the main hypothesis driving our hierarchical work is that such a mechanism could account for the discrepancy between, on the one hand, the observation that planning time correlates with the number of remaining actions in any given task [57, 123, 129]; and on the other hand, the observation that planning times seems to vary over a relatively small range regardless of the scale of the task in question. This prediction is difficult to test. Part of the reason for this is that there is a paucity of data on planning at very different scales. Geva-Sagiv et al. [4] argue that this is because these scales have traditionally been the preserve of different disciplines: psychologists have conducted experiments on (rodent) planning on the scale of laboratory tasks (e.g., the Tolman detour task [5, 21], the Morris water maze task [98]) while long-distance animal navigation has been investigated primarily by zoologists. Furthermore, data from different tasks is very difficult to compare rigorously. In Sec. 10.4.1 we argued that the planning times for a 5-disc Tower of London task requiring 2 to 12 movements to produce a solution [123] were in fact higher than the planning times for a task navigating through Soho that required 200 to 400 movements over distances of hundreds of metres [57]. However, the navigational task was conducted in a laboratory, so that participants watched a video of the route rather than physically making the movements, and participants only gave input at each street intersection. Are these inputs 'high-level movements' similar to those we model as being learned by the sequence

cells (with footsteps as state-action combinations)? Alternatively, are the street intersections actually the state-action combinations in this task? If the second scenario is true, then it is perfectly natural for the planning times to be lower in the navigational problem than in the 5-disc Tower of London problem and no hierarchical explanation is necessary.

Furthermore, it is very difficult to separate planning from other processes. In the Ward & Allport 1997 [123] experiment described above, participants were required to produce the entire plan in their head in advance and then to carry it out after a button press, failing the task if they took longer than 2.5s to produce any movement. In the Howard 2014 navigation task, participants were asked to output an action at every street intersection and their reaction times were measured [57]. It is, again, unclear how comparable this data is. Ultimately, we cannot rigorously test the hypothesis that the relationship between planning time and task size is as we predict without further experimental work.

More difficulty comes from the fact that as yet the model is not able to form cognitive maps for large environments. We have run experiments on environments of different sizes but it is not currently possible to encode state spaces that vary in size by orders of magnitude. Therefore it is impossible for the model in its current state to produce behaviour across tasks that vary in scale by 10x or 100x. To make predictions about the model's behaviour at larger scales we have therefore been forced to extrapolate from the model's performance on small-scale tasks (Fig. 8.24). There are several ways in which the model could be altered to do so, both by using more continuous and complex state representations and by making more efficient use of the limited number of state-action cells that the model has available.

10.4.5 *Further Work*

Further modelling and experimental work is necessary to investigate the hierarchical elements of model-based planning. Further model-

ling work should address the limitations of the sequence-cell and variable-resolution hierarchical mechanisms, and characterise their behaviours more widely. Modellers may also find other potential hierarchical mechanisms. Further experimental work is also needed, in order to distinguish which of these mechanisms is operating in the brain, as well as to confirm that a hierarchical mechanism is at work at all.

Perhaps the most important modelling problem at present is to investigate mechanisms for encoding continuous state spaces and consequently to plan in large, complex state spaces. This would allow the hierarchical mechanisms to be modelled and tested on tasks of different scales.

Furthermore, the sequences that the model learns, and the mechanisms that encode them, needed to be investigated and modelled in more depth. At present, the proposed model learns one sequence (usually of two to five state-action combinations according to Fig. 9.4) per sequence cell. These sequence cells are individually activated by activity propagating through the state-action layer, and provide this activity with a short-cut to travel through the layer faster. It is possible that in very large tasks (for example, taxi drivers crossing a city [121] or bats travelling hundreds or thousands of kilometres [4]) that this would require the activation of hundreds or thousands of sequence cells, and that this process would still be insufficient to plan in a reasonable period of time. It is therefore possible, even likely, that these sequence cells are combined into larger and more complex representations. If each sequence cell encodes between two to five actions, then a 'higher' sequence cell could receive connectivity from two to five sequences and therefore up to twenty-five state-actions. It would be very interesting to investigate this possibility of a 'hierarchy of hierarchies' and how it might interact with large-scale planning, habits, and the formation of skills.

Experimentally, as Geva-Sagiv [4] suggest, more experimental work is needed to investigate planning at different scales, merging the work done by experimental psychologists on laboratory planning tasks and the work done by biologists on long-distance navigation. It is neces-

sary to investigate the extent to which animals and humans can carry out goal-based planning at different scales, and whether the same planning mechanisms are used to do so. Ideally, such work would produce data that was directly comparable over these scales.

It is also necessary to investigate the role that frequently-used sequences play in planning: how the preference for familiar routes evolves over time and how adaptive and maladaptive sequences of actions are integrated into planning.

11

CONCLUSIONS

We began this project with three main interests: how to model the formation and usage of cognitive maps in the brain using neural mechanisms, how to do so whilst staying as close as possible to the neurobiology, and how to model a hierarchical mechanism for more efficient planning at larger spatial scales.

These areas had been the subject of previous modelling work, some of which we have reviewed in Ch. 2, but we felt that this work was limited in several important ways: there was little consensus on how these cognitive maps were represented and how they were formed, the planning mechanism that was used in these models seemed to be inherently limited to performing relatively simple tasks, and there had been little consideration of how these mechanisms would scale up beyond laboratory tasks.

