Relational representations for composition of knowledge and plans



Jacob J. W. Bakermans

Wadham College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hilary 2022

Acknowledgements

This DPhil has been an amazing journey, thanks to the wonderful people that I have had the luxury of being around.

First, I would like to thank Tim Behrens for his brilliant supervision. Tim cares deeply about science and about his people. He has set an example for me of doing creative yet rigorous science, while providing uncompromising personal support. I have learned a lot from him and I had a great time doing so. I would also like to thank Mark Woolrich for an exciting rotation and for his support throughout the DPhil.

Tim's lab is a stimulating environment. I feel fortunate to have had Anna Shpektor and Alon Baram by my side for the whole DPhil. Their company has been invaluable, and the conversations, meals, and trips that we shared are a big part of what made the journey so enjoyable. Their wisdom and encouragement, in smooth and challenging times, have meant a lot to me. James Whittington and Tim Muller have been generous with advice and ideas as well as friendship. Thanks to Veronika, David, Beatriz, Avital, Mohamady, Yves, Svenja, Philipp, Thomas, Cam, Shirley, Emma, and all the others hanging out in and with the lab for making it such a fun and inspiring place.

Oxford feels like home, and many people have contributed to that. Thanks to my Wadham friends and housemates for sharing many happy moments: Gareth, Rory, Leonie, Kathi, Jonny, Elise, Brooke, and Luana. Thanks to the great final Systems Biology DTC cohort for providing the best possible start to the DPhil – a special thanks to squash buds Rohan, Ravinda, and Lance for keeping me fit. Thijs's arrival to Oxford brought a seamless continuation of an old friendship and many chats about life, science, and bicycles.

During my time in Oxford I have felt closely connected to my Leiden friends. Thanks to Jos for his warmth and enthusiasm, and to Thed and Mel for always catching up. I am very grateful to my friends from Ἄτλας: Sam, Elger, Chris, Tim, Coen, Patrick, Robert, Martijn, Jos, Peter, Ruben, Milan, Joris, Thom, and Rik. They have provided much needed distractions from a distance, and when reunited we pick up right where we left off.

Special thanks go out to Martijn for his intellectual and moral support at every step along the way. His visits are among the highlights of my DPhil, he is a great listener and a careful thinker – I am lucky to have him around.

My family has been a constant source of reassurance. Thanks to papa, mama, Maartje, and Anne for their continuous interest, encouragement, and unconditional support. They provide me with a solid base that I can always fall back on.

Finally, thank you Marianne for embarking on this adventure with me. Thank you for your patience, your humour, your intelligence, your kindness. Life is better with you.

Abstract

Humans have a remarkable capacity to understand and act in situations they have not encountered before. This capacity relies on clever representations of what is common across experience. Relational representations are particularly powerful for this purpose, because they allow for composing novel situations from familiar building blocks. In this thesis, I use neuroimaging and computational modelling to investigate how the brain constructs knowledge and plans.

In a functional magnetic resonance imaging experiment I study the origin of knowledge about objects and situations. Bar stools look like tables, but are conceptually more similar to sofas. What is the source of these abstract similarities? I hypothesise that the compositional regularities of experience induce meaning in objects and situations, reflected in brain activity along the temporal to the frontal lobe. To test this hypothesis, I designed a paradigm that probes neural representations from shape to understanding. In a controlled experiment with precise quantitative neural predictions, I do not find convincing evidence of the hypothesised abstract representations in the brain.

Through a computational model I propose a mechanism for constructing plans. When navigating a new city, we know that to visit a sight on the other side of the river, we have to find a bridge first. I show how building cognitive maps from relative positions of such common structural elements allows to immediately reuse previously learned plans. Object vector cells in entorhinal cortex are an attractive neural substrate to represent these relative positions, because they imply actions and path integrate. I propose a central role for replay in constructing the compositional map in hippocampus. The resulting model generalises plans, suppresses path integration noise, and extends naturally to hierarchical tasks.

This thesis thus suggests a central role for relational representations and composition in making sense of the world and making decisions in it.

Contents

1 Introduction **1**

1.1 Reinforcement learning 3

1.1.1 Psychology: conditioning 3

1.1.2 Theory: Markov decision processes 5

1.1.3 Neuroscience: reinforcement learning in the brain 11

1.2 Cognitive maps 14

1.2.1 Psychology: latent learning 15

1.2.2 Theory: model based decisions 17

1.2.3 Neuroscience: maps and states 19

1.3 Relations and composition 23

1.3.1 Relational memory 23

1.3.2 Compositionality 28

1.4 Thesis outline: states and spaces 32

2 From shape to understanding **35**

2.1 Intermezzo: methods for efficient design 40

2.1.1 Balancing stimulus precedence 41

2.1.2 Bias in representational similarity analysis 45

2.2 Methods 49

2.2.1 Task 51

2.2.2 Data 59

2.2.3 Analysis 63

2.3 Results 72

2.3.1 Behaviour 73

2.3.2 Repetition suppression 75

2.3.3 Representational similarity 76

2.3.4 Encoding 80

2.4 Discussion 84

3 Planning through composition **93**
3.1 Constructing behaviour in structured environments 99
3.1.1 Methods 101
3.1.2 Results 111
3.1.3 Discussion 119
3.2 Hierarchical representation and replay 123
3.2.1 Methods 125
3.2.2 Results 129
3.2.3 Discussion 130
3.3 Action prediction yields vector representations 132
3.3.1 Methods 134
3.3.2 Results 139
3.3.3 Discussion 141
3.4 Action embedding from structure 143
3.4.1 Methods 144
3.4.2 Results 148
3.4.3 Discussion 150

4 Conclusion **151**

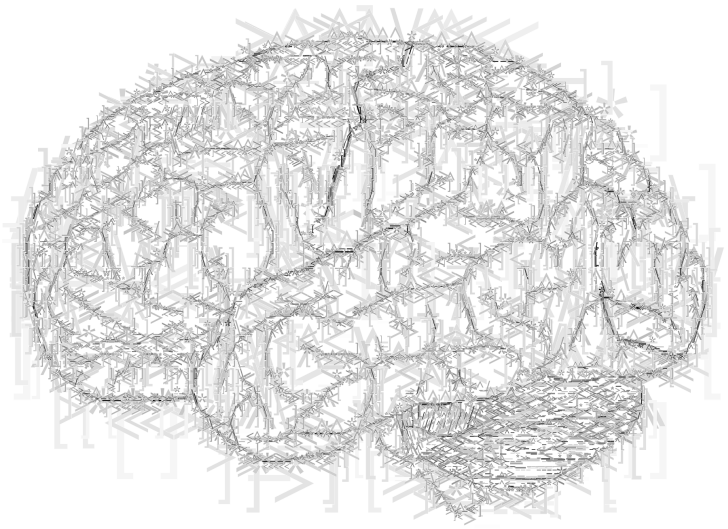
Appendices

Neuropoetry **159**

References **171**

1
Introduction

When you plan your day, you consider all the things you need to do: a shower, meetings, shopping, dinner, and hopefully a nice book or a game of football somewhere in between as well. You can imagine making a plan as finding a route passing through all activities, turning your to-do list into a to-do map. Like lists, maps can contain many items. But unlike lists, maps also tell you how these items are related. Additionally, if you know how a new item relates to existing items, you can construct a map item by item. This thesis argues that the brain organises experience like maps rather than lists.



Brain composed of relational symbols. ASCII-style art, own work, custom code.

Firstly, I hypothesise that situations themselves (like the football game) are like maps, built from objects (like a goalkeeper and goal) in particular relations (the goalkeeper shields the goal) – and that the brain learns the meaning of situations and their constituent objects from such maps. Secondly, I propose how building maps between situations (like the to-do map) is useful for learning to reach a goal in many different map configurations.

This chapter lays down the central question of this thesis: how does the brain use relational and compositional representations of states and spaces to understand and navigate the world? Here *relational* means *containing information about the items themselves and how they relate to another*, and *compositional* means *constructed by combining reusable building blocks*.

I start from the perspective of reinforcement learning (RL) in Section 1.1. Reinforcement learning tackles the problem of figuring out how to act to obtain reward. That is a hard problem: success is often the result of a series of actions, so how can you know which of them were crucial? I turn to the theory of Markov Decision Processes to provide a formal description and solution of this problem. This theory divides the world into states and actions and learns from experience how valuable each of these states and actions is in the long term.

However, particularly in situations that require adaptivity such as changing environments, such solutions do not account for all decision making behaviour. These situations demand an understanding about how the world works – often framed as 'cognitive maps' (Section 1.2). The cognitive map relies on the same states as RL, but links states up by their transitions into a network, which allows for reasoning about long term consequences of actions without having to experience them.

These cognitive maps are particularly powerful because they allow for generalisation: applying old knowledge to new problems. Section 1.3 presents two key ingredients that support generalisation: relational memory, for organising

experience in networks of relationships, and compositionality, to construct concepts and structures from building blocks.

This leads to Section 1.4 that introduces the core chapters of the thesis: composition *within* states for understanding the world, and *between* states for navigating the world. Chapter 2 presents how relations *within* states allow for construction of situations and stories. It describes a neuroimaging experiment that measures how this composition induces meaning in objects and situations, and probes corresponding neural representations. Chapter 3 focuses on relations *between* states in order to construct state spaces for navigation. By building state spaces from reusable building blocks, it is possible to know how to act in new situations without any additional learning, because appropriate actions can be pre-learned and then generalised for those building blocks.

1.1 Reinforcement learning

Reinforcement learning (RL) is learning how to act in order to maximise reward. There are no instructions: successful solutions need to be discovered from trying things out. RL has proven to be a highly successful framework for describing learning and decision making. Its roots lie in conditioning experiments from psychology, where stimuli or actions are predictably paired to rewards or punishments to reinforce behaviours. Since then, RL has been further formalised in terms of *Markov Decision Processes* (MDPs). Applying the precise mathematical language of MDPs and their algorithmic solutions has allowed for identifying substrates of RL computations in the brain.

1.1.1 Psychology: conditioning

Russian physiologist Ivan Pavlov's famous experiments are an example of *classical conditioning* (Pavlov 1927). Pavlov would precede feeding his dogs by the sound of a metronome, and over time the dogs started to salivate upon hearing the metronome's sound. Their innate response to food could be evoked by a stimulus that originally had nothing to do with food. In classical conditioning,

the reflexive salivation is called the *unconditioned response* (UR) to the naturally triggering *unconditioned stimulus* (US) food. Through repeatedly pairing the US to the *conditioned stimulus* (CS) sound, the animal learns that the CS predicts the US and produces the *conditioned response* (CR) by salivating on the sound. Often it is beneficial for an animal to have a CR in anticipation of the US: salivating before food arrives may improve digestion, and – in other experiments – closing an eyelid before instead of in response to eye irritation may avoid the irritation at all.

Since Pavlov, additional conditioning paradigms have further nuanced the relation between US, CS, and CR. In order to evoke CRs, the CS must reliably predict the US (Rescorla 1967). CRs appear not because of *pairing* of US and CS, but because of their *contingency*: random CSs, that sometimes pair with a US but equally often do not, will not produce CRs (Rescorla 1966). Moreover, the CS needs to have additional predictive value: to get CRs, the US must be surprising. In a *blocking* paradigm, the animal first learns to produce CRs to US-predicting CS A, and from that point CS A is presented together with CS B. However, since CS A already fully predicts the US, CS B will not evoke CRs (Kamin 1969). If instead CS A and CS B are presented simultaneously from the beginning, an *overshadowing* effect occurs: presenting CS A or CS B individually evokes CRs, but a higher CR rate for A means a lower CR rate for B and vice versa – as if the two compete (Reynolds 1961). Taken together these results suggest that animals use the predictive properties of the CS to learn expectations about the US.

In all classical conditioning experiments described so far, reward was contingent on stimuli, and behaviour was reflexive. Behaviour may be reinforced, but the reward does not actually depend on behaviour. *Instrumental conditioning* (or *operant conditioning*) describes learning when outcomes depend on actions. The American psychologist Edward Thorndike, pioneer of instrumental conditioning, expressed his frustration about accounts of animal learning in his time being anecdotal and keen to exaggerate intelligence. "*They have all been*

about animal intelligence, never about animal stupidity.", he complains, "*Dogs get lost hundreds of times and no one ever notices it or sends an account of it to a scientific magazine. But let one find his way from Brooklyn to Yonkers and the fact immediately becomes a circulating anecdote.*" (Thorndike 1898). Thorndike instead decided to systematically study instrumental conditioning by observing animals, including dogs, cats, and chickens, in puzzle boxes (Figure 1.1a). Escaping a puzzle box would require a sequence of actions, like pushing a lever, then pulling a string, then moving a bar. Thorndike noticed that repeated exposure to the same puzzle box would quickly reduce the escape time (Figure 1.1b). This observation led him to formulate his law of effect, which states that actions followed by 'satisfaction' become more likely to occur, while actions followed by 'discomfort' become less likely (Thorndike 1911). The law of effect thus expresses the principle of learning through trial-and-error. B.F. Skinner built on these ideas by systematically reinforcing spontaneous actions to show how animals can learn complex behaviours (Skinner 1937) – including a rat operating a vending machine (Iversen 1992) (Figure 1.1c), and a bowling pigeon (Skinner 1958). "*The result amazed us*", Skinner writes about the latter, "*In a few minutes, the ball was caroming off the walls of the box as if the pigeon had been a champion squash player.*"

As the next section casts the reinforcement learning problem in a mathematical format, the insights from these early conditioning experiments will feature in its algorithmic solutions. We will see how algorithms that solve reinforcement learning rely on tracking reward expectations and updating those through trial and error.

1.1.2 Theory: Markov decision processes

The framework of Markov Decision Processes (MDPs) formalises the essence of reinforcement learning, while being flexible enough to describe many different situations or experiments (Sutton and Barto 2018). An MDP considers an agent interacting with an environment. At each discrete time step t, the agent

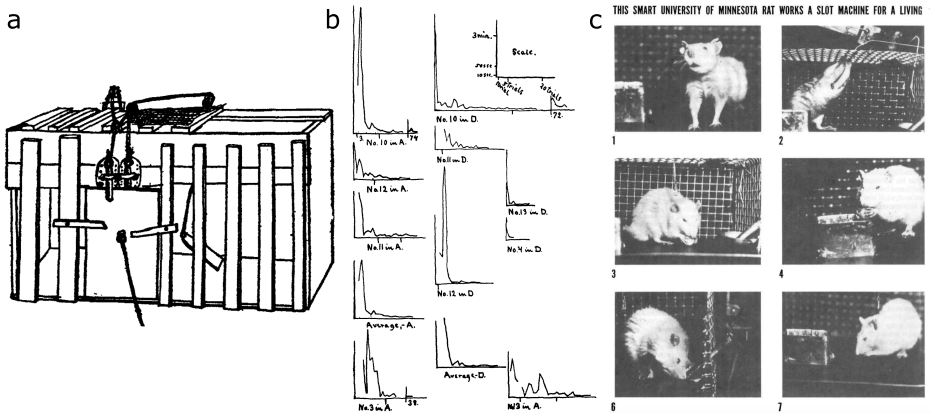


Figure 1.1: Instrumental conditioning. a. Example puzzle box for cats in Thorndike's experiments. b. Escape times for different animals in different boxes, plotted against trials in each box. With repeated experience animals quickly learn to escape. a, b adapted from Thorndike 1898 c. Photo series of a rat trained to operate a 'slot machine': first it needs to pull a chain, which releases a marble, and then it needs to deposit the marble in a slot for food to be delivered. Originally in *Life*, 1937; adapted from Iversen 1992.

receives the current *state* of the environment $s \in \mathcal{S}$, where $\mathcal{S}$ is the set of all possible states. The agent then selects an *action* $a \in \mathcal{A}(s)$ from the actions available to the agent in the current state $\mathcal{A}(s)$. As a result of that action, the environment produces a new state s', and reward r. More precisely, the agent selects an action from a *policy* probability distribution $\pi(a|s)$, and the new state s' and reward r are sampled from the *environment dynamics* probability distribution $p(s', r|a, s)$.

To make these abstract concepts more concrete, Figure 1.2 shows a toy example of an MDP that models the academic career. Starting as poor and unknown: $s = \text{P\&U}$, the academic can choose to either devote their time to writing a grant: $a = \text{G}$, or publishing an important paper: $a = \text{P}$. Being unknown, their grant will have no chance of success and they will be certain to end up in the same state, still without fame or money: $p(s' = \text{P\&U}, r = 0|s = \text{P\&U}, a = \text{G}) = 1$. Publishing on the other hand might take them to the poor and famous state, with the possibility of winning a grant next year. How can they find their optimal policy, allowing them to maximise their accumulated research funds throughout their career?

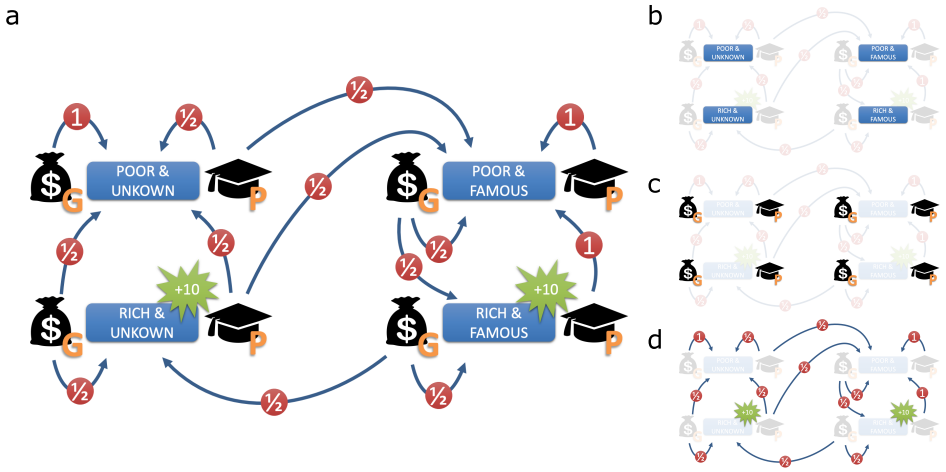


Figure 1.2: Academic Markov Decision Process. a. Example MDP that models the academic career. Publishing papers may lead to fame and writing grants might result in funding, but the success of either depends on the current state of the academic. b. The MDP has four states. c. The available actions are 'write grant (G)' and 'write paper (P)' in each state. d. The environment dynamics $p(s', r|a, s)$ determine the reward and next state after choosing an action.

In the case of an academic career that runs for a finite number of years T, the goal of learning is to maximise the expected *return* $g = \Sigma_{t=0}^{T} r_t$. But to accommodate for more general problems without a definite end, it is common to optimise the expected *discounted return* $g = \Sigma_{t=0}^{\infty} \gamma^t r_t$ instead, where $0 \leq \gamma \leq 1$ is the temporal discounting factor. A small γ upweights the importance of immediate rewards, while a large γ emphasizes long-term reward.

To know whether a particular state is attractive, the agent needs to know the expected discounted return given that initial state, called the state *value* $v(s)$:

$$v_\pi(s) = \mathrm{E}[\Sigma_{t=0}^{\infty} \gamma^t r_t | s_0 = s] \tag{1.1}$$

The expectation over the infinitely long future depends on what the agent will do, so it is policy-dependent, as indicated by the π subscript. Calculating this expectation seems complicated, but recognising a recursive relation in Equa-

tion (1.1) provides a useful expression:

$$\begin{aligned} v_\pi(s) &= \mathrm{E}[\Sigma_{t=0}^{\infty}\gamma^t r_t | s_0 = s] \\ &= \mathrm{E}[r_0 + \Sigma_{t=1}^{\infty}\gamma^t r_t | s_0 = s] \\ &= \Sigma_a \pi(a|s)\Sigma_{s',r} p(s', r|a, s)(r + \gamma \mathrm{E}[\Sigma_{t=0}^{\infty}\gamma^t r_t | s_0 = s']) \\ &= \Sigma_a \pi(a|s)\Sigma_{s',r} p(s', r|a, s)(r + \gamma v_\pi(s')) \end{aligned} \tag{1.2}$$

Equation (1.2) is called the *Bellman equation* for state values. After deciding on a policy, for example a random policy where $\pi(a|s) = \frac{1}{|\mathcal{A}(s)|}$, it allows for calculating the corresponding state values $V(s)$ for each state by iterating over states until convergence. That gives the academic an idea of their long-term budget if they would act randomly (Figure 1.3a, blue). However, they would probably be much more interested in the *optimal* policy: the policy with the highest possible expected discounted return for each state. The value $v_*(s)$ under the optimal policy π^* can be obtained from the *Bellman optimality equation*, in a similar iterative fashion called *value iteration*:

$$v_*(s) \leftarrow \max_{a \in \mathcal{A}(s)} \Sigma_{s',r} p(s', r|a, s)(r + \gamma v_\pi(s')) \tag{1.3}$$

After finding $v_*(s)$ for all states, getting the optimal policy only requires looking one step ahead:

$$\pi^*(a|s) = \begin{cases} 1 & \text{if } a = \underset{a \in \mathcal{A}(s)}{\operatorname{argmax}} \Sigma_{s',r} p(s', r|a, s)(r + \gamma v_*(s')) \\ 0 & \text{otherwise} \end{cases} \tag{1.4}$$

As expected, the values of states in the academic career MDP under the optimal policy (Figure 1.3a, purple) found through value iteration are higher than those for the random policy's values.

Although value iteration as described by Equation (1.3) and Equation (1.4) solves the problem of finding an optimal policy in an MDP, it has very little in common with the dogs, cats, and pigeons of Section 1.1.1. Instead of interacting with the environment, it simply iterates through a list of states. Even worse, it requires full knowledge of the environment dynamics $p(s', r|a, s)$, which

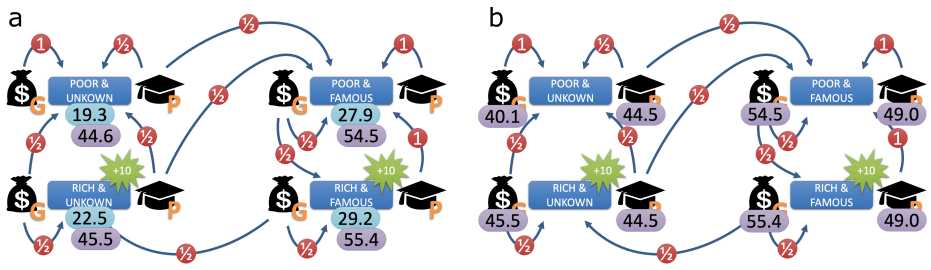


Figure 1.3: State and state-action values. a. State values $v(s)$ of the academic career MDP found through value iteration for the random policy (blue) and the optimal policy (purple) for $\gamma = 0.9$. b. State-action values $q(s, a)$ learned through Q-learning.

is rarely available to an animal. *Temporal difference* (TD) learning algorithms also find optimal policies, but do so by tracking and updating expectations through trial and error.

A central quantity in TD learning is the TD error:

$$\delta = r + \gamma v(s') - v(s) \tag{1.5}$$

After each step in the environment, having selected action a in state s and received a reward r and new state s' from the environment, the agent evaluates how wrong their expectation of discounted return was in state s. The new best guess of $v(s)$ is the immediate reward r plus the expected discounted reward ahead $v(s')$, giving the error of Equation (1.5). Then temporal difference learning updates its value estimate according to

$$v(s) \leftarrow v(s) + \alpha\delta \tag{1.6}$$

where α is the learning rate that determines how quickly value estimates are updated. Over many such updates, $v(s)$ converges to its true value. But to find out how to act, the agent would need to know what states they might end up in, and pick the action that leads to the highest-value state. Without that knowledge, they need *state-action values* $q(s, a)$ instead of state values $v(s)$; then the optimal action a in state s is simply the action with the highest expected discounted return $q(s, a)$. *Q-learning* is a TD algorithm that approx-

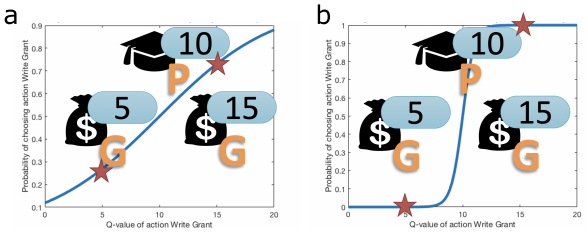


Figure 1.4: Softmax for explore versus exploit. a. Example softmax function for $\beta = 0.2$, when the q-value of 'write paper' is 10. If the q-value of 'write grant' equals 5 (left), the probability of that suboptimal action is 0.25. If the q-value of 'write grant' equals 15 (right), the probability of that optimal action is 0.75. b. Now $\beta = 2$, which gives probability 0 for 'write grant' if its q-value equals 5 (left) and 1 if its q-value equals 15.

imates state-action values under the optimal policy q_*. The TD update of Q-learning is as follows:

$$q(s,a) \leftarrow q(s,a) + \alpha(r + \gamma \max_{a'} q(s',a') - q(s,a)) \tag{1.7}$$

Figure 1.3b shows the state-action values resulting from Q-learning on the academic career MDP.

As the agent learns state-action values by interacting with the environment, they face a dilemma. On the one hand, having access to intermediate value estimates allows the agent to choose actions that are currently optimal for maximising reward. On the other hand, when these estimates are inaccurate, such a strategy will miss out on better options – and never find out, because Equation (1.7) only updates state-action values of action taken by the agent. The agent has to weigh between *exploitation* and *exploration*. The *softmax* function is commonly used to model this trade-off:

$$\pi(a|s) = \frac{e^{\beta q(s,a)}}{\sum_{a' \in \mathcal{A}(s)} e^{\beta q(s,a')}} \tag{1.8}$$

Here softmax temperature parameter β parametrises exploration versus exploitation (Figure 1.4): when β equals zero, every action is equally likely regardless of its value (explorative), but high βs strongly pick out the highest-valued action (exploitative).

The MDP and its solutions presented in this section are highly abstract. It is not always obvious how the real world maps onto the parameters in the equations above. But one reason for having such computational models of behaviour is that they give us quantitative measures to investigate neural activity. A formal description allows for precise experiments and hypotheses on how the brain solves reinforcement learning. The next section looks for evidence of neural activity related to the computations and model parameters presented here.

1.1.3 Neuroscience: reinforcement learning in the brain

One of the great successes of computational neuroscience is the *reward prediction error* (RPE) theory of dopamine. RPEs measure discrepancies between expected and encountered reward signals; the TD error δ of Equation (1.5) is a specific type of RPE for expected long-term rewards. Schultz and colleagues (Mirenowicz and Schultz 1996; Schultz et al. 1997) showed a spectacular correspondence between *phasic* ('bursty') responses of individual dopamine neurons in monkey ventral tegmental area and substantia nigra and the TD error in a conditioning experiment (Figure 1.5a). They found that these dopamine neurons respond with a burst of activity right after the delivery of a reward when the reward comes unannounced, or in early stages of conditioning. Indeed, defining states as discrete timesteps of the experiment, Equation (1.5) predicts a positive TD error on reward delivery, because both $v(s)$ and $v(s')$ are 0 while r equals a positive reward signal ρ. After successful conditioning, when the reward has been reliably preceded by a CS many times, this burst of activity on reward delivery vanishes, and instead appears on CS presentation. That also matches what TD-learning would predict. Conditioning assigns value $v(s) = \rho$ (assuming $\gamma = 1$: no discounting) to each of the states s between CS and reward, because the CS predicts reward. Now at reward delivery $v(s)$ equals $\rho + \gamma v(s')$ in Equation (1.5), so the TD error is 0. But on CS presentation, there is a sudden change in reward expectations: from $v(s) = 0$ to $v(s') = \rho$, producing a positive TD error. Finally, Schultz et al. removed the reward while still providing the

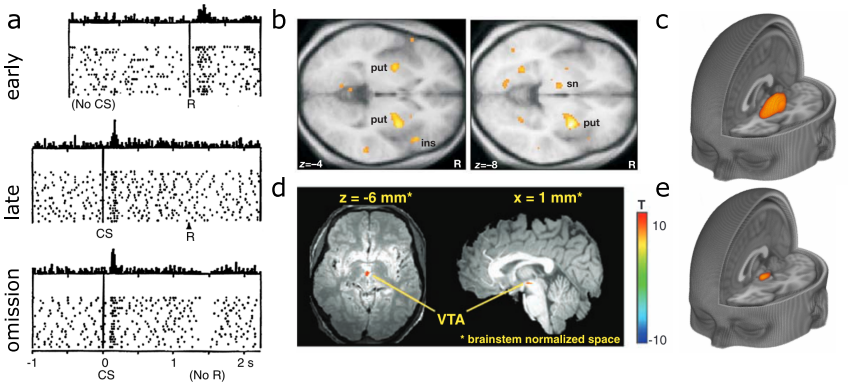


Figure 1.5: Reward prediction error δ a. The firing rate of single dopamine neurons in ventral tegmental area and substantia nigra track reward prediction errors. Before learning, or without a CS ('early'), there is a positive reward prediction error on reward delivery because reward occurred unexpectedly. After learning ('late') there is a positive reward prediction error on the CS, because from that moment on reward is expected. Finally, when the reward is ommitted unexpectedly ('ommission'), there is negative reward prediction error at the expected reward delivery time. Adapted from Schultz et al. 1997. b. Reward prediction error signal from fMRI in putamen, which is part of ventral striatum and receives input from the ventral tegmental area. Adapted from Seymour et al. 2004. c. Illustration of the putamen's location in the brain. Generated with *fsleyes* software, using the Harvard-Oxford Subcortical atlas. d. Reward prediction error signal from fMRI in ventral tegmental area. Adapted from D'Ardenne et al. 2008. e. Illustration of the location of ventral tegmental area from the Talaraich Deamon atlas in *fsleyes*.

CS in some trials. Now the positive TD error on the CS should remain, but when reward was expected to be delivered, there will be a *negative* TD error: $r + \gamma v(s') = 0$, while $v(s) = \rho$. Indeed, that is exactly the pattern of activity that Schultz et al. observed.

Human neuroimaging studies have similarly identified neural signals that correlate with reward prediction errors. Functional Magnetic Resonance Imaging (fMRI) measures the Blood-Oxygen Level Dependent (BOLD) signal as a proxy for neural activity. Using fMRI, Seymour et al. 2004 found reward prediction error signals in an aversive conditioning task in ventral striatum (Figure 1.5b), which receives input from dopamine neurons in the midbrain where Schultz et al. recorded. Imaging the midbrain itself is technically challenging, but D'Ardenne et al. 2008 managed using a custom protocol to detect reward prediction error signals in human ventral tegmental area (Figure 1.5d).

The discount factor γ in Equation (1.1) implements exponential discounting of delayed rewards. Exponential discounting enables the recursive relation of Equation (1.2), and is 'rational': equal time differences reduce the value by an equal factor. However, observed choice behaviour in delayed reward decisions overvalues immediate reward, consistent with (quasi-)hyperbolic discounting (Laibson 1997). Kable and Glimcher 2007 found BOLD signal in ventral striatum, medial prefrontal cortex, and posterior cingulate cortex as neural correlate of hyperbolic discounting (Figure 1.6a). McClure et al. 2004 instead proposed two distinct neural systems: short-run impatience driven by the limbic system, with activity in the same areas as Kable and Glimcher, versus long-run patience in lateral prefrontal cortex.

The learning rate α in Equation (1.6) determines how quickly prediction errors change value estimates. The appropriate value of α may depend on the environment. Behrens et al. 2007 presented a Bayesian model for decision making under uncertainty that optimally adapts α as required by the world's *volatility*. A highly volatile world, where rewards change frequently, requires rapid updating of value estimates: whether an option was good in the past holds little information about the present. In a stable world on the other hand an unexpected outcome is more likely to be an anomaly than an actual environmental change, so value estimate updates should be moderate. Like the model, humans were more responsive to new outcomes during volatile periods of a decision making task. fMRI showed that activity in anterior cingulate cortex tracks the estimate of the environment's volatility (Figure 1.6b).

Finally, inverse temperature β captures exploration versus exploitation in Equation (1.8). To find its neural correlates, Daw et al. 2006 had human participants repeatedly choose between four virtual slot machines. Spinning a slot machine produced a prize, randomly distributed around a mean – but that mean fluctuated over time, independently for each machine. To maximise the total reward, participants needed to choose the machine with the highest payout (exploit), while keeping an eye on whether other machines were taking over

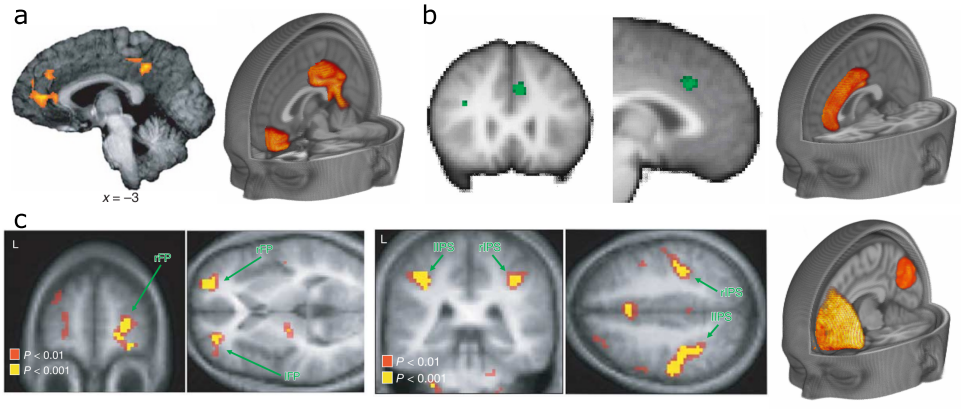


Figure 1.6: Discounting factor γ, learning rate α, softmax parameter β. a. Neural correlates of hyperbolic temporal discounting in medial prefrontal cortex and posterior cingulate cortex, with brain regions illustrated using the Harvard-Oxford Cortical atlas. Adapted from Kable and Glimcher 2007. b. Anterior cingulate cortex tracks the learning rate to adapt to the volatility of the task. Adapted from Behrens et al. 2007, illustrated using the Harvard-Oxford Cortical atlas c. Increased activity in frontopolar cortex and interparietal sulcus on explorative trials. Adapted from Daw et al. 2006, illustrated using the Harvard-Oxford Cortical atlas and the Juelich Histological Atlas.

(explore). Indeed, the softmax of Equation (1.8) captured choice behaviour. Neuroimaging showed that frontopolar cortex and interparietal sulcus significantly increased their activity on explorative trials, when participants chose a machine they believe to be non-optimal, over exploitative trials (Figure 1.6c).

1.2 Cognitive maps

The computational framework of reinforcement learning presented in Section 1.1 provided a formal problem, a solution to that problem, and some remarkably precise correspondences with neural data. However, there is whole class of behaviour that it fails to describe. Animals, and humans in particular, behave flexibly. They can learn in the absence of reinforcement signals, and make appropriate decisions in situations they have not experienced before. Psychologists proposed a *cognitive map* to account for this flexible behaviour. The cognitive map is embedded naturally in RL theory through the distinction between *model-free*, or *habitual*, and *model-based*, or *goal-directed*, control. The cognitive map and its application in decision making again led to a wealth of

neural results.

1.2.1 Psychology: latent learning

In his influential 1948 paper, American psychologist Edward Tolman reported experiments "*carried out by graduate students (or underpaid research assistants) who, supposedly, got some of their ideas from me.*" (Tolman 1948). One of them is an experiment on *latent learning* by Blodgett in 1929 (Blodgett 1929). In stark contrast to the types of learning that Skinner's experiments from Section 1.1.1 demonstrated, Blodgett observed learning *without* any reward in rats solving mazes (Figure 1.7a). One group of rats was introduced in the maze with food hidden on the other side, and this group reached the food with fewer mistakes (measured as entries into dead end maze arms) on each trial. Another group of rats first experienced many trials of the maze without any food, during which their mistakes remained at a constant high level. However, as soon as food was introduced, the number of mistakes dropped sharply – faster than the initial trials of the first group (Figure 1.7b). As a contemporary review (Wohlgemuth 1931) concludes, this indicates that "*during the non-reward period, the rats were developing a latent learning of the maze, which they were able to utilize as soon as reward was introduced.*" (The same review continues to remark that "*The author seems, however, to overlook the fact that there was no special drive for the rat before the introduction of the reward.*".)