In order to study these questions, we proposed a novel planning mechanism based on the principle that a propagating wave of goal-centred activity can carry information about the distance to the goal, originally proposed by Ponulak & Hopfield 2013 [3]. Unlike the model proposed by Ponulak et al., the mechanism that we proposed was able to work in a state-action map, able to output explicit actions to move towards the goal, able to perform planning without altering synaptic connectivity, able to plan in nondeterministic environments, and able to interface with a hierarchical behaviour mechanism. Furthermore, the planning mechanism that we proposed was not subject to the same limits as the decaying activation mechanism used by the models reviewed in Ch. 2. We studied what kind of neural mechanisms and connectivity were required to support the proposed mechanism and found that a particular cognitive map structure, consisting of state-action cells organised into minicolumns, was able to implement

this planning mechanism. We also found that a layer of gating cells was able to read off appropriate actions from this planning process.

We further found that all of these elements could be self-organised during unsupervised exploration of the environment whilst remaining reasonably constrained to the known neurobiology. In particular, we found that the lateral inhibition known to exist between neocortical minicolumns [95] allowed the model to produce 'state columns' which contain a set of state-action cells encoding different actions in the same state. We also showed how the read-out mechanism (the gating cells described above) could self-organise during the exploration process at the same time as the cognitive map. Finally we showed that a hierarchical planning mechanism could be implemented using a layer of sequence cells. This mechanism was able to learn frequently-used sequences during the planning process, and use them as mental 'shortcuts' to reduce the amount of planning required.

At present, there still remain many open questions about forming and using cognitive maps. On the theoretical level, there is the question of whether cognitive maps and model-based planning are necessary for planning at all. Contemporary experimental and modelling work [11, 12, 21] suggests that they are—the performance [12, 21] of rats and humans in detour task experiments cannot yet be replicated [11] by a purely model-free mechanism—but reinforcement learning is a fast-moving space and may advance in unexpected directions.

At the neural level, there are various proposed cellular substrates that might encode a cognitive map: place cells, state-action cells, and transition cells. There are also various proposed explanations for how these cells could self-organise. However, although these models propose that different cell types encode the cognitive map, they all agree that the recurrent connectivity between these cells is responsible for encoding individual state transitions within that map. They further agree that this recurrent connectivity is learned by a mechanism that joins cells which fire consecutively in time. Current models also agree to some extent on the planning mechanism that produces goal-directed actions using a cognitive map. They agree that this mechanism works by propagating activity through the recurrent connectivity that en-

codes the synaptic map, and that this activity can produce a gradient which can be climbed to reach the goal (see Sec. 10.2.2.2 for a more detailed description of this mechanism). However, they propose subtly different implementations of this planning mechanism, and a variety of mechanisms for extending it. The model that we propose agrees with some of the theories described above, and disagrees with others. It also provides novel alternative theories: for example, the idea that columnar WTA and bandstop mechanisms could cooperate to self-organise SA and gating cells, the probabilistic planning mechanism, and the formation and use of a sequence cell hierarchy.

All of these theories are in principle testable. However, although these theories make various predictions about elements of the exploration and planning process, most of these predictions are currently difficult to test. For example, the decaying activation planning mechanism and the propagating wavefront mechanism can be compared experimentally by investigating the planning time taken for tasks that require different numbers of actions. Unfortunately most experimental investigations of multi-step planning or navigation do not explicitly separate planning time from other processes, or seek to measure it, and so planning time is usually invisible.

Furthermore, in many cases the models and theories that have been advanced have not yet developed far enough to produce testable predictions. For example, two forms of hierarchical mechanism have been proposed: the low-resolution map described by Martinet 2011 and the sequence cells proposed in this thesis. Ideally, these would produce testable predictions about planning at different scales. However, the models themselves cannot at present encode large state spaces, so it is difficult to produce such predictions except speculatively. We hope that further modelling work will improve and extend both the model that we have proposed in this paper as well as other models in the field, and that these models will in turn suggest profitable avenues for further experimental work.

BIBLIOGRAPHY

[1] Ray J. Dolan and Peter Dayan. *Goals and habits in the brain*. 2013 (cit. on pp. v, 1, 3, 4, 11, 14, 21, 311).

[2] Louis-Emmanuel Martinet et al. 'Spatial Learning and Action Planning in a Prefrontal Cortical Network Model.' In: *PLoS Comput. Biol.* 7.5 (2011), e1002045 (cit. on pp. v, 7, 31, 37, 61, 62, 76, 109, 137, 185, 203–205, 208, 214, 221–227, 289, 294).

[3] Filip Ponulak and John J Hopfield. 'Rapid, parallel path planning by propagating wavefronts of spiking neural activity.' In: *Front. Comput. Neurosci.* 7 (2013), p. 98. arXiv: 1205.0335 (cit. on pp. v, 77, 219, 301, 339).

[4] Maya Geva-Sagiv et al. *Spatial cognition in bats and rats: From sensory acquisition to multiscale maps and navigation*. 2015 (cit. on pp. vi, 258, 295, 329, 335, 337).

[5] E. C. Tolman. 'Cognitive Maps in Rats and Men.' In: *Psychol. Rev.* 55.4 (1948), pp. 189–208 (cit. on pp. 1, 6, 18, 21, 31, 226, 294, 296, 335).

[6] E. C. Tolman. 'The determiners of behavior at a choice point.' In: *Psychol. Rev.* 45.1 (1938), pp. 1–41 (cit. on pp. 1, 7).

[7] EC Tolman, BF Ritchie, and D Kalish. 'Studies in Spatial Learning. I. orientation andthe short-cut.' In: *J. Exp. Psychol.* 36 (1946), pp. 13–24 (cit. on pp. 1, 7, 8).

[8] A. N. Hampton, P. Bossaerts, and J. P. O'Doherty. 'The Role of the Ventromedial Prefrontal Cortex in Abstract State-Based Inference during Decision Making in Humans.' In: *J. Neurosci.* 26.32 (2006), pp. 8360–8367 (cit. on pp. 1, 9).

[9] Zeb Kurth-Nelson et al. 'Fast Sequences of Non-spatial State Representations in Humans.' In: *Neuron* 91.1 (2016), pp. 194–204 (cit. on pp. 1, 9, 107, 307, 312).

[10] Dmitriy Aronov, Rhino Nevers, and David W. Tank. 'Mapping of a non-spatial dimension by the hippocampal–entorhinal circuit.' In: *Nature* 543.7647 (2017), pp. 719–722. arXiv: nature21692 [10.1038] (cit. on pp. 1, 10).

[11] Evan M. Russek et al. 'Predictive representations can link model-based reinforcement learning to model-free mechanisms.' In: *PLoS Comput. Biol.* 13.October (2016), p. 083857. arXiv: arXiv:1612.00429v2 (cit. on pp. 1, 18, 21, 22, 203, 286, 289, 298, 306, 340).

[12] Pegah Fakhari, Arash Khodadadi, and Jerome R. Busemeyer. 'The detour problem in a stochastic environment: Tolman revisited.' In: *Cogn. Psychol.* 101 (2018), pp. 29–49. arXiv: 1709.09761 (cit. on pp. 1, 8, 203, 289, 306, 340).

[13] Matthew M Botvinick, Yael Niv, and Andrew C Barto. 'Hierarchically organized behavior and its neural foundations: a reinforcement learning perspective.' In: *Cognition* 113.3 (2009), pp. 262–80 (cit. on pp. 2, 26, 27, 273).

[14] Pavan Ramkumar et al. 'Chunking as the result of an efficiency computation trade-off.' In: *Nat. Commun.* 7 (2016), p. 12176 (cit. on p. 2).

[15] P Ivan Pavlov. 'An investigation of the physiological activity of the cerebral cortex.' In: *Ann. Neurosci.* (1927) (cit. on pp. 3, 4).

[16] Edward L. Thorndike. 'Animal intelligence: An experimental study of the associative processes in animals.' In: *Psychol. Rev. Monogr. Suppl.* (1898). arXiv: arXiv:1011.1669v3 (cit. on pp. 3, 4).

[17] John M. Pearce. *Animal Learning and Cognition: An Introduction*. Psychology Press, 1997. Chap. 1 (cit. on p. 4).

[18] Wolfgang Kohler. *The Mentality of Apes*. Kegan Paul, Trench, Trübner & Co., 1925 (cit. on p. 5).

[19] E. C. Tolman. 'Purposive behavior.' In: *Psychol. Rev.* (1928) (cit. on p. 6).

[20] Francine Ciancia. 'Tolman and Honzik (1930) revisited: or The mazes of psychology (1930-1980).' In: *Psychol. Rec.* 41.4 (1991), pp. 461–472 (cit. on p. 8).

[21] Alice Alvernhe, Etienne Save, and Bruno Poucet. 'Local remapping of place cell firing in the Tolman detour task.' In: *Eur. J. Neurosci.* 33.9 (2011), pp. 1696–1705 (cit. on pp. 8, 31, 203, 204, 224, 226, 294, 335, 340).

[22] John O'Keefe and Lynn Nadel. *The Hippocampus as a Cognitive Map*. Oxford University Press, 1978 (cit. on pp. 8, 313).

[23] Torkel Hafting et al. 'Microstructure of a spatial map in the entorhinal cortex.' In: *Nature* 436.7052 (2005), pp. 801–806 (cit. on p. 8).

[24] Edmund T Rolls, Simon M Stringer, and Thomas Elliot. 'Entorhinal cortex grid cells can map to hippocampal place cells by competitive learning.' In: *Network* 17.4 (2006), pp. 447–65 (cit. on pp. 8, 319).

[25] Bruce L. McNaughton et al. 'Path integration and the neural basis of the 'cognitive map'.' In: *Nat. Rev. Neurosci.* 7.8 (2006), pp. 663–678 (cit. on pp. 8, 319).

[26] A. Johnson and A. D. Redish. 'Neural Ensembles in CA3 Transiently Encode Paths Forward of the Animal at a Decision Point.' In: *J. Neurosci.* 27.45 (2007), pp. 12176–12189 (cit. on pp. 8, 9, 63, 107, 308, 312).