Based on this experiment and others, Tolman proposed the existence of a *cognitive map*, a rich mental representation of environmental relationships. Learning a cognitive map requires learning about relations between *stimuli* rather than between stimulus and response. Once constructed, it allows for dealing with environmental changes in ways that stimulus-response associations cannot. For example, Tolman et al. 1946 showed that rats take shortcuts without prior experience when they become available. Initially, the only way to the goal is a single maze arm that takes a detour (Figure 1.7c). After a few trials, the maze changes: the detour arm is blocked, but an array of straight arms

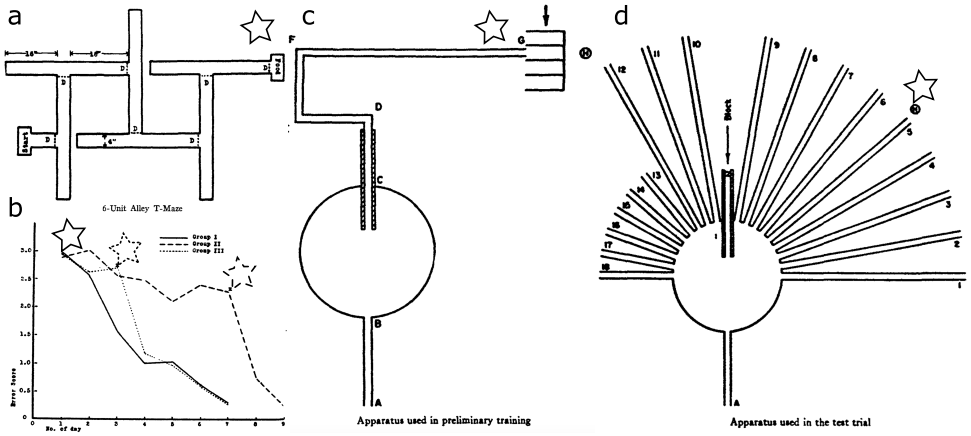


Figure 1.7: Using a cognitive map. a. Maze used in Blodgett 1929 to demonstrate latent learning. For one group of animals reward is always provided at the star, for other groups reward only appears after some days. b. Number of errors, measured as dead-end entries, across days for the different groups, where the star indicates the day that reward is first introduced. The number of errors after reward introduction drops steepest for the group for which reward appears last, suggesting they have learned about the maze before reward introduction. c. Maze used to demonstrate shortcut behaviour in Tolman et al. 1946. On the first day, animals are trained to find reward at the star via a detour. d. Afterwards, the original maze arm is blocked, but new arms open up in many directions. Animals prefer the arm that leads them directly to the reward even through they have not experienced taking it before. a-d adapted from Tolman 1948.

in many directions is opened up (Figure 1.7d). In this new configurations rats strongly preferred the arm oriented directly along the goal direction. Tolman hypothesised that cognitive maps exist beyond space, to organise knowledge in all domains of behaviour.

This idea of systematic organisation of knowledge for flexible behaviour was studied by Harlow around the same time in monkeys (Figure 1.8a). Harlow (Harlow 1949) noticed that monkeys playing many iterations of a task where one out of two objects is rewarded, improved at choosing the rewarding object after objects change (Figure 1.8b). The monkeys had *learned to learn*: they discovered the rule that if one object is rewarding, the other one is not, and quickly applied it to new objects. Harlow described this behaviour as acquisition of a *learning set* for the object-quality discrimination problem. Like cognitive maps, learning sets describe relations between experiences, but more than cognitive maps, they generalise: they abstract away specifics to only retain

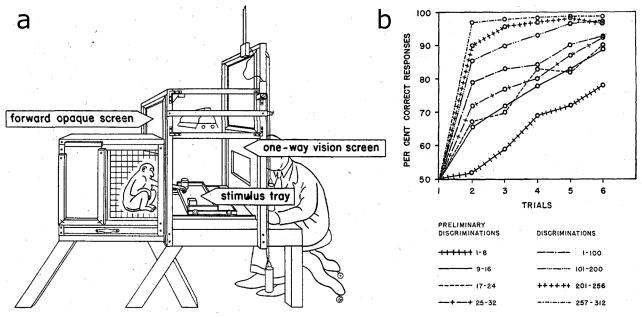


Figure 1.8: Learning sets. a. Apparatus used by Harlow 1949 to present two objects to a monkey, one if which is rewarding while the other one is not. b. Over many such tasks, with different objects but the same rule, the animal makes fewer mistakes each time. It has 'learned to learn'. a-b adapted from Harlow 1949.

what the experiences have in common.

The behaviours observed in the experiments of the current section do not align well with the temporal difference RL algorithms of Section 1.1. However, with minor adaptations the same framework accommodates these types of flexible behaviour (Baram 2020). The next section extends the solutions of Section 1.1.2, so we can benefit from the same mathematical precision on a broader scope.

1.2.2 Theory: model based decisions

In terms of the MDP of Section 1.1.2, the cognitive map acts as a *model of the environment*. Specifically, it represents the environment's *transition structure*:

$$p(s'|a,s) = \Sigma_r p(s',r|a,s) \tag{1.9}$$

TD-learning methods like Q-learning do not require this environment transition model, and are therefore called *model-free*. But there is a different class of algorithms that supports the types of behaviour in Section 1.2.1, like learning without reward and acting without experience. These algorithms do rely on knowledge of state transitions and are called *model-based* (Dayan and Niv 2008; Dolan and Dayan 2013).

In model-based RL, the agent uses the transition model to simulate multi-step futures and evaluate their values. This process is often referred to as

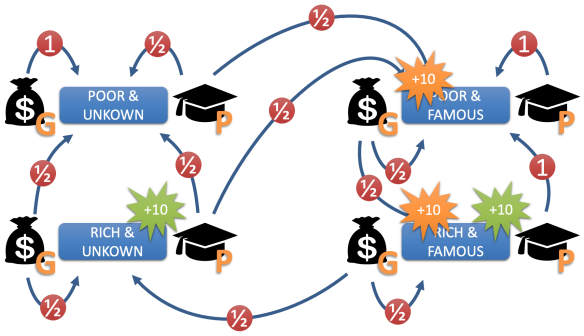


Figure 1.9: Model-based learning. When the rewards change in the academic career MDP, here from acquiring money (green stars) to acquiring fame (orange stars), a model-based academic will realise they should choose 'write paper' in the 'rich and famous' state, while the q-values learned previously would recommend 'write grant'.

planning. Once the academic has an understanding of the dynamics of the career MDP, they can imagine how first writing a paper and then writing a grant might take them from poor and unknown to rich and famous. To decide on a good course of action, they consider a decision tree, and select the path that leads to the best outcome. This method, called *tree search*, is computationally expensive. Decision making in model-free RL is much simpler: it only requires comparing each action's stored expect future rewards, built up over many experiences.

However, the advantage of planning over possible futures becomes obvious when the environment changes. For example, in the utopia of securing a source of long-term funding, the academic might start valuing being famous over being rich: a change in reward signal of the MDP (Figure 1.9). The stored Q-values do not immediately reflect this change, and a model-free academic may decide to write a grant instead of publishing while rich and famous, which is no longer in their interest. For appropriate Q-values, a model-free agent needs to access a state, act, and experience the consequences of their actions (Equation (1.7)) – possibly many times. The model-based agent on the other hand does not need this kind of experience. When their model changes, their policy updates immediately through planning.

In summary, model-free RL is quick and robust at decision time, but needs many experiences to arrive at accurate state-action values, which is particularly problematic for environment changes. Model-based RL is slow and error-prone at decision time, because planning is hard, but flexibly adjusts to new situations with little experience. This trade-off suggests that the preferred method of choice is situational. Daw et al. 2005 proposed uncertainty-based arbitration, assuming that the agent acts on whichever method's value estimate is deemed more reliable. Early on in learning, model-based RL quickly builds up a map from short term transitions, outperforming model-free RL in data efficiency. Later on, the long-term value estimates from model-free RL become more trustworthy than the model-based planning process. Therefore this account expects a shift from model-based to model-free RL over continued experience.

Armed with these extended methods for modelling behaviour in RL experiments, the next section again explores neural evidence for model components. It focuses in particular on how the brain represents the transition structure – the states and their relations – of the cognitive map, and on the use thereof in model-based RL.

1.2.3 Neuroscience: maps and states

Another famous set of single-neuron recordings provided a beautiful demonstration of a spatial cognitive map in the brain of rats. O'Keefe 1976 found single neurons in rat hippocampus that are active whenever the animal is in a specific location in the environment (Figure 1.10a). A population of such *place cells* cover the entire environment, so that O'Keefe and Nadel 1978 considered the hippocampus to be the core of the memory network supporting the cognitive map. The discovery of place cells was followed by reports of other cells in the hippocampal-entorhinal cortex system that fire with spatial specificity, and may thus contribute to the representation of cognitive maps. The most well-known of these are *grid cells* (Hafting et al. 2005), active at regular periodic

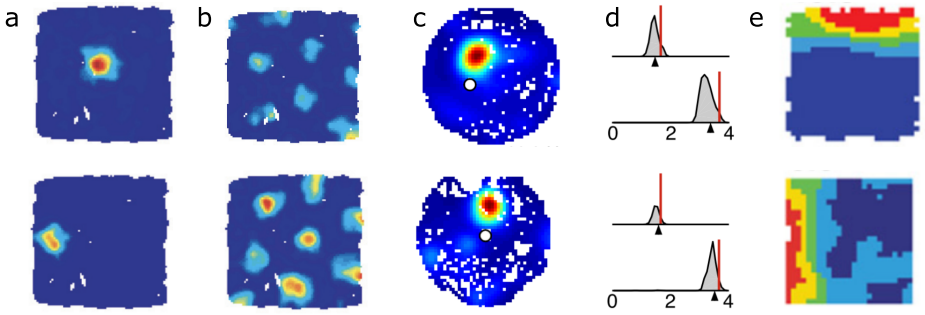


Figure 1.10: Spatially tuned neurons. a. Firing rate plotted as heat map across the arena (blue is low rate, red is high) to demonstrate spatial tuning. Two examples of place cells that fire a a single location in the environment. b. Two examples of grid cells with firing fields arranged on a triangular lattice. a-b adapted from Hafting et al. 2005. c. Example of a single object vector cell that is active at fixed distance and angle from an object (white circle), moving along when the object moves. Adapted from Høydal et al. 2019. d. Two examples of reward cells that fire near reward on a linear track, and move their firing field when the reward moves. Adapted from Gauthier and Tank 2018. e. Two examples of boundary vector cells with firing fields parallel to arena boundaries. Adapted from Lever et al. 2009.

locations along a triangular grid across the environment (Figure 1.10b). A population of grid cells, with the same type of lattice at different shifts, rotations, and scales, can support vector navigation (Bush et al. 2015). Other examples of reported spatially selective cell types are object vector cells (Høydal et al. 2019) (Figure 1.10c), reward cells (Gauthier and Tank 2018) (Figure 1.10d), and boundary vector cells (Lever et al. 2009) (Figure 1.10e).

Doeller et al. 2010 invented a clever method for measuring signatures of the same grid code in humans in fMRI. The firing rate of a grid cell will be different during motion aligned with the grid lattice directions compared to misaligned motion, because firing fields are crossed at different rates. This yields a sixfold symmetry in the BOLD signal as a function of movement direction, which Doeller et al. observed when participants navigated a virtual environment. And such signals also exist for abstract spaces: Constantinescu et al. 2016 showed the same signatures of grid coding in a conceptual space of stretchy birds. Participants in an fMRI study learned about bird stimuli as locations in a space of birds with neck length on one and leg length on the other axis (Figure 1.11a). Moving from one bird stimulus to another then has a corresponding direction

in this abstract bird space. Constantinescu et al. found variations in BOLD signal for directions along a hexagonal symmetry, which could indicate a grid-like code for bird space (Figure 1.11b).

Even more closely related to the RL formulation of the cognitive map, with its discrete states and transitions, is the hypothesis that orbitofrontal cortex represents the *task state* (Wikenheiser and Schoenbaum 2016). So far the RL state s was simply a given, but having access to whatever information is relevant for the current decision is far from trivial. It does not even have to be directly observable: sometimes the same stimulus affords different futures, depending on its history. Wilson et al. 2014 suggested that orbitofrontal cortex tracks states and is particularly critical when tasks involve states that include unobservable information. Schuck et al. 2016 designed such a task, in which partipants need to report the age of either a house or a face stimulus overlaid on top of each other (Figure 1.11c). The currently relevant category of the age judgement depends on the relevant category and age of the previous stimulus. This design induces a state space of 16 states that contain all information required for the correct response. Schuck et al. showed that the identity of each task state can be decoded from fMRI patterns in medial orbitofrontal cortex (Figure 1.11d).

Model-based RL relies on these cognitive map representations. In order to find neural substrates of model-based RL, it is necessary to precisely distinguish model-based and model-free RL. Daw et al. 2011 invented the *two-step task* that can tease apart model-free and model-based decision making by predicting different behaviour for each (Figure 1.12a). The two-step task consists of two stages. In the first stage, participants choose between two actions a and b. Depending on their choice they transition to one of the two states A or B of the second stage, via a common (70 % in the original experiment) transition $a \rightarrow A$ and $b \rightarrow B$, on an uncommon transition (30 %) $a \rightarrow B$ and $b \rightarrow A$. Then at the second stage there is another binary decision, for which the reward fluctuates over time, so participants need to keep updating their value estimates. The crucial difference between model-based and model-free learning in this task

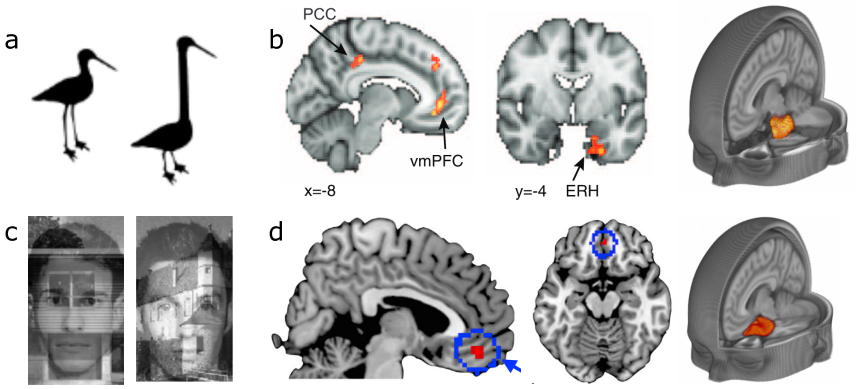


Figure 1.11: Non-spatial maps. a. 'Stretchy bird' stimuli. By varying the length of neck and legs of bird stimuli, Constantinescu et al. 2016 create a continuous two-dimensional 'bird-space'. b. A hexagonal symmetry in BOLD signal for transitions through bird space suggests a grid-like coding in posterior cingulate cortex, ventromedial prefrontal cortex, and entorhinal cortex. The latter is illustrated using the Juelich Histological atlas. c. Stimuli of faces and houses overlaid on top of each other for age judgement in Schuck et al. 2016. A task where the relevant category depends on the previously relevant category induces unobservable task states. d. Neural representations of task state are found in medial orbitofrontal cortex, illustrated using the Oxford-Harvard Cortical atlas.

is what happens on a *surprisingly high* reward after an *uncommon* transition, for example $b \rightarrow A$. For a model-free agent this reward would reinforce their initial choice b; a model-based agent that has learned the transition structure would realise initial choice a is more likely to lead to the same reward. In other words, such a reward *increases* the probability of repeating the same first-stage action for model-free RL but *decreases* it for model-based RL, and the other way around for a surprisingly low reward (Figure 1.12b). Daw et al. found that humans appear to do a mixture of both, and showed that the regions with model-free prediction error activity from Section 1.1.3 also carry model-based error information. Miller et al. 2017 adapted the two-step task for rats and found that inactivation of dorsal hippocampus and orbitofrontal cortex reduced their ability to use model-based strategies. Similarly, Vikbladh et al. 2019 showed that patients with hippocampal lesions are skewed towards model-free behaviour compared to controls.

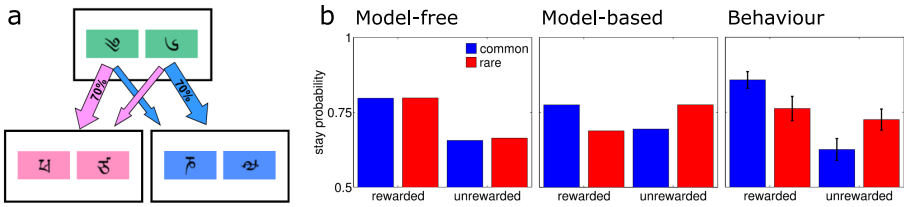


Figure 1.12: Two-step task. a. Design of two-step task in Daw et al. 2011. The left green stimulus commonly transitions to the pink second stage, and rarely to the blue second stage; vice versa for the right green stimulus. b. Receiving a reward in the second stage reinforces the first-stage decision of a model-free agent, independent on the transition. A model-based agent will repeat the first stage decision after a second-stage reward through a common transition, but will switch choices if that reward followed a rare transition. Measured human behaviour appears to be a mixture of both. Adapted from Daw et al. 2011.

1.3 Relations and composition

In addition to the flexible behaviours of Section 1.2, cognitive-map-like representations have another great advantage: they support *generalisation*, the application of prior knowledge in different but related situations. Being able to generalise means rarely having to learn from scratch, but instead relying on commonalities with previous experience. Clearly generalisation requires memory of these previous experiences, but organising memories in a relational way proves to be particularly useful. Compositionality takes full advantage of relational representations for the construction of new wholes from known parts.

1.3.1 Relational memory

The role of hippocampus in representing spatial maps and supporting their navigation in Section 1.2.3 is only one side of an extensive literature on hippocampal function. Another line of research originated from severe memory impairments observed in patients with hippocampal lesions. Brenda Milner (Scoville and Milner 1957) † examined the consequences of bilateral medial temporal lobe resection on patient H.M. – a "*frankly experimental operation considered justifiable because the patient was totally incapacitated by his seizures and these had proven refractory to a medical approach*". The result was a "*striking and*

†poem in Appendix

totally unexpected behavioural result: a grave loss of recent memory". H.M.'s personality, mood, understanding and reasoning had remained unchanged, his early childhood memories vivid and intact. But he was unable to form any new memories. Cohen and Squire 1980 realised that amnesic patients like H.M. could retain something: they were able to improve procedural skills, such as reading mirror-image words. Hippocampus was thus assumed essential for *declarative* memory, memory that can be recalled and declared. That sounds like a very different function from the place cell map – but is it?