[27] Fabian Chersi and Giovanni Pezzulo. 'Using hippocampal-striatal loops for spatial navigation and goal-directed decision-making.' In: *Cogn. Process.* 13.1 SUPPL (2012), pp. 125–129 (cit. on pp. 9, 63, 107, 308, 313).

[28] Mehdi Keramati, Amir Dezfouli, and Payam Piray. 'Speed/Accuracy Trade-Off between the Habitual and the Goal-Directed Processes.' In: *PLoS Comput. Biol.* 7.5 (2011). Ed. by Tim Behrens, e1002055 (cit. on pp. 11, 311).

[29] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. 1998, p. 331 (cit. on pp. 13, 14).

[30] J. Andrew Bagnell. 'Reinforcement Learning in Robotics: A Survey.' In: *Springer Tracts Adv. Robot.* 97 (2014), pp. 9–67 (cit. on p. 14).

[31] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. 'Reinforcement learning: A survey.' In: *J. Artif. Intell. Res.* 4 (1996), pp. 237–285. arXiv: 9605103 [cs] (cit. on pp. 14, 16).

[32] P. R. Kumar and Pravian Varaiya. *Stochastic Systems: Estimation, Identification and Adaptive Control*. Ed. by Gloria Jordan. Prentice-Hall, Inc., 1986 (cit. on p. 15).

[33] Christopher J. C. H. Watkins and Peter Dayan. 'Q-learning.' In: *Mach. Learn.* 8.3-4 (1992), pp. 279–292 (cit. on p. 16).

[34] Paul L. Brown and Herbert M. Jenkins. 'Auto-shaping of the pigeon's key-peck.' In: *J. Exp. Anal. Behav.* (1968) (cit. on p. 17).

[35] Volodymyr Mnih et al. 'Human-level control through deep reinforcement learning.' In: *Nature* 518.7540 (2015), pp. 529–533. arXiv: 1312.5602 (cit. on pp. 18, 19, 286).

[36] Yuxi Li. 'Deep Reinforcement Learning: An Overview.' In: (2017), pp. 1–70. arXiv: 1701.07274 (cit. on pp. 18–20).

[37] Hado van Hasselt, Arthur Guez, and David Silver. 'Deep Reinforcement Learning with Double Q-learning.' In: (2015). arXiv: 1509.06461 (cit. on p. 19).

[38] Tom Schaul et al. 'Universal Value Function Approximators.' In: *Proc. 32nd Int. Conf. Mach. Learn.* 2015, pp. 1312–1320 (cit. on p. 19).

[39] Volodymyr Mnih et al. 'Asynchronous Methods for Deep Reinforcement Learning.' In: *arXiv* 48 (2016), pp. 1–28. arXiv: 1602.01783 (cit. on p. 19).

[40] John Schulman et al. 'Proximal Policy Optimization Algorithms.' In: *Curr. Opin. Neurobiol.* 17.4 (2017), pp. 456–464. arXiv: 1707.06347 (cit. on p. 19).

[41] Peter Henderson et al. 'Deep Reinforcement Learning that Matters.' In: (2017). arXiv: 1709.06560 (cit. on p. 20).

[42] Alex Irpan. *Deep Reinforcement Learning Doesn't Work Yet*. 2018 (cit. on pp. 20, 286).

[43] Anusha Nagabandi et al. 'Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning.' In: (2017). arXiv: 1708.02596 (cit. on pp. 20, 311).

[44] Yevgen Chebotar et al. 'Combining Model-Based and Model-Free Updates for Trajectory-Centric Reinforcement Learning.' In: *CEUR Workshop Proc.* 1680 (2017), pp. 60–66. arXiv: 1703.03078 (cit. on pp. 20, 311).

[45] Kimberly L. Stachenfeld, Matthew M. Botvinick, and Samuel J. Gershman. 'The hippocampus as a predictive map.' In: *Nat. Neurosci.* 28.3 (2017), pp. 391–397. arXiv: 097170 [10.1101] (cit. on p. 21).

[46] Peter Dayan. 'Improving Generalization for Temporal Difference Learning: The Successor Representation.' In: *Neural Comput.* 5.4 (1993), pp. 613–624 (cit. on p. 21).

[47] Kl L Stachenfeld, M M Botvinick, and S J Gershman. 'Design Principles of the Hippocampal Cognitive Map.' In: *Adv. Neural Inf. Process. Syst.* 27. 2014, pp. 1–9 (cit. on pp. 21, 22).

[48] D. A. Simon and N. D. Daw. 'Neural Correlates of Forward Planning in a Spatial Decision Task in Humans.' In: *J. Neurosci.* 31.14 (2011), pp. 5526–5539 (cit. on pp. 22, 203).

[49] John C Hirtle, J; Jonides. 'Evidence of hierarchcies in cognitive maps.' In: *Mem. Cogn.* 13.3 (1985), pp. 208–217 (cit. on pp. 22, 23, 34, 59, 229, 258, 288).

[50] Christoph Hölscher et al. 'Adaptivity of wayfinding strategies in a multi-building ensemble: The effects of spatial structure, task requirements, and metric information.' In: *J. Environ. Psychol.* 29.2 (2008), pp. 208–219 (cit. on pp. 23, 34, 229, 288).

[51] Hannes Noack et al. 'Sleep enhances knowledge of routes and regions in spatial environments.' In: *Learn. Mem.* 24.3 (2017), pp. 140–144 (cit. on pp. 23, 34, 229, 288).

[52] Alexander Klippel, Heike Tappe, and Christopher Habel. 'Pictorial representations of routes: Chunking route segments during comprehension.' In: *Spat. Cogn. III* (2003), pp. 11–33 (cit. on pp. 23, 34, 229, 230, 257–259, 288).

[53] Martin Tomko, Stephan Winter, and Christophe Claramunt. 'Experiential hierarchies of streets.' In: *Comput. Environ. Urban Syst.* 32.1 (2008), pp. 41–52 (cit. on p. 23).

[54] S Timpf and W Kuhn. 'Granularity transformations in wayfinding.' In: *Lect. Notes Artif. Intell. (Subseries Lect. Notes Comput. Sci.* 2685 (2003), pp. 77–88 (cit. on pp. 23, 34, 229, 230, 258, 288).

[55] J.M. Jan M Wiener and Hanspeter A. Mallot. ''Fine-to-Coarse' Route Planning and Navigation in Regionalized Environments.' In: *Spat. Cogn. Comput.* 3.4 (2003), pp. 331–358 (cit. on pp. 23, 30, 34, 229, 230, 258, 288).

[56] Rashmi Premnath Payyanadan. 'UNDERSTANDING THE INFLUENCE OF FAMILIARITY ON ROUTE CHOICE AMONG OLDER DRIVERS.' PhD thesis. 2018 (cit. on pp. 23, 229, 231, 256, 333).

[57] Lorelei R. Howard et al. 'The hippocampus and entorhinal cortex encode the path and euclidean distances to goals during navigation.' In: *Curr. Biol.* 24.12 (2014), pp. 1331–1340 (cit. on pp. 24, 67, 230, 258, 279, 307, 309, 330, 335, 336).

[58] Paul Cisek, Donald J Crammond, and John F Kalaska. 'Neural activity in primary motor and dorsal premotor cortex in reaching tasks with the contralateral versus ipsilateral arm.' In: *J. Neurophysiol.* 89.2 (2003), pp. 922–42 (cit. on pp. 24, 107, 323).

[59] S.M. Stringer and E.T. Rolls. 'Hierarchical dynamical models of motor function.' In: *Neurocomputing* 70.4-6 (2007), pp. 975–990 (cit. on p. 24).

[60] Kiyoshi Kurata. 'Hierarchical organization within the ventral premotor cortex of the macaque monkey.' In: *Neuroscience* May (2018) (cit. on p. 24).

[61] David Abel, D. Ellis Hershkowitz, and Michael L. Littman. 'Near Optimal Behavior via Approximate State Abstraction.' In: 48 (2017), pp. 1–18. arXiv: 1701.04113 (cit. on p. 25).

[62] Thomas G Dietterich. 'The MAXQ Method for Hierarchical Reinforcement Learning.' In: *Proc. fifteenth Int. Conf. Mach. Learn.* (1998), pp. 118–126 (cit. on p. 25).

[63] Stefan Elfwing et al. 'Evolutionary development of hierarchical learning structures.' In: *IEEE Trans. Evol. Comput.* 11.2 (2007), pp. 249–264 (cit. on p. 25).

[64] Neville Mehta, S Ray, and P Tadepalli. 'Automatic discovery and transfer of MAXQ hierarchies.' In: *Proc. 25th* (2008), pp. 648–655 (cit. on p. 26).

[65] Richard S. Sutton, NNOdoina Precup, and Satinder Singh. 'Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning.' In: *Artif. Intell.* 112.1-2 (1999), pp. 181–211. arXiv: arXiv:1011.1669v3 (cit. on pp. 26, 27).

[66] Nasrin Taghizadeh and Hamid Beigy. 'A novel graphical approach to automatic abstraction in reinforcement learning.' In: *Rob. Auton. Syst.* 61.8 (2013), pp. 821–835 (cit. on pp. 28, 266).

[67] Tejas D. Kulkarni et al. 'Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation.' In: *ArXiv* (2016), pp. 1–13. arXiv: 1604.06057 (cit. on p. 28).

[68] Chen Tessler et al. 'A Deep Hierarchical Approach to Lifelong Learning in Minecraft.' In: (2016), pp. 1553–1561. arXiv: 1604.07255 (cit. on p. 28).

[69] Alexander Sasha Vezhnevets et al. 'FeUdal Networks for Hierarchical Reinforcement Learning.' In: (2017). arXiv: 1703.01161 (cit. on p. 28).

[70] Matthew Michael Botvinick. *Hierarchical reinforcement learning and decision making*. 2012 (cit. on p. 28).

[71] Carlos Florensa, Yan Duan, and Pieter Abbeel. 'Stochastic Neural Networks for Hierarchical Reinforcement Learning.' In: (2017), pp. 1–17. arXiv: 1704.03012 (cit. on pp. 28, 29).

[72] S. Girgin and F. Polat. 'Option Discovery in Reinforcement Learning using Frequent Common Subsequences of Actions.' In: *Int. Conf. Comput. Intell. Model. Control Autom. Int. Conf. Intell. Agents, Web Technol. Internet Commer.* Vol. 1. 2005, pp. 371–376 (cit. on p. 28).