Howard Eichenbaum, together with Cohen, suggested that hippocampus supports declarative memory through a general relational processing capacity (Eichenbaum and Cohen 2014). Eichenbaum proposed hippocampus is critical for encoding relations between items in memory, to systematically organise memories in a network of relationships (Eichenbaum 1999). Hippocampus rapidly forms associations between new input and reactivated relational memories, to bind common elements of experience and memories. Memories organised like this allow for flexible memory expression: relationships that have never been observed can be inferred by connecting other items that are only indirectly related. *Transitive inference* is a good example of this flexible memory expression. If a car is faster than a bike, and an airplane is faster than a car, then an airplane must be faster than a bike – it is not necessary to observe that comparison. Indeed, Dusek and Eichenbaum 1997 showed that hippocampus is required for the acquisition and application of an ordered series of odour relationships. Rats trained on pairwise preferences of B over A, C over B, and D over C prefer D over B without further training, but after a hippocampus disconnection they lose the capacity to compare indirectly related options. Eichenbaum concluded that the cognitive map's place cells are just a particular example of relational memory and flexible memory expression (Eichenbaum et al. 1999). Figure 1.13 illustrates how 'spatial memory' is a special case of 'memory space'. Spatial memories of places along routes, that can be integrated in maps, are examples

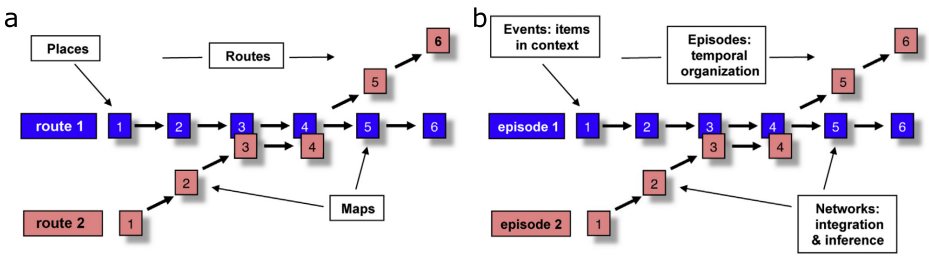


Figure 1.13: Spatial memory and memory space. a. Places are spatial memories, encountered along different routes. Linking the routes together, for example at locations where they overlap, creates an integrated map that relates places traversed at different times. b. Eichenbaum and Cohen 2014 proposed that this mechanism of constructing maps of 'spatial memory' is a specific example of a general mechanism of organising 'memory space'. Events are experienced over different episodes, and organised into networks to support integration and inference. Adapted from Eichenbaum and Cohen 2014.

of a general mechanism for organising events along episodes, integrated in networks in memory space.

So what is the mechanism by which hippocampus builds relational memories? Manns and Eichenbaum 2006 sketched an outline of a model for declarative memory encoding based from anatomical considerations. They recognised two largely parallel pathways that converge in hippocampus: non-spatial information about individual items and objects through perirhinal and lateral entorhinal cortex, and spatial or contextual information via parahippocampal and medial entorhinal cortex. Hippocampus can therefore create item-context associations, for example through a *conjunction* of the two.

Whittington et al. 2020 formalised this idea in an artificial neural network model: the Tolman-Eichenbaum machine (TEM). TEM implements structural knowledge generalisation and relational memory through the *factorisation* of objects and space and their *conjunction* in memory, as Manns and Eichenbaum proposed. Factorised representations separate the different aspects of experience, here object and space, whereas conjunctive representations bind the separate parts together. TEM applies these principles as follows. The world is modelled as a network of states, for example a regular square grid, connected by actions, for example go-north, go-east, go-south, and go-west. Each state

has an associated observation, which might be different for each different world that TEM encounters. TEM experiences the world step by step, with an action and the resulting observation at each step. Its job is to predict the next observation (Figure 1.14a). In order to make such predictions, TEM has to learn about the regularities of the action transitions, so it can keep track of where it is. In the grid example, a go-north then go-east then go-south then go-west action returns it to its starting point. But TEM also has to remember what the observation was at that starting point, if it is to predict it (Figure 1.14b). TEM uses the learned transition rules to represent its position in medial entorhinal cortex, and binds this location signal together with the observation from lateral entorhinal cortex, in a combined representation in hippocampus (Figure 1.14c). The medial entorhinal transition rules are learned slowly, because they extract the commonalities of experience across worlds, through backpropagation. The hippocampal conjunctions are learned quickly, through attractor dynamics. The artificial neurons in TEM trained on many two-dimensional spatial worlds closely match the hippocampal place cells and entorhinal grid cells observed in rodents (Figure 1.14d).

The principle of relational memory through factorisation and conjunction as applied in TEM is so powerful because it allows for *generalisation* (Behrens et al. 2018). Across many different grid worlds, the same rules apply for how the actions relate to each other. Abstracting, or factorising, this structural knowledge from sensory experience is like building a relational template of all possible grid worlds. With access to that template, building a cognitive map of one specific grid world is easy: it only requires slotting in the observations in the right template locations. The slotting in of concrete experience in the abstract relational template is exactly what conjunction achieves. The relational template captures the abstract rules, the conjunctive map makes these rules applicable to the current situation (Figure 1.15). It allows for predicting the observation after the north-east-south-west loop, without having experienced that final go-west action in the current world before. It enables the transitive

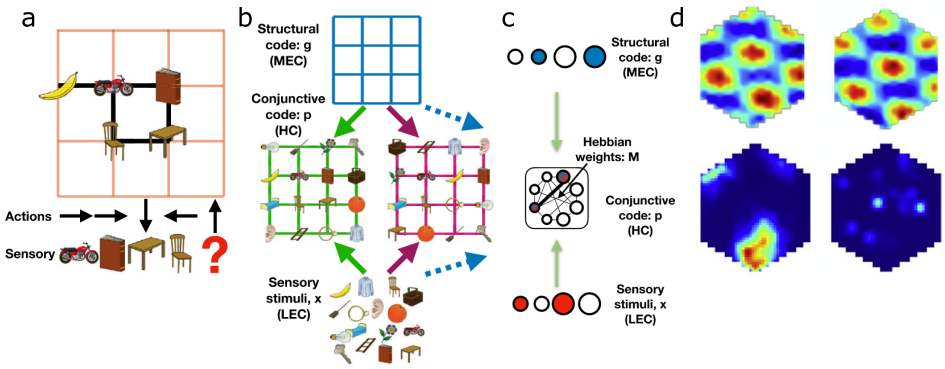


Figure 1.14: The Tolman-Eichenbaum Machine. a. TEM experiences the world by taking actions (arrows) and observing the resulting sensory stimuli (icons). Its job is to predict the next stimulus (red question mark). In the regular square grid sketched here, TEM can predict the next stimulus will be a motorbike if it has learned that east-south-west-north actions take it to a previously visited place, and if it remembers a motorbike at that location. b. TEM factorises sensory stimuli from abstract relational structure, and combines the two into a conjunctive map of the current environment. c. The abstract structure is represented in medial entorhinal cortex, the sensory stimuli in lateral entorhinal cortex, and the conjunctive map in hippocampus. d. The units of the artificial neural network show a grid pattern in entorhinal cortex and place fiels in hippocampus. a-d adapted from Whittington et al. 2020.

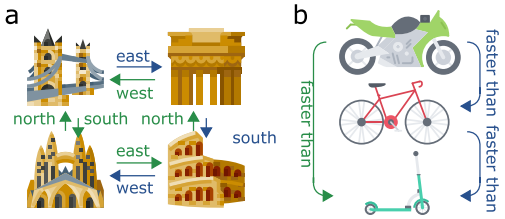


Figure 1.15: Generalisation. a. Example of how generalisation of relational structure affords inferences in new situations. After learning about the regularities of two-dimensional space, all the relations with green arrows can be inferred from experience of only the relations with blue arrows. b. Similarly, regularities of one-dimensional ordering allow for transitive inference.

inference that the airplane is faster than the bike, because the car is faster than the bike and the airplane is faster than the car. Harlow's learning sets are examples of relational templates: if between two options one is always rewarding while the other one is not, finding out that the cube on the table is 'good' means the cylinder next to it is 'bad'.

1.3.2 Compositionality

Building a cognitive map by tying together items in relations is a form of *compositionality*: the construction of new representations through the combination of primitive elements. Lake et al. 2017 listed compositionality as one of the core ingredients of human intelligence. Composition supports *combinatorial generalisation* (Battaglia et al. 2018) by reusing a small set of known building blocks to obtain novel constructs in an almost limitless way.

Composition has been well-studied in the context of linguistics, where infinitely many sentences can be constructed from a set of words following the rules of grammar (Chomsky 1965), but that could be a thesis by itself. Lake et al. 2015 showed how even the characters that make up words are compositional (Figure 1.16a). Their *Bayesian Program Learning* framework represents concepts as probabilistic programs, or procedural generative models, consisting of parts and their spatial relationships. A program that generates the character *t* for example would consist of a downwards stroke that curls up to the right, and a horizontal bar that crosses it just above the middle, with some added random variance. After training on a set of background characters, taken from for example the Greek and Japanese alphabet, the framework achieved human-level performance in tasks that examine generalisation from few examples with different alphabets, for example Hebrew and Korean. The model can classify unseen characters from a single example, and human judges identified model-generated characters between human writing in a 'visual Turing test' barely above chance. Moreover, like people, the model can generate completely new concepts: given a few characters from Korean, the model can invent new Korean characters – and judges performed at chance when distinguishing those from human invented characters (Figure 1.16b).

A similar type of imagining novel stimuli was studied by Barron et al. 2013 in a completely different domain: taste. They presented pictures of food familiar to their participants, like tea and jelly, but also novel dishes that combine these foods, like tea-jelly, that participants have never tasted before (Figure 1.16c).

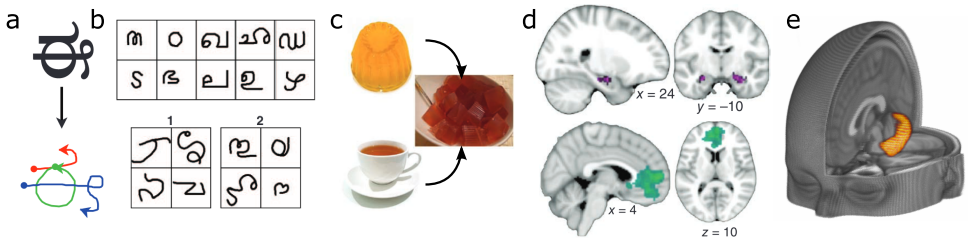


Figure 1.16: Composition of concepts. a. Handwritten characters can be composed from strokes and their spatial relations. b. Bayesian program learning (Lake et al. 2015) learns probabilistic models for generating characters. From a few character examples (top row), it can invent new characters in the same alphabet that pass a 'visual Turing test': judges are unable to distinguish model versus human invented characters (bottom row). The solution for this example can be found in the caption footnote†. a-b adapted from Lake et al. 2015. c. The sensation of new dishes, like tea-jelly, can be imagined from their ingredients, like tea and jelly. d. Representations between compounds and their constituents overlap in hippocampus and medial prefrontal cortex (Barron et al. 2013). c-d adapted from Barron et al. 2013. e. Illustration of hippocampus using the Harvard-Oxford Subcortical atlas.
† 1: human, 2: machine

Using fMRI they found that the neural codes in hippocampus and medial prefrontal cortex code overlapped for the compound dishes and their constituents (Figure 1.16d,e), and that the overlap decreased with repeated presentation of the compound stimulus. Then a second group of participants actually got to taste each compound dish before the scan. The overlap of compound and constituent representations decreased from the first to the second group. These results suggest that representations of novel goods are constructed from memories of its constituents, and that repeated presentation or sensory experience of the compound decreases the need for online construction.

These studies examine compositional representations of sensory items. Items themselves can in turn be combined into compositional experiences during *scene construction* (Maguire et al. 2016), defined by Hassabis and Maguire 2007 as 'the process of mentally generating and maintaining a complex and coherent scene or event'. Given H.M.'s inability to recollect the past, Hassabis et al. 2007 wondered how patients with hippocampal damage would compare to control subjects when constructing new imagined experiences. They provided participants with short verbal cues, such as 'Imagine that you are standing in

the main hall of a museum containing many exhibits', and encouraged them to describe something new. Compared to controls, patients provided fewer details, and their imagined experiences are fragmentary and lacking in coherence. But the actual descriptions are even more striking. "*To be honest there's not a lot coming. (...) My imagination isn't... well, I'm not imagining it, let's put it that way. Normally you can picture it can't you? I'm not picturing anything at the moment.*". McCormick et al. 2017 further specified the impairments of scene processing in patients with hippocampal damage. They asked patients and controls to judge whether scenes are semantically impossible, like someone hoovering a tree (Figure 1.17a), or constructively impossible, such as M.C. Escher-like arches (Figure 1.17b). Patients were selectively impaired for the latter judgement.

On a more abstract level, structures of knowledge can be composed from concepts (Tenenbaum et al. 2011). For example, the organisation of biological species fits well in a tree structure, and social networks are often organised into cliques. Such composition requires the discovery of structural form – a tree? a ring? clusters? – and then constructing the specific instance of the structure that best explains the data. Kemp and Tenenbaum 2008 modeled this process as maximising the posterior probability $\mathrm{p}(S, \mathrm{F}|\mathrm{D}) \propto \mathrm{p}(\mathrm{D}|S)\mathrm{p}(S|\mathrm{F})\mathrm{p}(\mathrm{F})$, where F is a *form*, S is a *structure*, and D is the data, described by a set of features. In other words, they build a generative model of how forms produce structures and how structures produce data, and then turn it around to ask which form and which structure were most likely to produce particular observed data. Using *graph grammars* they grow structures step by step, following rules specific to each form, until they find the structure that minimises the difference between neighbouring concepts (Figure 1.17c). Provided with a matrix of animal species and their biological and ecological properties the best scoring form is a tree (Figure 1.17d), and the best tree separates species into meaningful subtrees (species into mammals into rodents, for example). Provided with a matrix of US

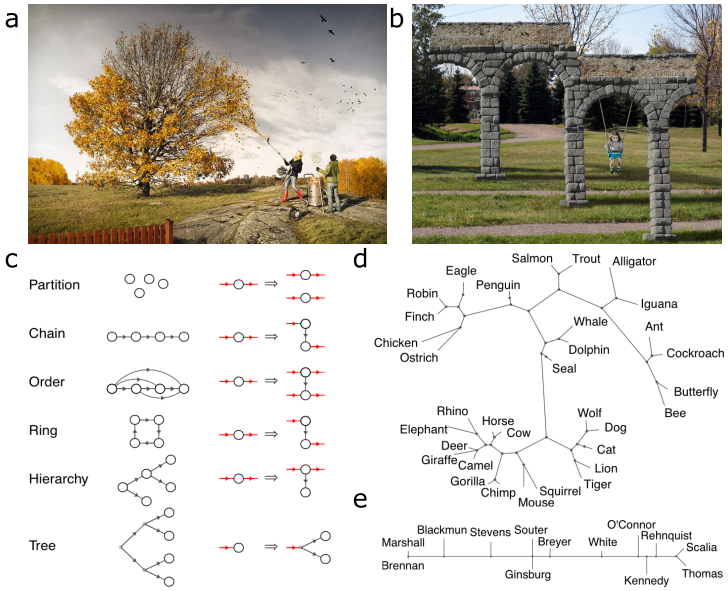


Figure 1.17: Composition from concepts. a. Example of a semantic violation stimulus used in McCormick et al. 2017. b. Example of a constructive violation stimulus used in McCormick et al. 2017. Patients with hippocampal damage are selectively impaired to detect the latter. a-b adapted from McCormick et al. 2017. c. Generative process for growing structures of different forms, used to discover the most likely organisation of concepts in Kemp and Tenenbaum 2008. d. Optimal structure of animals given data on their ecological and biological properties: a tree. e. Optimal structure of US supreme court judges given data on their voting behaviour: an ordering. Adapted from Kemp and Tenenbaum 2008

supreme court judges and their votes, the optimal structure is a line that ranks judges from liberal to conservative (Figure 1.17e).

These studies of compositionality, theoretical and empirical, on different levels of granularity and abstraction, illustrate the central role of composition in flexible behaviour and understanding of the world. It allows for learning from single examples, recombining existing knowledge to reason over experiences without experience, and for organising knowledge through relational structure (Summerfield et al. 2020). Composition extends the notion of generalisation from the previous section. Factorising relational templates from sensory experience enables composition of new maps by tying concrete experience to the abstract template. Such factorisation-and-conjunction thus generalises across situations where the same templates apply to different items. But even greater generalisation can be achieved by composition of the relational templates *them-*

selves. Through experience of many different situations with different relational templates, the relational building blocks that make up these templates can be abstracted and flexibly recombined into new templates. Structural elements like walls and doors make up rooms, that combine into abstract templates for buildings. A marriage links up two family tree structures into a new template that can then be applied to specific family members.

1.4 Thesis outline: states and spaces

In summary, Section 1.1 presented the framework of reinforcement learning to formalise the problem of learning to act to maximise reward, and temporal difference algorithms to solve that problem by learning from experience. Section 1.2 extended this framework by introducing cognitive maps and their application in model-based algorithms that allow for flexible behaviours where model-free temporal difference learning would not. Finally, Section 1.3 shows how relational memory and compositionality enable generalisation through the construction of new concepts, experiences, or maps from previous experience. Taken together, these sections provided motivation for and evidence of the use of relational and compositional representations for understanding and navigating the world. The remainder of this thesis presents experimental and theoretical work on how this understanding and navigation takes place.