[73] Sertan Girgin, Faruk Polat, and Reda Alhajj. 'Learning by automatic option discovery from conditionally terminating sequences.' In: *Front. Artif. Intell. Appl.* 141 (2006), pp. 494–498 (cit. on p. 29).

[74] Sertan Girgin, Faruk Polat, and Reda Alhajj. 'Improving reinforcement learning by using sequence trees.' In: *Mach. Learn.* 81.3 (2010), pp. 283–331 (cit. on p. 29).

[75] Marc Pickett and Andrew G Barto. 'PolicyBlocks: An Algorithm for Creating Useful Macro-Actions in Reinforcement Learning.' In: *Proc. 19th Int. Conf. Mach. Learn.* August (2002), pp. 506–513. arXiv: arXiv:1011.1669v3 (cit. on pp. 29, 273).

[76] Roland Siegwart, Illah R Nourbakhsh, and Davide Scaramuzza. *Introduction to Autonomous Mobile Robots (2nd Edition)*. MIT Press, 2011 (cit. on p. 30).

[77] Jean Arcady Meyer and David Filliat. *Map-based navigation in mobile robots: II. A review of map-learning and path-planning strategies*. 2003 (cit. on p. 30).

[78] E W Dijkstra. 'A note on two problems in connexion with graphs.' In: *Numer. Math.* 1.1 (1959), pp. 269–271 (cit. on p. 34).

[79] Claire Stansfield, Alison O'Mara-Eves, and James Thomas. 'Text mining for search term development in systematic reviewing: A discussion of some methods and challenges.' In: *Res. Synth. Methods* 8.3 (2017), pp. 355–365 (cit. on p. 36).

[80] Nicolas Cuperlier, Mathias Quoy, and Philippe Gaussier. 'Neurobiologically inspired mobile robot navigation and planning.' In: *Front. Neurorobot.* 1.NOV (2007) (cit. on pp. 37, 38, 40, 185).

[81] Michael E Hasselmo. 'A model of prefrontal cortical mechanisms for goal-directed behavior.' In: *J. Cogn. Neurosci.* 17.7 (2005), pp. 1115–1129 (cit. on pp. 37, 109, 137, 185).

[82] U??ur M. Erdem and Michael Hasselmo. 'A goal-directed spatial navigation model using forward trajectory planning based on grid cells.' In: *Eur. J. Neurosci.* 35.6 (2012), pp. 916–931 (cit. on pp. 37, 52).

[83] J Friedrich and M. Lengyel. 'Goal-Directed Decision Making with Spiking Neurons.' In: *J. Neurosci.* 36.5 (2016), pp. 1529–1546 (cit. on pp. 37, 215).

[84] Michael L. Littman. 'A tutorial on partially observable Markov decision processes.' In: *J. Math. Psychol.* 53.3 (2009), pp. 119–125 (cit. on p. 41).

[85] Hector Geffner. 'Computational models of planning.' In: *WIREs Cogn. Sci.* 4.4 (2013). Ed. by Asmuth Albore Astrom, Attias, Bacchus, Backstrom, Ballard, Barto, Bellman, Bertsekas, Biere, Blum, Bonet, Bonet, Bonet, Bonet, Bonet, Bonet, Bonet, Botvinick, Boutilier, Bowling, Brafman, Brafman, Brooks, Bylander, Camerer, Chapman, Coles, Cormen, Damasio, Denn, pp. 341–356 (cit. on pp. 41, 305).

[86] Horatiu Voicu and Nestor Schmajuk. 'Latent learning, shortcuts and detours: A computational model.' In: *Behav. Processes* 59.2 (2002), pp. 67–86 (cit. on p. 57).

[87] Jumpei Matsumoto et al. 'A computational model of the hippocampus that represents environmental structure and goal location, and guides movement.' In: *Biol. Cybern.* 105.2 (2011), pp. 139–152 (cit. on p. 64).

[88] Jonathan D. Wallis, Kathleen C. Anderson, and Earl K. Miller. 'Single neurons in prefrontal cortex encode abstract roles.' In:

Nature 411.6840 (2001), pp. 953–956 (cit. on pp. 73, 107, 137, 313, 318, 323).

[89] Jun Tanji and Eiji Hoshi. 'Role of the lateral prefrontal cortex in executive behavioral control.' In: *Physiol. Rev.* 88.140 (2008), pp. 37–57 (cit. on p. 73).

[90] Joaquın M Fuster. 'The prefrontal cortex–an update: time is of the essence.' In: *Neuron* 30.2 (2001), pp. 319–33 (cit. on p. 73).

[91] W F Asaad, G Rainer, and E K Miller. 'Neural activity in the primate prefrontal cortex during associative learning.' In: *Neuron* 21.6 (1998), pp. 1399–1407 (cit. on pp. 73, 107, 318, 323).

[92] Robert C. Wilson et al. 'Orbitofrontal cortex as a cognitive map of task space.' In: *Neuron* 81.2 (2014), pp. 267–278. arXiv: NIHMS150003 (cit. on pp. 73, 107, 313, 318, 323).