Chapter 2 investigates composition *within* states, and how the statistics of composition induce meaning in objects and situations. Perceptually, a bar stool is similar to a table: both have a flat top and pointy legs. But conceptually, a bar stool is more like a sofa, because both are for sitting *on* while a table is for sitting *at*. In other words, objects that compose the same way across situations acquire similar meaning. A parallel argument is made for situations: situations composed of similar objects are conceptually similar. And, regardless of objects, the way stories are constructed from situations adds another dimension of similarity. I designed, carried out, and analysed an fMRI experiment that probes neural representations of objects and situations from perceptual

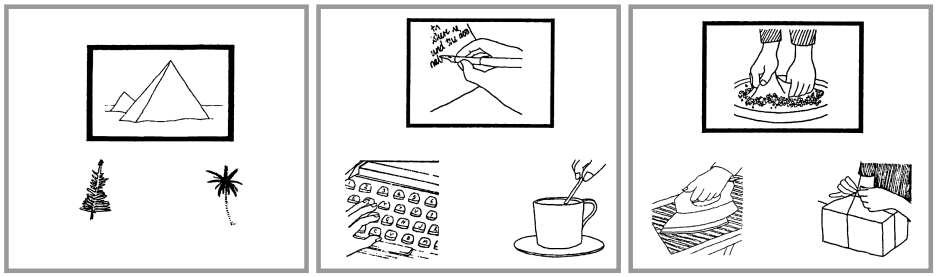
to conceptual to examine how semantic understanding might arise from the statistics of composition.

Chapter 3 considers composition *between* states: how state spaces are constructed from states. I show how an optimal policy can be constructed without *any* additional learning of reward expectations in a new environment. The trick is to use structural building blocks, like barriers and goals, that immediately imply the right course of action when combined. For example, when there is a wall on one side and a reward on the other, the optimal policy directly approaches reward. But if the wall is positioned in front of the reward, the wall imposes a detour. By building relational templates from structural building blocks, and tying the template to the concrete observations of a new world, it is possible to directly map representations to actions. Latent learning allows for constructing a template before reward is encountered, and replay – the process of sequentially reinstating remote experience in memory – plays a central role in building a map of the current world from this constructed template.

2
From shape to understanding

Patients with semantic dementia have difficulties remembering words. For example, neuropsychologist Elizabeth Warrington described a patient who fails to explain hay *("I've forgotten"),* poster, *("No idea"),* acorn *("Don't know"), or* too many cooks spoil the broth *("If I knew what broth was I expect I could explain it"). But this impairment is not just about language. The 'Pyramids and Palm trees' test below shows an image, for example a pyramid, and asks to match it to one of two other images, for example a palm tree (correct) and a pine tree (incorrect). And a variation on the same test, the 'Kissing and Dancing' test below, shows images of verbs or situations – writing goes with typing not stirring; washing goes with ironing not knotting – instead of objects. Patients with semantic dementia also struggle on these non-verbal tasks. It turns out conceptual knowledge is what deteriorates in semantic dementia patients: words, objects, and concepts lose their meaning.*

In this Chapter, I will investigate where this meaning comes from in the first place. How do we make sense of what we see to understand the world around us – or, specifically: what makes the correct responses in 'Pyramids and Palm trees' or 'Kissing and Dancing' similar to the image on top? The answer, I propose, is the regularity of our experiences. The palm tree is like the pyramid because both occur in similar situations (deserts); typing is like writing because



Adapted from Bak and Hodges 2003

both involve similar objects (ink and paper); and ironing is like washing because one usually follows the other.

To find out if this is indeed how the brain organises conceptual knowledge, I show participants videos of situations with objects moving around. Initially, the objects and situations do not mean anything to them. But after watching many of these videos and learning about their regularities, I expect objects and situations to acquire meaning for the observers. I then measure participants' brain activity while watching the same videos in an MRI scanner to examine how that acquired meaning is represented in the brain.

We perceive our external world as a constant stream of complex information. Yet, we do not have any problems making sense of most things we go through. To understand our current situation we can rely on prior experience. However, it is unlikely that we will encounter perceptually identical situations multiple times. In order to apply previous knowledge in new situations, we need to abstract away sensory specifics and recognise commonalities across experiences. Having been to a few different restaurants, we know that to order food we need to attract the attention of the waiter, and not of the people at the table next to us – even in a restaurant that we have never visited before.

To support this kind of understanding, representations of the environment in the brain range from perceptual and sensory to more conceptual and abstract. An example of the former is the representation of object shape in inferotemporal cortex (Bao et al. 2020)[†]. Objects with similar shapes produce similar

[†]poem in Appendix

patterns of activity; representations in inferotemporal cortex thus depend on the object's sensory properties. At the other end of the spectrum, representations of abstract schemata have been found in medial prefrontal cortex (Baldassano et al. 2018). When watching movies clips set in airports, similar patterns of activity are evoked at the recurring stages of air travel (like going through security, boarding and take-off) across movies – and even across modalities, when playing audio instead of video airport stories. These representations are completely independent of sensory specifics, and instead capture what it *means* to go through an airport. Here, I aim to identify both these ends of sensory and abstract representations, and the intermediates as well. How is the hierarchy of abstraction, from shape to understanding, organised in the brain?

I will illustrate the hypothesised hierarchy of abstraction with a set of examples (Figure 2.1). If I consider three objects: a tennis ball, an apple, and a football boot, the tennis ball and the apple would be represented similarly in terms of their perceptual features, and different from the football boot (Figure 2.1a). Such shape representations have been found in lateral occipitial complex (Grill-Spector et al. 1999; Kourtzi and Kanwisher 2001) and inferotemporal cortex (Kriegeskorte et al. 2008; Bao et al. 2020). But in terms of their meaning on a conceptual level, the football boot and tennis ball are similar to each other and dissimilar to the apple. What exactly gives an object meaning to us (Love 2017)? I hypothesise that this conceptual similarity between objects is the result of the fact that they occur in similar situations (Tomlinson and Love 2006; Jones and Love 2007): both the tennis ball and the football boot are used for playing sports (Figure 2.1b). This type of similarity is often studied as semantic meaning in language, represented along the temporal lobe (Huth et al. 2012; Bonner and Epstein 2021), or specifically perirhinal cortex (Clarke and Tyler 2014). The interpretation of conceptual meaning here does not depend on language but on the statistical regularities of object occurrence (Mack et al. 2018; Hornsby et al. 2020). The same kind of non-verbal understanding is probed in the 'Pyramids and Palm Trees' test, in which participants judge whether a palm tree or a

pine tree is associated stronger with a pyramid stimulus (Howard and Patterson 1992). Semantic dementia patients with damage to the temporal pole are impaired on this task (Bozeat et al. 2000).

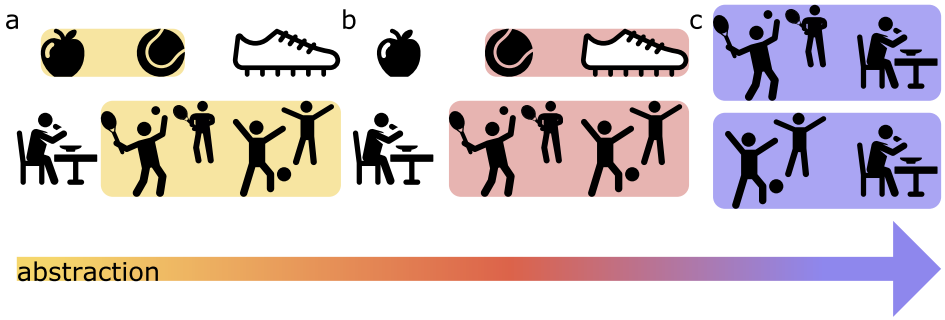


Figure 2.1: A hierarchy of abstraction. a. Perceptual similarity between objects and situations. An apple looks like the a tennis ball, because of their similar shape. Tennis looks like football, because of their similar dynamics. b. Conceptual similarity. In terms of meaning, an tennis ball is more like a football boot, because they appear in similar situations. The tennis and football situation are conceptually similar because they contain similar elements. c. Sequential similarity. Situations that usually follow each other, like having lunch after playing sports, acquire sequential similarity. Along the hierarchy of abstraction, similarities depend less on the sensory properties and more on the statistical regularities of the world.

With conceptual similarity between objects defined by their situational overlap, I turn to similarity of the situations themselves. The football boot is used for playing football, the tennis ball for playing tennis, and the apple occurs in a lunch situation. How are these three situations related? *Perceptually*, tennis and football have some similarities in their dynamics, with players chasing the ball and then kicking or hitting it away again (Figure 2.1a). Early motion detection is thought to take place in V5 (Peuskens et al. 2005), and the superior temporal sulcus has been found to particularly respond to biological motion (Herrington et al. 2011). But what underlies *conceptual* situation similarity? I hypothesise that situations are conceptually similar if they contain similar objects. Football and tennis are conceptually similar, because both involve an opponent, and a well-defined playing arena – but both are conceptually dissimilar from lunch (Figure 2.1b). Situations as particular configurations of objects are related to scenes and contexts; posterior parietal cortex codes for scene structure and content (Chafee et al. 2007; Summerfield et al. 2020), while context representations

have been found in parahippocampal cortex (Aminoff and Bar 2003; Bar 2004; Davachi 2006). Finally, independent of the objects involved, situations link together in time to form sequences or stories. I hypothesise that the sequential regularities of these stories induces similarity between situations in close temporal proximity (Brady and Oliva 2008). Football and tennis are rarely played in quick succession, but because playing sports makes you hungry, both are often followed by lunch (Figure 2.1c). This sequential similarity would thus be high between football and lunch, or tennis and lunch, but not between football and tennis – and analogously, going through airport security acquires similar meaning to boarding the airplane. Learning sequential regularities requires medial temporal lobe (Schapiro et al. 2014), with representations of temporal proximity found in entorhinal cortex (Garvert et al. 2017). Representations of stories of situations have been found in medial prefrontal cortex (Baldassano et al. 2018).

Here, I present a paradigm that probes each of the described similarities in the full abstraction hierarchy in the brain in a controlled way. It measures representations of perceptual object shape, the conceptual meaning of objects from their situation regularities, the perceptual dynamics of situations and the conceptual meaning of situations from their constituent object regularities, and the sequential similarity from stories of situations. I avoid prior stimulus associations by training participants on procedurally generated video stimuli. These videos depict situations wherein objects, built from geometrical three-dimensional shapes, move around in space according to dynamic interactions. Importantly, not all objects can occur in all dynamic interactions, and interactions link together in particular order to form stories from situations. Initially, objects and situations in those stimuli are unfamiliar to participants, but they acquire meaning through their regularities over training. Because I use newly generated stimuli, I can design regularities in such a way that correlations between similarities are minimised, to enable maximal distinction. Additionally, I can randomise regularities across participants, to control for any alternative sources of similarity. After training, I use functional magnetic resonance imaging (fMRI)

to measure brain activity in response to the video stimuli. The imaging task design allows for multiple orthogonal analysis methods, so that one approach can inform the other, to maximise power.

Disappointingly, despite a well-controlled design, a relatively large sample size of 56 participants, and extensive analysis, I did not find convincing evidence of the full gradient of abstraction mapped out in the brain. Perceptual representations generally produced stronger signals than conceptual ones, mostly around visual areas. There are hints of conceptual representations along the temporal lobe, and of story representations in prefrontal cortex, but since these signals do not survive statistical tests nor stand out consistently, I do not present them as reliable results. There are many possible reasons for these ambiguous outcomes, from participant training and task design to analysis choices and noise sources – or the alternative explanation that the hypothesised mechanisms of abstraction in representation cannot be measured in this paradigm. With a well-controlled experiment and a well-defined hypothesis, it is possible to simply be wrong.

2.1 Intermezzo: methods for efficient design

Before I move on to the experiment's methods in Section 2.2, this Section describes two approaches in method development that contributed to the imaging task design detailed there. An important part of task design is optimising the design efficiency: the accuracy with which the model parameters can be estimated from a limited amount of data (Dale 1999). Here I present two design approaches that improve efficiency. The first approach is an algorithm that randomises the order of sequences of stimuli while balancing stimulus precedence. Such balanced randomisation eliminates variance introduced by stimulus history effects, so stimulus responses can be modelled accurately. The same approach also produces efficient designs for methods that involve pairs of subsequent stimuli, like repetition suppression, by providing the shortest possible random sequences that contain the required pairs. The second approach identifies a

source of bias in representational similarity analysis, quantifies the bias, and proposes ways to avoid it. Avoiding this bias through experimental design again eliminates associated variance and improves design efficiency.

Both approaches presented here are the result of encountering a design problem, analysing it, finding a solution, and documenting the solution – only to realise in that final step that others have reported a very similar solution. An appropriate alternative section heading would be 'Reinventing the wheel: methods for efficient design'. In addition to presenting the two methods for realising efficient designs, this Section thus also illustrates that it is usually better to first search the literature and then address a problem than the other way around.

2.1.1 Balancing stimulus precedence

Many experimental paradigms in psychology and neuroscience present stimuli sequentially, meaning that past stimuli might influence the current neural response. Such stimulus history effects are interesting to measure as they may imply rich temporal processing in the brain (Akrami et al. 2018; Fritsche et al. 2022), but they are also important to consider when trying to measure representational similarity via repetition suppression (Gross et al. 1967; Tootell et al. 1995; Grill-Spector et al. 2006) or simply to deconfound the response to the current stimulus. It is therefore desirable to control experimentally the frequency with which different stimulus histories occur. Sequence position can be controlled for by randomising the order of stimulus presentation. However, randomisation by shuffling the presentation order will not fully control for stimulus history effects, as certain pairs of subsequent stimuli might happen to occur more frequently than others. How can I randomise stimulus order with identical precedence across stimuli?

Methods for counterbalancing stimulus history (Aguirre 2007) have focused on hemodynamic response estimation efficiency (Buračas and Boynton 2002), orthogonality of stimulus and sequence position (Nonyane and Theobald 2007), or high hypothesis detection power (Aguirre et al. 2011), but not on provable

uniform sampling. Here I present a sequence randomisation method based on the Euler algorithm introduced by Kandel et al. 1996 that uniformly samples sequences with balanced history of arbitrary length. I show extensions and provide a Python implementation for generating sequences under a range of experimental requirements. In principle, although I developed it independently, this method is highly similar to Aguirre et al. 2011 apart from the sampling mechanism; in practice I hope the description and implementation provided here still prove useful.

Methods

The goal is to generate a random sequence of stimuli that contains each pair of subsequent stimuli equally often. Figure 2.2a shows an example of such a sequence S for four stimuli. The first pair in S is $1 \rightarrow 3$ (labelled A in Figure 2.2a), followed by the pair $3 \rightarrow 4$ (B), then $4 \rightarrow 2$ (C), until all possible pairs have come along. In addition to equal stimulus pair count, S also contains each individual stimulus equally often, except for one additional presentation of the first stimulus. In order to randomly generate these sequences, I will consider them as walks on a graph. I start by constructing a complete digraph G with a vertex for each stimulus as in Figure 2.2b. Each edge represents the transition from one stimulus to another. Sequence S corresponds to a walk that starts at vertex (stimulus) 1, follows the edge (transition) labelled A to vertex 3, *et cetera*, until it arrives back at vertex 1 – after using each edge exactly once. It turns out that any walk that traverses each edge exactly once produces the kind of sequence we are looking for. Such a walk is called an *Euler tour*. Since each vertex of G has equal indegree and outdegree, G is Eulerian, and an Euler tour always exists. The task boils down to sampling an Euler tour on G; Kandel et al. 1996 invented the Euler algorithm that provably samples Euler tours randomly uniformly.

The crucial step of the Euler algorithm is sampling an *arborescence*: a subset of the edges in G so that there is exactly one path from each vertex to a

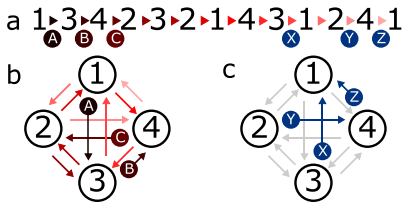


Figure 2.2: Sequences as Euler tours. a. Example sequence S for four stimuli. Each pair is indicated by an arrow coloured by sequence position. b. Complete digraph G for four stimuli with edges coloured to correspond to their position on the Euler tour of sequence S. The first three edges are labelled A, B, C in both the graph G and sequence S. c. Sampled arborescence for generating sequence S, with edges in the arborescence in blue. The last transition from each vertex is taken from the arborescence, shown by the pairs labelled X, Y, Z in S.

root vertex (the first stimulus of the sequence). There are many arborescences for G; Figure 2.2c shows an example arborescence for root vertex 1. I generate an Euler tour by walking along G, picking an unused outgoing edge at random at each vertex – but I only follow the edge that belongs to the arboresence at a vertex once no other unused edges remain. This way, the arboresence serves two purposes: it makes sure that I can complete the tour by returning to the initial vertex in the end, and it prevents me from getting stuck there before having used all other edges.

So how do I sample a random arborescence? The Euler algorithm constructs an arborescence using a backward random walk. Starting at the root vertex v (stimulus 1 in Figure 2.2c), I randomly choose an incoming edge e (e.g. edge X in Figure 2.2c), and transition in the opposite direction of e to its source v' (stimulus 3 in Figure 2.2c). If this is the first time v' (stimulus 3) is reached in the random walk, I add e (edge X) to the arborescence. Then I repeat the same procedure from v' (stimulus 3) to continue the random walk until all vertices have been covered. Kandel et al. 1996 prove that the resulting arborescence is drawn from the uniform distribution over arborescences.

Results

The described method produces random sequences that contain all pairs of stimuli equally often. By adapting the graph G I can accommodate for various

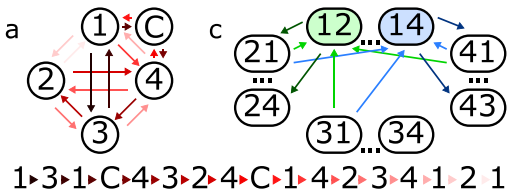


Figure 2.3: Extended sequence graphs. a. Graph G augmented with catch vertex C. Edges are coloured from dark to light red along an example sampled Euler tour. b. Sequence with catch trials corresponding to example Euler tour in *a* c. Graph that produces uniform multi-step precedence. For controlling triplets instead of pairs, redefine vertices as pairs (e.g. $1 - 2/1 - 4$ in green/blue) and add edges between vertices that form triplets (light hue: incoming edges, dark hue: outgoing edges).

experimental requirements within the same algorithm. Here I list a number of such adaptations, further illustrated in Figure 2.3.

- *Repeat stimuli.* To include pairs consisting of identical stimuli, such as $1 \rightarrow 1$, add a loop for each vertex: an edge from the vertex back to itself.

- *Repeat pairs.* There are two ways to generate longer sequences with multiple repetitions of each stimulus pair: either duplicate edges between vertices, or generate multiple tours and stitch them together. The latter guarantees equal numbers of pairs in both halves of the resulting sequence.

- *Add catch trials.* Catch trials test the participant's attention but are not of (main) interest. Randomly inserting catch trials after generating a sequence would result in certain stimulus pairs being 'broken' by the catch trial. A better way to incorporate catch trials in the sequence is to add a catch vertex to G, and adding bidirectional edges until the desired catch frequency is achieved as in Figure 2.3a. Indeed, the example sequence of Figure 2.3b contains all pairs of subsequent stimuli, despite the insertion of catch trials. If it is crucial that catch trials precede each stimulus equally often, the catch vertex should be connected to all stimulus vertices.

- *Control multi-step precedence.* I can control not only for direct stimulus precedence, but for an arbitrary number of steps (Kandel et al. 1996). In

case of two steps, I need each triplet of stimuli to occur equally often in the sequence instead of each pair. This is achieved by redefining G so that vertices represent stimulus pairs (e.g. $1 \rightarrow 2$, $2 \rightarrow 4$) and only adding directed edges between pairs that form triplets (e.g. $1 \rightarrow 2$ and $2 \rightarrow 4$ form $1 \rightarrow 2 \rightarrow 4$) (Figure 2.3c).

- *Induce non-uniform precedence*. So far I have described procedures for generating sequences with uniform stimulus precedence. The method also allows for non-uniform precedence, with certain stimulus pairs occurring more often than others. Graph G can be rewired to produce different pair frequencies, as long as it remains Eulerian: the indegree and outdegree need to be identical for each vertex.

A Python implementation is available at `https://github.com/jbakermans/stimulus-history-euler-tours`.

2.1.2 Bias in representational similarity analysis

Representational Similarity Analysis (RSA) (Kriegeskorte 2008) is a powerful method for making inferences about models and data in neuroscience and psychology. It constructs a Representational Dissimilarity Matrix (RDM) that quantifies a representation distance between a set of conditions, both from data and from theoretical considerations. Then, instead of directly comparing measured responses to hypotheses, RSA uses these RDMs to compare *similarities* between responses to *similarities* between hypotheses. Thanks to this abstraction, RSA naturally allows for various model and data types, and has found applications across domains, species, and modalities (Clarke and Tyler 2014; Hunt et al. 2018; Park et al. 2020; Zhou et al. 2021). However, the same abstraction could lead to biased results. When fitting a General Linear Model (GLM) to estimate condition responses for the data RDM, the distribution of test statistics of random data for particular hypothesis RDMs will be significantly different from zero (Figure 2.4). Estimating condition responses with GLMs, with the

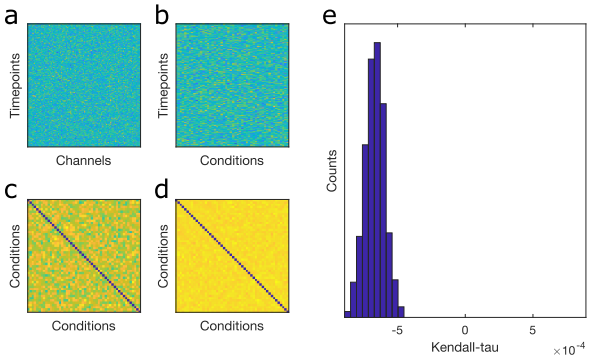


Figure 2.4: Bias in RSA. a. Example random dataset Y. b. Example design matrix X. c. Measured RDM A: correlation distance between responses obtained by fitting a GLM with the design matrix in *b* to the data in *a*. d. Example hypothesis RDM B. e. Bias demonstration: for many different samples of datasets as in *a*, but the same design matrix in *b* and hypothesis in *d*, the distribution of Kendall's tau between A and B is not centred around zero. A neuroimaging equivalent of this simulation would be to do RSA for the same subject in many different searchlights.

potential of biased results, is a common step in RSA pipelines (Clarke and Tyler 2014; Schuck et al. 2016; Theves et al. 2019). What is the source of the bias, and how can it be accounted for?

Here, I show how the bias in RSA results arises from correlations in the GLM design matrix, and how it can be dealt with through cross-block experimental design and analysis. Although developed independently, the results and recommendations here are closely related to Cai et al. 2019. I hope to add to the understanding of those findings by providing a different derivation and demonstration.

Methods

I will derive an expression for the bias of RSA statistics: the expectation over data of the similarity between measured (A) and hypothesised (B) RDM. Assuming the covariance between the upper triangle U of A and B as similarity metric, I thus need to calculate $\mathbb{E}[\mathrm{cov}(U_A, U_B)]$. I assume the following:

- The dataset Y consists of neural activity on V channels (e.g. voxels in functional magnetic resonance imaging) measured over T timepoints
- Measurements are drawn from a standard normal distribution: $Y \sim \mathcal{N}(0, 1)$

- Condition response patterns β are modelled by a GLM: $Y = \beta X + \epsilon$, where X is the design matrix specifying C conditions and ϵ the residuals.
- The measured RDM A is the covariance of the response patterns β:

$$A = \mathrm{cov}(\beta^T) = \frac{1}{V-1}(\beta - \bar{\beta})(\beta - \bar{\beta})^T \tag{2.1}$$

Fitting the GLM gives

$$\beta = (X^T X)^{-1} X^T Y = PY, P = (X^T X)^{-1} X^T \tag{2.2}$$

so then

$$\beta_{ij} = \Sigma_k^T P_{ik} Y_{kj} \tag{2.3}$$

Substituting Equation (2.3) in Equation (2.1) gives

$$A_{ij} = \frac{1}{V-1}\Sigma_k^T P_{ik} \Sigma_l^T P_{jl} \Sigma_m^V Y_{lm}(Y_{km} - \frac{1}{V}\Sigma_n^V Y_{kn}) \tag{2.4}$$

The goal is to calculate $\mathbb{E}[\mathrm{cov}(U_A, U_B)]$, and because

$$\mathbb{E}[Y_{ab} Y_{cd}] = \delta_{ac}\delta_{bd} \tag{2.5}$$

I can substitute Equation (2.4) to get

$$\begin{aligned}
&\mathbb{E}[\frac{2}{C(C-1)-2}\Sigma_i^C \Sigma_{j>i}^C (A_{ij} - \bar{A})(B_{ij} - \bar{B})] \\
=&\mathbb{E}[\frac{2}{C(C-1)-2}\Sigma_i^C \Sigma_{j>i}^C \\
&(\frac{1}{V-1}\Sigma_k^T P_{ik} \Sigma_l^T P_{jl} \Sigma_m^V Y_{lm}(Y_{km} - \frac{1}{V}\Sigma_n^V Y_{kn}) \\
&- \frac{2}{C(C-1)}\Sigma_p^C \Sigma_{q>p} A_{pq}) \\
&(B_{ij} - \bar{B})] \\
=&\frac{2}{C(C-1)-2}\Sigma_i^C \Sigma_{j>i}^C \\
&(B_{ij}\Sigma_k^T P_{ik} P_{jk} - \bar{B}\Sigma_p^C \Sigma_{q>p}^C \Sigma_k^T P_{pk} P_{qk} \\
&- \bar{B}\Sigma_k^T P_{ik} P_{jk} + \bar{B}\Sigma_p^C \Sigma_{q>p}^C \Sigma_k^T P_{pk} P_{qk}) \\
=&\frac{2}{C(C-1)-2)}\Sigma_i^C \Sigma_{j>i}^C \Sigma_k^T P_{ik} P_{jk}(B_{ij} - \bar{B}) \\
=&\frac{2}{C(C-1)-2)}\Sigma_i^C \Sigma_{j>i}^C (X^T X)_{ij}^{-1}(B_{ij} - \bar{B})
\end{aligned} \tag{2.6}$$

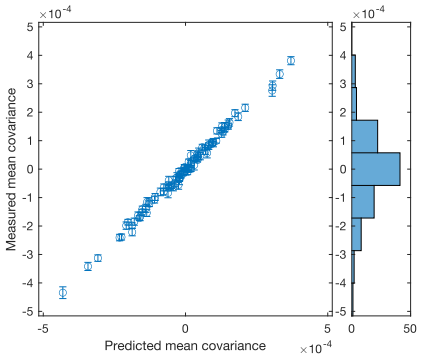


Figure 2.5: Simulated and predicted bias. Mean covariance between measured and hypothesis RDMs across sampled datasets for many different experiments (points on graph), plotted against the covariance expectation from Equation (2.6). Each experiment has a random design matrix X and hypothesis B; the histogram inset shows no bias across experiments. However, the additional bias variance will reduce design efficiency.

Results

Indeed, simulating many different RSA experiments, sampling a different hypothesis RDM B and multiple datasets Y for each, I find that Equation (2.6) accurately predicts the average similarity between A and B (Figure 2.5). Since there is no bias across different Bs, randomising B across subjects will prevent group-level bias, but the additional variance due to subject-level bias will reduce the design efficiency. It is therefore preferable to avoid bias at all.

Equation (2.6) shows that the expected similarity between A and B will be zero if the design matrix X is orthogonal. To achieve that, one option is orthogonalising the design matrix, which is undesirable because then X will depend on orthogonalisation order. A better alternative is using a multi-block experimental design, where conditions are repeated in separate blocks, and only including dissimilarity across blocks for RSA (Figure 2.6) (Walther et al. 2016; Baram et al. 2021). The entries in A that measure the dissimilarity between conditions across blocks will not induce bias, because their condition regressors are orthogonal. This cross-block section of A, marked in red in Figure 2.6c, is not necessarily symmetric, but can be made so by averaging the upper and lower triangles. Its diagonal is no longer zero, but holds meaningful information

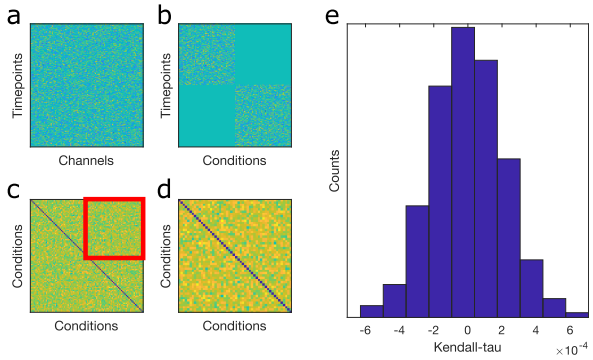


Figure 2.6: Bias-free RSA. a. Example random dataset. b. Example design matrix with conditions repeated across two blocks. c. Measured RDM $\mathcal{A}$ calculated using the design matrix in *b*. The red square marks the cross-block dissimilarities. Example hypothesis RDM $\mathcal{B}$. e. No bias in the distribution of Kendall's tau when comparing the cross-block dissimilarities of *c* to the hypothesis in *d* for many different samples of datasets as in *a*.

about representational dissimilarity between different repetitions of the same condition. Figure 2.6e shows that when using cross-block dissimilarity, the distribution of Kendall's tau across datasets is centred around zero.

2.2 Methods

In this experiment I want to probe representations that range from sensory to abstract, in a controlled way. To avoid prior associations, and to maximise randomisation, I present participants with procedurally generated stimuli that they have never seen before the experiment. These video stimuli render situations that consist of three-dimensional objects in dynamic relations, as in the example stimuli in Figure 2.7. The apple and the tennis ball of the example have similar shapes; the three-dimensional objects in the stimuli are designed to measure responses to shape (Figure 2.7a). But in terms of conceptual meaning, the tennis ball is more similar to the football boot of the example, because they appear in similar situations. In the video stimuli, I implement this conceptual meaning through a specific distribution of objects over situations: objects that occur in the same situations, acquire similar meaning (Figure 2.7b). The situations in the video stimuli themselves have perceptual similarities (Figure 2.7a) – for example the object speeds and the distance between them – but situations also acquire

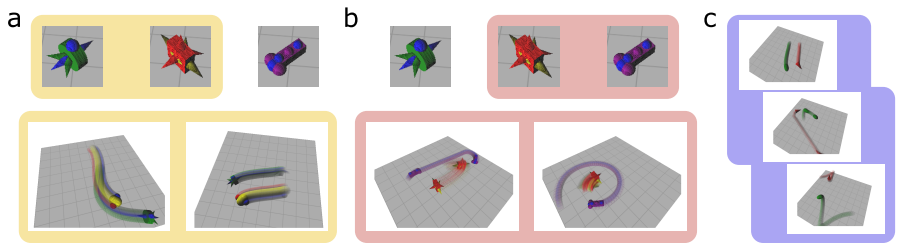


Figure 2.7: Experiment similarities. Participants watch video stimuli (bottom) that render three-dimensional objects (top) in dynamic relations, to emulate the similarities of Figure 2.1 a. Perceptual similarity between objects and situations. The green-and-blue and red-and-yellow objects have similar shapes: both are short and pointy. The two situations in the depicted video stimuli have similar dynamics, with objects at similar speeds and distances b. Conceptual similarity. Because the two different situations both contain the red-and-yellow and purple-and-blue objects, these objects and the situations become conceptually similar. c. Sequential similarity. In video stimuli that play out stories of situations, the situations that occur in close temporal proximity acquire sequential similarity.

conceptual meaning. Just like playing football and tennis have similar meaning, because both involve an opponent and a well-defined arena, the distribution of objects over situations induces conceptual similarity between situations when they share many objects (Figure 2.7b). Finally, I generate video stimuli of objects in sequences (or 'stories') of dynamic relations, to probe situation similarity from temporal proximity (Figure 2.7c) – like playing football and having lunch in the example.

To identify brain regions that represent the world at these different levels of abstraction, I will need to generate a large set of stimuli, train participants on the their regularities, and then present the same kind of stimuli in an MRI scanner. I specify how I do that in Section 2.2.1. Then, I describe the details of data acquisition and preprocessing in Section 2.2.2. I have designed the task in the scanner in a way that allows for multiple types of analyses, with one analysis method informing the other. Section 2.2.3 presents the analysis methods that I apply to the neural data to find signatures of the the described similarities in brain representation.

2.2.1 Task

Here I describe the experimental task. At its core is a set of video stimuli with specific regularities. Before the neuroimaging session, participants carry out an online training task on three days to get familiar with these regularities. On the final day, after three of those behavioural training sessions, participants are invited for an MRI scan. During the MRI scan they perform another task, designed to measure brain responses to the stimuli they by then have extensive experience with.

Stimuli

The video stimuli in the task feature three-dimensional objects. I constructed eight different objects from three-dimensional primitive shapes like cuboids, spheres, and cones, in various colours. To maximally activate shape-responsive brain areas, the objects vary along axes that these areas are particularly sensitive to: a 'short and fat' versus 'long and thin' dimension, and a 'pointy' versus 'stubby' dimension (Figure 2.8a). The video stimuli present situations by placing two of these objects in a dynamic relation. There are eight different dynamic relations (Figure 2.8b). Each dynamic relation is defined by its equations of motion, that can also be stitched together to form stories. I numerically integrate these equations of motion to produce trajectories, then assign objects to the trajectories, and then render the resulting three-dimensional scene to generate moving stimuli. The camera viewpoint remains fixed during a single stimulus but varies randomly between stimuli. The full stimulus generation pipeline is implemented in custom GameMaker:Studio software.

Crucially, there are specific regularities for which situations contain which objects, and how situations form stories. These regularities aim to induce abstract, conceptual similarities between objects and situations. Therefore not all objects can appear in all situations. The matrix in Figure 2.8c specifies the distribution of objects over situations. Each row is an object, each column a situation, and a black entry means the corresponding object can occur in

the corresponding situation. Specifically, situation 1 (for example, attract) can contain object 1, 2, 3, and 4 – not all at the same time, because there are always only two objects in a stimulus, but any combination of these four. Conversely, object 1 (for example, the green-and-blue pointy one) can appear in situation 1, 4, 5, and 8. I hypothesise that objects that share situations become similar (like object 1 and 2, or the tennis ball and football boot; Figure 2.8d), and situation that share objects become similar (like situation 1 and 5, or playing football and tennis; Figure 2.8e). Finally, stories of situations are defined by a situation transition matrix, plotted in Figure 2.8f. These story stimuli link situation together in sequences; the sequence order is defined by the transition matrix. The topology specified by the transition matrix here is a ring, like days in a week. Specifically, situation 1 (for example, attract) is followed by situation 7 (for example, align), which is followed by situation 4 (for example, collide), *et cetera*, until situation 8 is followed by 1 again. Here, I hypothesise that situations in close temporal proximity (like situation 1 and 7, or playing football and lunch; Figure 2.8g) become similar. An additional important detail on story stimuli is that they contain simple shapes, like cones and ellipsoids, instead of the objects in Figure 2.8a. If story situations would use the same objects as the separate situation stimuli, subsequent story situations would need to contain the same objects – and conceptual and sequential similarity would end up correlated. Figure 2.8h summarises the regularities that cause the various abstract similarities of the experiment.