[93] Nicolas W. Schuck et al. 'Human Orbitofrontal Cortex Represents a Cognitive Map of State Space.' In: *Neuron* 91.6 (2016), pp. 1402–1412 (cit. on pp. 73, 107, 313, 318, 323).

[94] Ramon Nogueira et al. 'Lateral orbitofrontal cortex anticipates choices and integrates prior with current information.' In: *Nat. Commun.* 8 (2017), p. 14823 (cit. on pp. 73, 107, 137, 313, 318, 323).

[95] D. P. Buxhoeveden. 'The minicolumn hypothesis in neuroscience.' In: *Brain* 125.5 (2002), pp. 935–951 (cit. on pp. 73, 124, 147, 326, 340).

[96] Wolfgang Maass. 'Noise as a resource for computation and learning in networks of spiking neurons.' In: *Proc. IEEE* 102.5 (2014), pp. 860–880 (cit. on pp. 90, 92, 216, 309).

[97] T. P. Vogels et al. 'Inhibitory plasticity balances excitation and inhibition in sensory pathways and memory networks.' In: *Science (80-.).* 334.6062 (2011), pp. 1569–1573. arXiv: 20 (cit. on pp. 92, 202).

[98] Richard G.M. Morris. 'Spatial localization does not require the presence of local cues.' In: *Learn. Motiv.* 12.2 (1981), pp. 239–260 (cit. on pp. 106, 335).

[99] Richard Cooper and Tim Shallice. 'Contention scheduling and the control of routine activities.' In: *Cogn. Neuropsychol.* 17.4 (2000), pp. 297–338 (cit. on p. 106).

[100] Michael J. Frank and David Badre. *Mechanisms of hierarchical reinforcement learning in corticostriatal circuits 1: Computational analysis*. 2012 (cit. on pp. 107, 323).

[101] Peter Dayan and L F Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. The MIT Press, 2005, p. 480. arXiv: 0-262-04199-5 (cit. on pp. 110, 285).

[102] Yanqing Chen, Jeffrey L. McKinstry, and Gerald M. Edelman. 'Versatile networks of simulated spiking neurons displaying winner-take-all behavior.' In: *Front. Comput. Neurosci.* 7 (2013) (cit. on p. 110).

[103] Yanqing Chen. 'Mechanisms of Winner-Take-All and Group Selection in Neuronal Spiking Networks.' In: *Front. Comput. Neurosci.* 11 (2017) (cit. on p. 110).

[104] D. M. Walters and S. M. Stringer. 'Path integration of head direction: Updating a packet of neural activity at the correct speed using neuronal time constants.' In: *Biol. Cybern.* (2010) (cit. on pp. 111, 120, 121).

[105] Brian M Adams et al. *Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.6 User's Manual*. 2014 (cit. on pp. 119, 136).

[106] James M. Tromans, Irina Higgins, and Simon M. Stringer. *Learning view invariant recognition with partially occluded objects*. 2012 (cit. on p. 120).

[107] Peiran Gao et al. 'A theory of multineuronal dimensionality, dynamics and measurement.' In: *bioRxiv* (2017), p. 214262. arXiv: bioRxiv214262 (cit. on pp. 137, 316).

[108] Henry Markram et al. 'Interneurons of the neocortical inhibitory system.' In: *Nat. Rev. Neurosci.* 5 (2004), pp. 793–807 (cit. on p. 147).

[109] Henning Sprekeler. *Functional consequences of inhibitory plasticity: homeostasis, the excitation-inhibition balance and beyond*. 2017 (cit. on p. 147).

[110] Claudia Clopath et al. *Receptive field formation by interacting excitatory and inhibitory synaptic plasticity*. Tech. rep. 2016, p. 066589 (cit. on p. 147).

[111] E T Rolls and T Milward. 'A model of invariant object recognition in the visual system: learning rules, activation functions, lateral inhibition, and information-based performance measures.' In: *Neural Comput.* 12 (2000), pp. 2547–2572 (cit. on p. 159).

[112] Wikpedia. *Precision and Recall*. 2018 (cit. on p. 165).

[113] David L. Olson and Dursun. Delen. *Advanced data mining techniques*. Springer, 2008, p. 138 (cit. on p. 165).

[114] Thomas Rogerson et al. 'Synaptic tagging during memory allocation.' In: *Nat. Rev. Neurosci.* 15.3 (2014), pp. 157–169. arXiv: NIHMS150003 (cit. on pp. 166, 168, 182, 265, 327).

[115] Nuo Li and James J. DiCarlo. 'Unsupervised natural experience rapidly alters invariant object representation in visual cortex.' In: *Science* 321.5895 (2008), pp. 1502–1507. arXiv: 10.1126 (cit. on p. 168).

[116] Benjamin D. Evans and Simon M. Stringer. 'Transformation-invariant visual representations in self-organizing spiking neural networks.' In: *Front. Comput. Neurosci.* 6.July (2012), pp. 1–19. arXiv: NIHMS150003 (cit. on pp. 168, 182).

[117] Gavin Perry, Edmund T. Rolls, and Simon M. Stringer. 'Spatial vs temporal continuity in view invariant visual object recognition learning.' In: *Vision Res.* (2006) (cit. on p. 168).