Together, the regularities underlying the video stimuli create a virtual conceptual environment. Analogous to the real world, objects in similar situations are conceptually similar (like the tennis ball and the football boot), and situations with similar objects are conceptually similar (like playing tennis and football), and stories make sense through sequential structure (like playing football then having lunch). But contrary to the real world, I have precise control over these regularities. I can choose which objects and situations are conceptually similar, by assigning particular concrete objects (for example, the green-and-blue

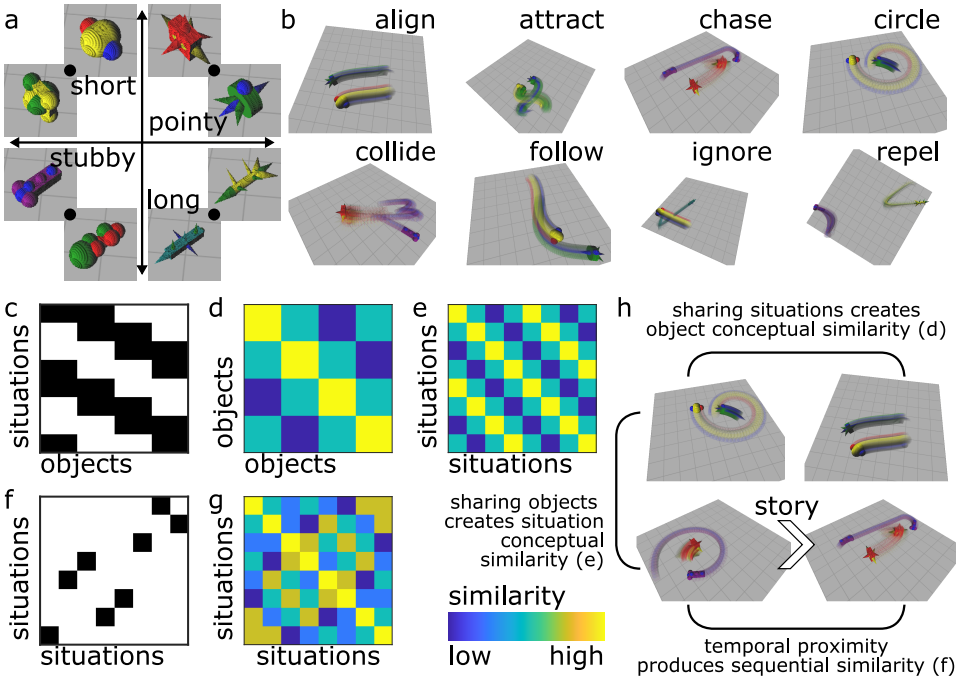


Figure 2.8: Stimuli. a. The stimuli can contain eight different rendered three-dimensional objects, with shapes that occupy the four quadrants of the space spanned by a 'stubby' versus 'pointy' axis and a 'long and thing' versus 'short and fat' axis. b. Each situation video stimulus consists of two of these objects in one of eight dynamic relations. c. Conceptual similarity results from the distribution of objects over situations. A black matrix entry means that the row's situation can contain the column's object. Certain situations have symmetric object roles (for example, both objects attract each other) while others are asymmetric (for example, one object follows the other), but I do not distinguish between roles: each object can play either role. d. Resulting conceptual situation similarity: situations that can contain the same set of objects are highly similar. e. Conversely, objects that can occur in the same set of situations have high conceptual similarity. f. Stories of situations are defined by a transition matrix. A black entry means that the row's situation is followed by the column's situation. g. Resulting sequential situation similarity: situations that occur near each other in time in the story stimuli are more similar. h. Summary of how the stimulus regularities induce abstract similarities.

pointy one) and relations (for example, attract) to the abstract objects and relations in the rows and columns matrix A and B. This has three important advantages. First, any influence of prior experience is avoided. Second, I can make sure that different perceptual and conceptual similarities are distinguishable. Football and tennis have similar perceptual dynamics (running towards a ball, then hitting it away again) but also similar conceptual meaning. By carefully choosing the assignments of concrete objects and relations to the abstract ones, I can make different similarities as uncorrelated as possible. Third, I can

randomise these assignments across participants. Because of this randomisation, requiring different stimuli across participants, I end up generating over 50,000 video stimuli.

Training

The purpose of the stimulus regularities is to induce abstract similarities. To achieve that, participants need extensive experience with the stimuli to learn about their regularities. Participants therefore carry out a training task in a custom-built web application on three subsequent days. There are two parts to the training task: task A and task B. I will describe both in detail here, but the best way to get an impression of the task is probably by trying it out in a demonstration version, available at `https://jjwbakermans.com/task_demo/?SUBJECT_TAG=s01id01`.

Task A, illustrated in Figure 2.9, familiarises participants with the object-situation associations that underlie conceptual similarity. In the first part of task A, a prediction game implicitly teaches these object-situation associations (Figure 2.9a). Participants watch a video stimulus of two objects in a dynamic relation, then one object turns invisible (Figure 2.9b). Shortly after, the movie ends, and participants are asked where the hidden object has gone. Participants are also asked to classify the dynamic relation of the situation in the video. The second part of task A explicitly tests participants' object-situation associations (Figure 2.9c). It poses two multiple-choice questions: first, *Which object is most likely to appear in [dynamic relation]?*, with five objects to choose from; second, *Which relation is most likely to contain the object below?*, with five dynamic relations to choose from. The available options are selected so that only one of them is correct. In addition to the correct answer, there are exactly four incorrect alternatives: there are always four objects that a given relation cannot contain, and four relations an object cannot be part of (see Figure 2.8c). Task A consists of 40 prediction game trials, and 8 pairs of object-situation questions. Participants always receive feedback on their responses.

In a training session, task A is repeated twice, with small but important differences in the prediction game. The first time, both objects are visible for a long time ($4 - 7$ seconds) and predictions only require short extrapolation ($0.4 - 0.7$ seconds). The dynamic relation type is printed so that classification is trivial. This setup allows participants to become familiar with the dynamics of each possible relation. The second time, the relation type is not printed, and the time with both objects visible is shorter ($0.3 - 1.3$ seconds, respectively) while the extrapolation time is longer ($0.8 - 1.3$ seconds in both). To make good predictions over longer periods, it is crucial to recognise the situation's dynamics. And because certain situations can only contain certain objects, for this recognition it becomes increasingly important to know these object-situation associations when the dynamics are visible only briefly. Two additional features of the prediction game stimuli further encourage participants to rely on situation-specific dynamics (and thus on object-situation associations). First, the moment at which one of the objects turns invisible is not picked randomly. I define key frames for each dynamic relation: moments where knowing the specific relation type is particularly important for predicting the future. For example, when two objects are just about to meet, the future will look very different if they are in a collide relation (they will bounce off each other) than if they are in a attract relation (they will smoothly continue their trajectory). The prediction game stimuli show both objects before hiding one of them right before the key frame. Second, the incorrect alternatives are sampled from where the hidden object could have been, if the dynamics were governed by a different type of relation.

Task B, shown in Figure 2.10a, teaches participants the structure of stories that produces sequential similarity between situations. In each trial, participants watch a story video stimulus, consisting of a sequence of three situations (Figure 2.10b). The three situations each last 3 seconds, so the full story stimulus takes 9 seconds. Each three-situation story is a sub-sequence of the full eight-situation sequence defined by the transition matrix in Figure 2.8f. At the end of the video stimulus, participants are asked to predict which situation will occur

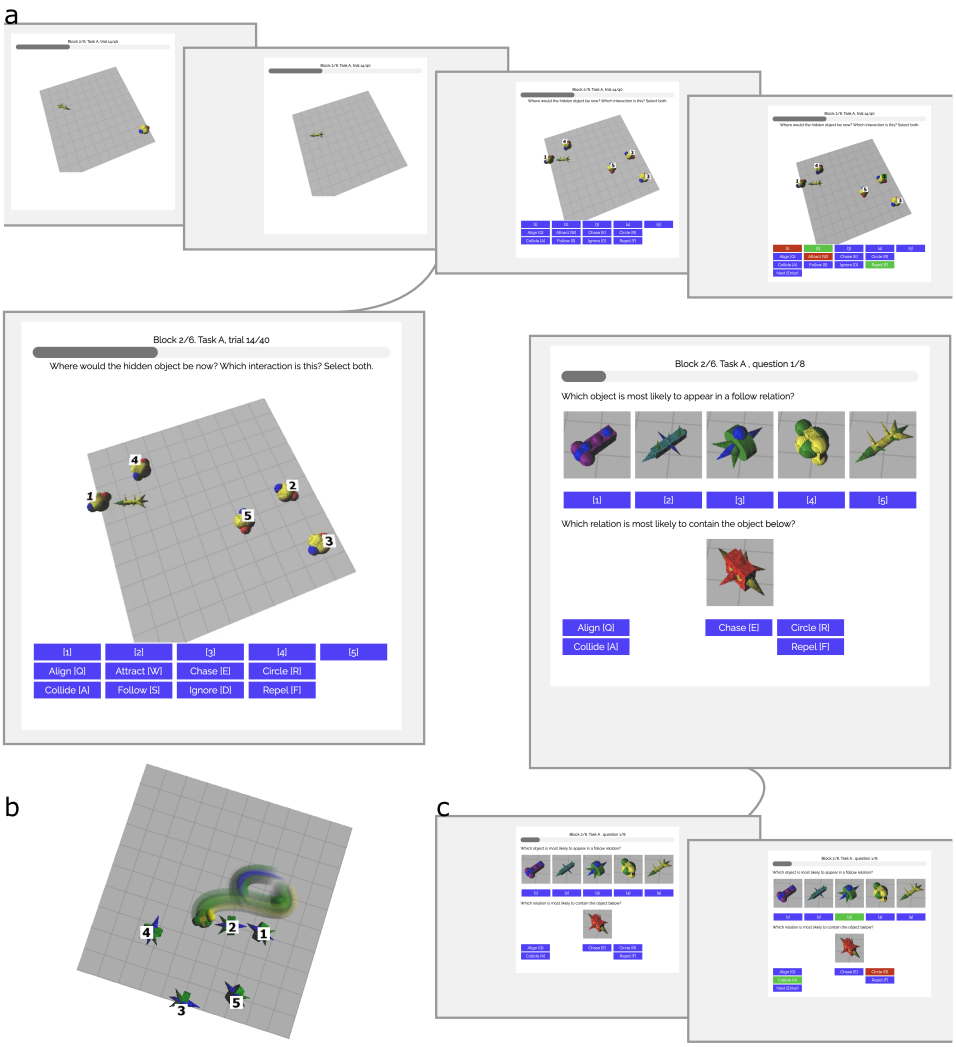


Figure 2.9: Task A. a. Web application page during a trial of the prediction game in task A. Initially, the participant sees both objects. Then one of them disappears, and shortly after the participants has to guess the hidden object's position from five options. They also report the situation they have seen. They receive feedback and then move on to the next trial. b. Example stimulus for the motion prediction game. c. Web application page for a pair of which-object and which-relation questions, with feedback after response.

soonest after. Importantly, this is a multiple-choice question, and the situation that immediately follows the one at the end of the story is not always available as an option. The correct answer might be two or three steps away. Participants therefore have to attend to long-range sequence structure, beyond just pairs of subsequent situations. It is impossible to get all predictions right from the start, but as participants experience more stories, they can piece together the full eight-situation sequence from the parts they have seen. There are 32 story

stimuli in task B, and task B is repeated twice in one training session.

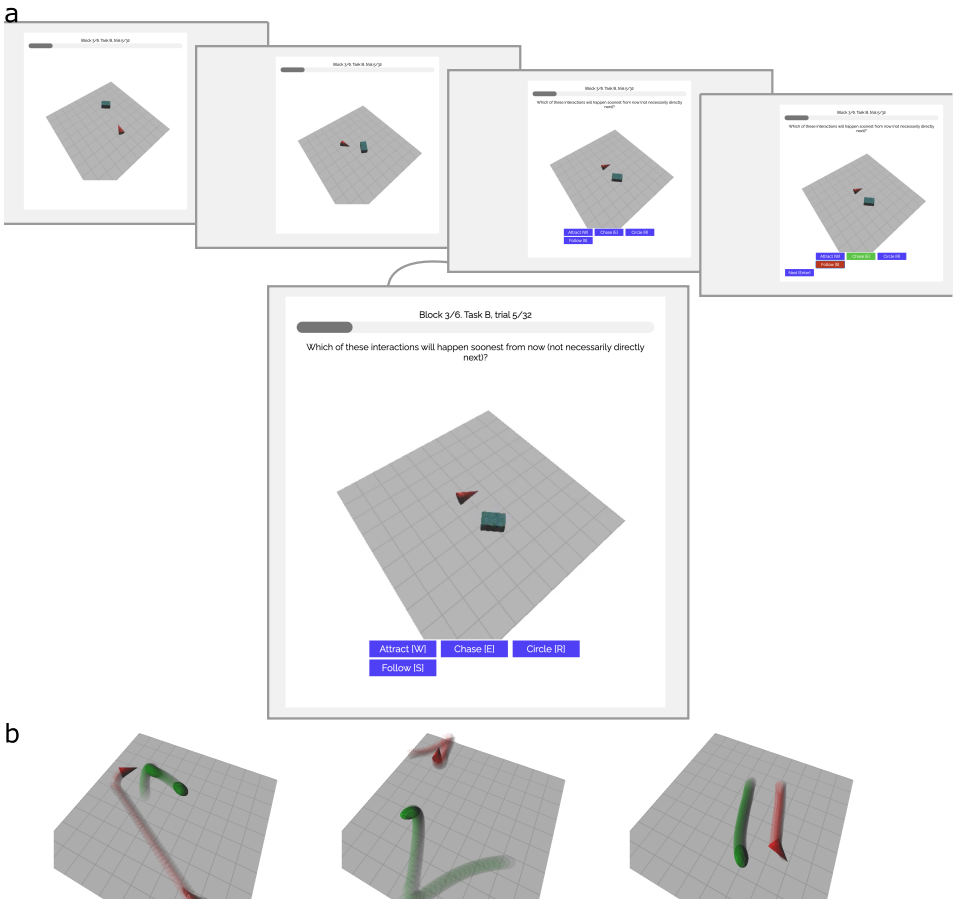


Figure 2.10: Task B. a. Web application page during a trial of task B. The participant watches a story consisting of three situations, and is then asked which situation will occur soonest after from four options (not necessarily including the immediate next situation). They receive feedback on their response in each trial. b. Example task B stimulus of two objects in three sequential situations. For visualisation the full three-situation stimulus is split into separate parts, but in the task this is one continuous movie stimulus without interruption.

In summary, the full training task is structured as follows: task A (40 'easy' prediction trials, 8 questions) – task A (40 'hard' prediction trials, 8 questions) – task B (32 story trials) – task A (40 'easy' prediction trials, 8 questions) – task A (40 'hard' prediction trials, 8 questions) – task B (32 story trials). Participants earn £0.01 in bonus payment for each correct response. The full task takes about an hour to complete. The web application is implemented using HTML, javascript and PHP; all responses are written to a web server. Participants perform the task on three consecutive days: at home, at a time that suits them, on the first

two days; and on-site, an hour before the MRI scan, on the final day.

Magnetic resonance imaging

After completing three training sessions, participants have extensive experience with the stimuli and their regularities. They are then invited for an MRI scan. During the fMRI session, participants carry out another task, designed to measure brain activity corresponding to each type of stimulus similarity. Like the training task, the scan task involves blocks of task A and blocks of task B.

In the task A block in the scanner, participants watch video stimuli in quick succession, as shown in Figure 2.11a. Each stimulus lasts 2 seconds, and there are no inter-stimulus intervals. As during training, these video stimuli show situations of two objects in a dynamic relation. However, in the scanner I include only 8 different situations: one situation of one particular pair of objects for each dynamic relation. There are two reasons for doing so. First, in repetition suppression analysis (also see Section 2.2.3) I measure similarity for subsequent pairs of stimuli, so if I want multiple repeats of subsequent pairs, I cannot have too many different stimuli. Second, because situation overlap induces object similarity and object overlap induces situation similarity, these two conceptual similarities are unavoidably correlated across all situations. But if I carefully select the situations I show in the scanner, I can minimise correlations within that selection – for example by picking objects that are as different as possible in situations that are similar. I will specify the exact stimulus selection in Section 2.2.3. Given these 8 selected situations, I present all pairs of subsequent stimuli excluding same-stimulus repeats, and I repeat all ordered pairs 3 times in a block. Here by *repeating* a stimulus I mean *presenting a situation with the same objects in the same dynamic relation*; the actual trajectories and the rendered viewpoint are different in each video. To make sure participants are attending the stimuli on the screen, I randomly interleave catch trials, on average once every 9 trials. In a catch trial, the 2-second video continues for another 1 second with one object hidden, and afterwards participants get 4

seconds to predict the location of the hidden object as in the training task. Using the algorithm from Section 2.1.1, having 3 repeats of each ordered pair of 8 stimuli plus catch trials without same-stimulus pairs produces $9 \cdot 8 \cdot 6 + 1 = 217$ trials per block.

Task B in the scanner is almost identical to scanner task A, apart from two differences (Figure 2.11b). First, to mirror the story stimuli during training, the objects are replaced by primitive geometric shapes like cones and ellipsoids. Second, the catch trial question reflects task B from training: participants are asked which of four dynamic relations would occur soonest after the catch trial's stimulus, and need to respond within 5 seconds.

The full scan task involves two blocks of task A and two blocks of task B, with order *ABAB* or *BABA* randomised across participants. Participants hold a button box with four buttons to respond in catch trials; they earn £0.10 in bonus payment for each correct response. There is a 10 second break between blocks. In total, the scan task takes about 38 minutes. It is implemented using the Psychtoolbox for MATLAB to guarantee precise timing information.

2.2.2 Data

Having defined the task performed by participants, I will now provide the details of data acquisition. This section describes the specifics of participants included in the experiment, the parameters of the MRI scans for data collection, and the preprocessing steps required to produce the full analysis-ready data set. Parts of the descriptions here have previously been reported in Kuechenhoff 2021, a MSc thesis that I supervised.