[118] L. F. Abbott and Wade G. Regehr. 'Synaptic computation.' In: *Nature* 431.7010 (2004), pp. 796–803 (cit. on p. 216).

[119] E C Tolman and C H Honzik. '"Insight" in rats.' In: *Univ. Calif. Publ. Psychol.* 4.14 (1930), pp. 215–232 (cit. on p. 227).

[120] Tad T. Brunyé et al. 'The path more travelled: Time pressure increases reliance on familiar route-based strategies during navigation.' In: *Q. J. Exp. Psychol.* 70.8 (2017), pp. 1439–1452 (cit. on pp. 231, 256, 280, 283, 333).

[121] Hugo J Spiers and Eleanor A Maguire. 'The dynamic nature of cognition during wayfinding.' In: *J. Environ. Psychol.* 28.3 (2008), pp. 232–249 (cit. on pp. 256, 258, 308, 333, 337).

[122] Benjamin Kuipers, Dan G. Tecuci, and Brian J. Stankiewicz. 'The skeleton in the cognitive map: A computational and empirical exploration.' In: *Environ. Behav.* 35.1 (2003), pp. 81–106 (cit. on p. 256).

[123] Geoff Ward and Alan Allport. 'Planning and Problem-solving Using the Five-disc Tower of London Task.' In: 1 (1997) (cit. on pp. 257, 307, 309, 310, 330, 331, 334–336).

[124] Nestor Schmajuk and Horatiu Voicu. 'Exploration and Navigation Using Hierarchical Cognitive Maps.' In: *Comp. Neural Comput. Approaches*. 2006 (cit. on pp. 257, 258).

[125] Alexander W . Siegel and Sheldon H . White. 'the Development of Spatial Representations of Large-Scale Environments.' In: *Adv. Child Dev. Behav.* 10 (1975), pp. 9–55 (cit. on pp. 258, 259).

[126] Keisetsu Shima and Jun Tanji. 'Neuronal activity in the supplementary and presupplementary motor areas for temporal organization of multiple movements.' In: *J. Neurophysiol.* 84.4 (2000), pp. 2148–2160 (cit. on pp. 259–262, 335).

[127] Giuseppe Luppino et al. 'Corticocortical connections of area F3 (SMA-proper) and area F6 (pre-SMA) in the macaque monkey.' In: *J. Comp. Neurol.* 338.1 (1993), pp. 114–140. arXiv: arXiv:1011.1669v3 (cit. on p. 259).

[128] Neil Burgess, Michael Recce, and John O'Keefe. 'A model of hippocampal function.' In: *Neural Networks* 7.6-7 (1994), pp. 1065–1081 (cit. on p. 307).

[129] F. Cazalis et al. 'Individual differences in prefrontal cortical activation on the Tower of London planning task: Implication for effortful processing.' In: *Eur. J. Neurosci.* 17.10 (2003), pp. 2219–2225 (cit. on pp. 307, 309, 310, 335).

[130] Gabrielle Girardeau et al. 'Selective suppression of hippocampal ripples impairs spatial memory.' In: *Nat. Neurosci.* (2009) (cit. on p. 308).

[131] Valérie Ego-Stengel and Matthew A. Wilson. 'Disruption of ripple-associated hippocampal activity during rest impairs spatial learning in the rat.' In: *Hippocampus* (2010) (cit. on p. 308).

[132] A. Aldo Faisal, Luc P.J. Selen, and Daniel M. Wolpert. 'Noise in the nervous system.' In: *Nat. Rev. Neurosci.* 9.4 (2008), pp. 292–303. arXiv: 1609.08144 (cit. on p. 309).

[133] Michael London et al. 'Sensitivity to perturbations in vivo implies high noise and suggests rate coding in cortex.' In: *Nature* 466.7302 (2010), pp. 123–127 (cit. on p. 309).

[134] Francis Crick. *The recent excitement about neural networks*. 1989 (cit. on p. 314).

[135] Edmund T Rolls and Allesandro Treves. *Neural Networks and Brain Function*. 1st ed. Oxford University Press, 1998, p. 15 (cit. on p. 314).

[136] Adam Henry Marblestone, Greg Wayne, and Konrad P Kording. *Towards an integration of deep learning and neuroscience*. Tech. rep. 2016 (cit. on p. 314).

[137] Pieter R. Roelfsema and Anthony Holtmaat. 'Control of synaptic plasticity in deep cortical networks.' In: *Nat. Rev. Neurosci.* 19.3 (2018), pp. 166–180 (cit. on p. 314).

[138] Vernon B. Mountcastle. *The columnar organization of the neocortex*. 1997 (cit. on p. 326).

[139] R. Becket Ebitz, Eddy Albarran, and Tirin Moore. 'Exploration Disrupts Choice-Predictive Signals and Alters Dynamics in Prefrontal Cortex.' In: *Neuron* (2017) (cit. on p. 328).

[140] Máté Lengyel, Zsófia Huhn, and Péter Érdi. 'Computational theories on the function of theta oscillations.' In: *Biol. Cybern.* 92.6 (2005), pp. 393–408 (cit. on p. 328).

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography *"The Elements of Typographic Style"*. `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured at:

http://postcards.miede.de/

Final Version as of September 12, 2019 (`classicthesis` submission version).