Participants

I recruited 67 participants to take part in the experiment over two periods of data collection, from April to June 2019 and from March to May 2021. Seven of them dropped out before completing the scan session. A further four participants were excluded after the scan session due to excessive motion during

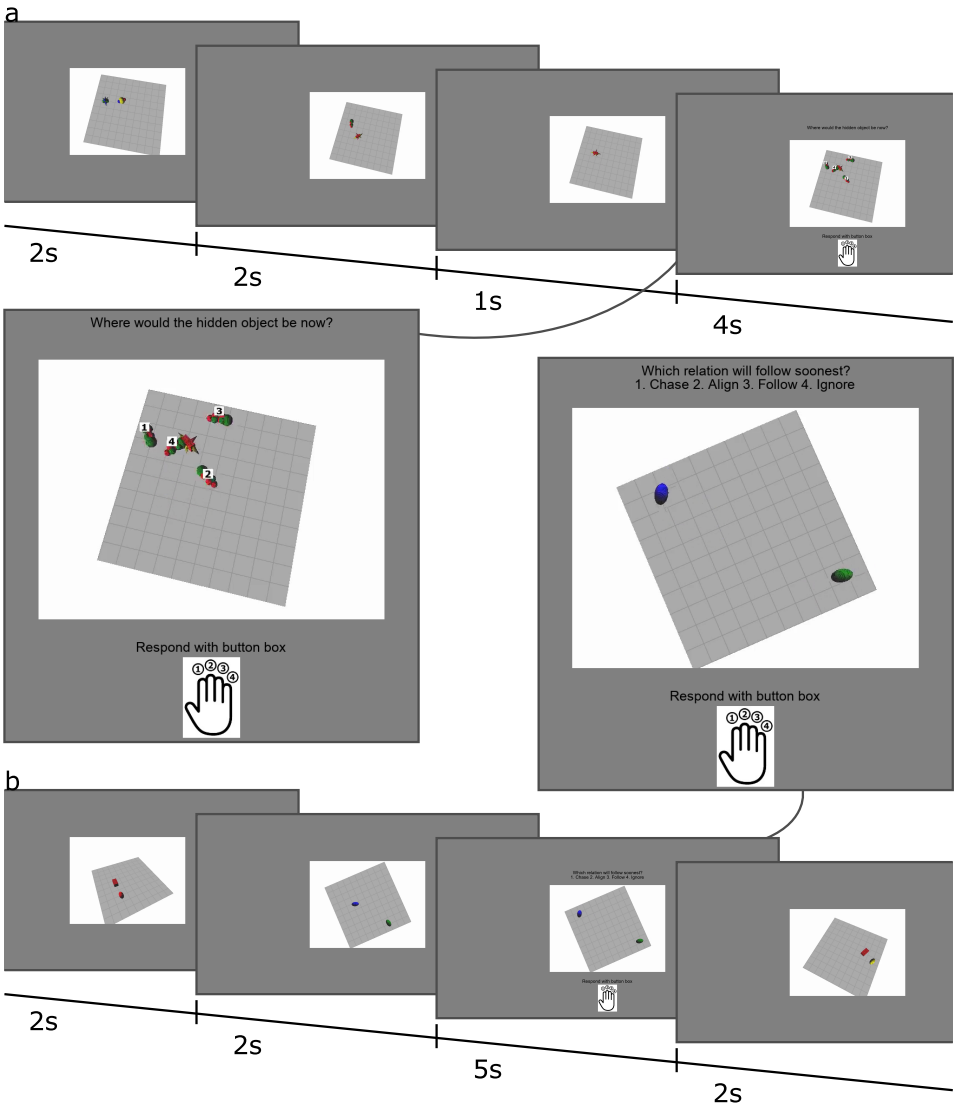


Figure 2.11: Scan task. a. Task A in the scanner. Situation stimuli as in training task A are played for 2 seconds in rapid succession. In a catch trial one of the objects disappears for 1s second after the 2s stimulus, and afterwards the participant needs to respond where that object has gone within 4s using the button box. b. In task B in the scanner, stimuli are again played for 2 seconds. In a catch trial, the participant is asked which of four situations follows soonest after the presented stimulus in the stories that they have learned during training. They have to respond within 5s with the button box.

the MRI scan, defined as the number of instantaneous movements over 0.5 mm exceeding the mean plus 1 standard deviation (Power et al. 2012). Therefore in total I included $N = 56$ participants, aged between 19 and 37 years, mean age $\pm$ standard deviation 24.7 ± 4.2; 36 of them are female. All participants are fluent English speakers, have normal or corrected to normal vision and had no known neurological conditions nor showed contraindications for MRI

scanning. Prior to training, participants gave informed consent for participation in the study but were naive to its exact purpose. They were financially reimbursed for their time (£10 per hour for the training, £15 per hour in the scanner, and up to £16 bonus payments for correct responses during training or scan tasks). The study was approved by the University of Oxford Central University Research Ethics Committee (reference R51214/RE003). MRI scans took place at the Wellcome Centre for Integrative Neuroimaging (WIN), at the John Radcliffe Hospital in Oxford, UK.

Acquisition

I collect MRI data on a 3 Tesla Siemens Prisma scanner, using a 32-channel head coil. A scan session starts with a short localiser sequence, and then I collect the functional data while the scan task runs. The task receives a trigger signal from the scanner to synchronise task and data timings. Functional images are acquired using a T2*-weighted echo-planar imaging sequence (EPI) with a multi-band acceleration factor of 3 and within-plane acceleration factor (iPAT) of 2. I collect volumes of $2 \times 2 \times 2$ mm resolution, with a repetition time $\mathrm{TR} = 1.235$ secs, echo time $\mathrm{TE} = 20$ ms and flip angle of 65°. After finishing the scan task, I record field map dual echo-time images ($\mathrm{TE}_1 = 4.92$ ms, $\mathrm{TE}_2 = 7.38$ ms, whole-brain coverage, voxel size $2 \times 2 \times 2$ mm) to correct for geometric distortions induced by susceptibility-induced field inhomogeneities. Finally, I collect a high-resolution structural scan using a T1-weighted MP-RAGE sequence with $1 \times 1 \times 1$ mm voxels.

Preprocessing

For data preprocessing I use fMRI Expert Analysis Tool (FEAT) (Woolrich et al. 2001) which is part of FMRIB's Software Library (FSL) (Jenkinson et al. 2012). I run BET brain extraction on the structural and the fieldmap images to prepare them for FEAT preprocessing. I then run FEAT preprocessing on the functional data. This includes brain extraction, motion correction to align the functional

volumes to the pre-saturation volume using MCFLIRT (Jenkinson et al. 2002), B_0 unwarping using the fieldmap images, and registration. Registration calculates a linear mapping from the functional images to the high-resolution structural space using FLIRT (Jenkinson and Smith 2001), and a non-linear mapping from structural space to MNI152 standard space with FNIRT. During preprocessing, I apply a high-pass temporal filter with a cutoff at 100 seconds, but I do not apply any spatial smoothing. Finally, I use MELODIC (Beckmann and Smith 2004) for independent component analysis to decompose the four-dimensional functional data into spatial and temporal components.

After this initial preprocessing stage, I take additional steps aimed at reducing noise in the functional data. I track down noise components among the independent components identified by MELODIC so they can be removed from the data. For that purpose I first run FIX (Salimi-Khorshidi et al. 2014), which automatically classifies independent components as 'noise', 'signal', or 'unknown', based on a training set of labelled components from a similar dataset (with $TR = 1.3$s, $6\times$ multiband acceleration, $2 \times 2 \times 2$mm resolution, no spatial smoothing, 100s highpass temporal filtering). I then manually verify the resulting classification, following guidelines in Griffanti et al. 2017. Then I use FIX to remove all unique variance associated with the noise components, and also variance for motion confounds produced during motion correction. After this cleanup, I apply SUSAN (Smith and Brady 1997) spatial smoothing of 5.0mm full-width at half maximum – usually this is part of FEAT preprocessing, but by doing it after noise cleanup I make sure that analysis pipelines on smoothed and unsmoothed data start from the same input. Finally, I generate additional nuisance regressors that can be included in later analyses. First, I produce motion outlier regressors: I detect timepoints where functional data may have been corrupted by large motion, and include a regressor for each such timepoint to negate its effect. Second, I calculate a cerebrospinal fluid regressor that tracks the average signal in the ventricles. I identify cerebrospinal fluid voxels through segmentation using FAST (Zhang et al. 2001) on the structural image, then warp

this mask to functional space, erode the mask to exclude voxels at the edges of the ventricles and average the BOLD-signal across the remaining voxels.

2.2.3 Analysis

With the data preprocessed and cleaned, they are now ready for analysis. The central goal is to find representations of stimuli at different levels of abstraction in the brain. Stimuli can be similar because of what they look like, or based on what they mean; how are these similarities reflected in the BOLD signal measured in each voxel during the MRI scan? This section presents different approaches to revealing neural representations of stimulus similarities. But first, I will more precisely define the different hypothesised similarities they aim to find.

There are five types of similarity that this experiment is designed to identify. *Perceptual object similarity* is proportional to the number of matching shape dimensions ('short and fat' versus 'long and thin', and 'pointy' versus 'stubby') between objects. Two long and spiky objects are maximally similar to each other, and maximally different from a short and stubby object. *Conceptual object similarity* is proportional to the number of situations that two objects can both occur in. Objects that share all their available situations are maximally similar, objects that never share situations are maximally different. *Perceptual situation similarity* expresses how similar the dynamics between two situations are. I calculate it by collecting a number of features of the dynamics, like average object speed, distance, and relative orientation between objects over many trajectories for each situation, and then correlating these feature vectors between situations. *Conceptual situation similarity* between two situations is given by their object overlap: it is proportional to the number of objects that can occur in either. Situations that can contain the same four objects are maximally similar, while situations that cannot have any objects in common are maximally different. *Sequential situation similarity* is proportional to the negative distance between situations in stories. Situations are maximally different if one

follows the other immediately, and get more different the more situations can come in between.

In the scanner, I do not present objects in isolation, but only in pairs in situations. Therefore I cannot directly measure similarities between two objects, but only between two pairs of objects. How does the object similarity of Figure 2.8d translate into similarity between scan stimuli? One option is to sum all four possible object similarities between two stimuli: between object 1 in stimulus 1 and object 1 in stimulus 2, and object 1 in stimulus 1 and object 2 in stimulus 2, *et cetera*. But that has as consequence that the self-similarity for a stimulus with two objects that are similar to each other is higher than the self-similarity for a stimulus with two objects that are different from each other. Instead, I choose to define object similarity between stimuli based on the best matching pairs of objects from both stimuli: either the similarity between object 1 in stimulus 1 and object 1 in stimulus 2 plus the similarity between between object 2 in stimulus 1 and object 2 in stimulus 2, or the opposite pairing – whichever total similarity is higher. This choice produces stimulus similarity matrices with a diagonal that is constant and maximal.

Because task B in the scanner shows stimuli that contain simple shapes instead of objects, I can only look for *object* similarities in task A blocks, but for *situation* similarities in both. Across participants, the *perceptual* similarities are always the same: the green-and-blue short pointy object always looks similar to the red-and-yellow short pointy object, and the align-dynamics always look similar to follow-dynamics because of similar between-object distances and speeds. The *abstract* similarities are different for each participant because of the randomisation of which situation corresponds to which row and which object to which column in the matrix of Figure 2.8c.

All considerations above, given the particular selection of stimuli presented in the scanner in Figure 2.12a, produce five hypothesis stimulus similarity matrices, plotted for a single participant in Figure 2.12b-f. In the following, I will

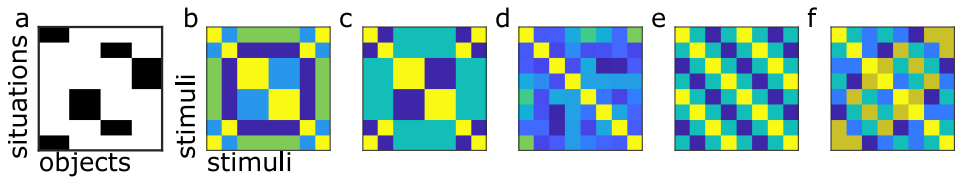


Figure 2.12: Scan stimuli similarities. a. Selection of stimuli presented in the scanner, chosen for minimal correlations between the resulting similarity matrices. There is a single stimulus for each situation (matrix row), containing two out of the four possible objects in that situation (matrix columns). Each row therefore defines one stimulus. The following similarity matrices all plot similarity between these eight stimuli for one particular participant – because of randomisation, the perceptual and abstract similarities are shuffled relative to another across participants. b. Perceptual object similarity. c. Conceptual object similarity. d. Perceptual situation similarity. e. Conceptual situation similarity. f. Sequential situation similarity.

describe different ways of finding neural signatures of the hypothesised similarities.

Repetition suppression

On repeated presentation of the same stimulus, neurons that respond to that stimulus will reduce their activity (Gross et al. 1967). This phenomenon is referred to as *repetition suppression* (Grill-Spector et al. 2006). Importantly, that does not only apply to identical stimuli. On presentation of different stimuli that are similar on some dimension, neurons that represent that dimension also show repetition suppression (Lueschow et al. 1994; Grill-Spector et al. 1999). And because repetition suppression can be measured in fMRI (Demb et al. 1995), it turns out to be a useful tool to index neural representations (Malach 2012; Barron et al. 2016). In the context of the experiment here, I expect a shape-responsive area to show suppression when two subsequent stimuli both contain pointy objects. On they other hand, a brain region that represents conceptual meaning should suppress its activity when the situations of two subsequent stimuli have become conceptually similar over training. Therefore each of the hypothesised similarity matrices of Figure 2.12 implies a pattern of suppression through time, if a voxel were to represent that similarity.

I model these hypothesised timeseries of suppression in a general linear model (GLM) using FEAT. I separate regressors for task A and task B in the same model. Each similarity hypothesis contributes one parametric regressor to the design of the GLM (except that there are no object similarity regressors in task B). That means that I also model conceptual similarity – learned in, and most relevant for, task A – during task B, and sequential similarity – learned in, and most relevant for, task B – during task A. In addition to the hypothesised stimulus similarities, I add one more parametric similarity regressor: a low-level pixel-based similarity. This regressor models strong suppression if the pixel values of the first frame of two subsequent stimuli are highly correlated. I expect this regressor to strongly predict activity in early visual areas, so it can serve as a sanity check. I include separate onset regressors for each stimulus, and also regressors for the previous stimulus identity. Catch trials, and trials directly following catch trials, are excluded from the similarity regressors and have their own onset regressor. I also add a regressor for button press responses during catch trials. Taken together, there are 45 regressors in total, which amounts to 90 as I also include temporal derivatives of all regressors. Finally, I include the motion outlier volumes and the cerebrospinal fluid signal calculated during proprocessing as nuisance regressors.

FEAT convolves all regressors, except for the nuisance regressors, with the default gamma hemodynamic response function with a width of 3 seconds and a lag of 6 seconds, and then fits the GLM. The input to FEAT is the smoothed preprocessed functional data in the subject functional space. I specify a number of contrasts on the output parameter estimates for use in the group-level analysis. There is a one-hot contrast for each parametric similarity regressor, and a two-hot contrast that sums the same similarity in task A and task B.

Representational similarity

I have introduced representational similarity analysis (RSA) in Section 2.1.2, and I will now show its application in the context of this experiment. RSA considers

representational dissimilarity matrices (RDMs), and compares hypothesis RDMs to measured data RDMs. The five hypothesised stimulus similarity matrices immediately provide the hypothesis RDM by simply taking the negative similarity. How do I get the data RDM? I use a searchlight-based approach to calculate one data RDM for every voxel. First, I construct a searchlight for every voxel: a list of the voxel's 100 closest neighbours. Then, for each voxel in a given searchlight, I find the average response to each stimulus. That produces a 100-dimensional vector of activations across voxels for each stimulus. To get the representational dissimilarity between stimuli, I calculate correlation distances between these searchlight representation vectors. I can then test if this searchlight represents stimuli according to any of the hypothesised similarities by comparing the hypothesis RDMs to the resulting data RDM, for example through rank-correlation or regression. Because it considers patterns of activity across voxels in a searchlight, RSA is a multivariate method, in contrast to the univariate repetition suppression approach that treats voxels separately. And because RSA acts on average stimulus responses, whereas suppression compares pairs of subsequent stimuli, the two methods are orthogonal – meaning that one can inform the other without 'double dipping' (Kriegeskorte et al. 2009).

The RSA pipeline is built on the RSA toolbox (Nili et al. 2014), which is based on statistical parametric mapping (SPM) 12 (Friston et al. 1994). For RSA I use the unsmoothed data from preprocessing, because I probe granular patterns of activity, and I transform these data from functional to MNI standard space. I only include cerebral cortical and hippocampal voxels by defining a mask from the *Harvard-Oxford Subcortical Structural Atlas* in *fsleyes*. Within this mask, I define 100-voxel searchlights for each voxel. I then find average stimulus responses across a searchlight by fitting a GLM to the functional data of all voxels within the searchlight. This GLM models every one of the four task blocks (two blocks of task A, two blocks of task B) separately, which guarantees uncorrelated regressors across blocks to avoid the bias identified in Section 2.1.2. In each block I include onset regressors for each of the 8 stimuli, for the catch trials

with each of the 8 stimuli, and for button press responses. I add the motion outlier frames and cerebrospinal signal as motion regressors. SPM convolves regressors, apart from nuisance regressors, with a hemodynamic response function and fits the GLM to produce activation patterns for each stimulus. I then calculate an 'across-block' RDM for task A as the correlation distance between stimulus activation patterns in the two task A blocks, and average the upper and lower triangle of the RDM to make it symmetric. I do the same for task B. As discussed in Section 2.1.2, this across-block RDM avoids bias due to correlated regressors. I thus end up with two 8×8 data RDMs, one for task A and one for task B, in every voxel. As the final step of the within-subject RSA pipeline, I fit a GLM with the upper triangle of each hypothesis RDM (excluding object similarities in task B) as regressors to the data RDM upper triangle. That produces a map of regression coefficients for each similarity hypothesis in each voxel for task A and task B. I smooth these parameter maps with a 5 mm full width at half maximum kernel as a final preparation for group-level analysis.

Encoding

I implement one more type of analysis: a univariate encoding approach. This analysis is motivated by the fact that both the repetition suppression and the representational similarity analysis are well-suited for identifying voxels that represent *one* of the hypothesised similarities, but not so much for voxels that represent *any* of them. Finding the latter kind of voxels relies on all similarities at the same time, and may therefore be more robust in noisy data. After detecting regions with representations of any of the hypothesised similarities, I can investigate the organisation of the individual similarities within those regions. To localise these regions, I predict voxel responses to held-out stimuli from a set of stimulus features. If a voxel represents a stimulus by its hypothesised similarity features, then fitting how that voxel's response to the features of one set of stimuli should enable the prediction of its response to stimuli with different features.

To measure a voxel's average response to each stimulus, I construct a GLM of stimulus onset regressors, very much like the searchlight GLM in RSA. I run FEAT with the same set of regressors as in RSA to fit the GLM on smoothed pre-processed data in subject functional space. The resulting parameter estimates produce four 8-dimensional vectors, two for the task A blocks and two for the task B blocks, of stimulus responses for each voxel. These are the values I am going to predict based on stimulus features. So far I only defined hypothesised similarities between stimuli, which are like distances in their similarity space. Now I need stimulus features, which are more like coordinates in that similarity space. To obtain these, I calculate the eigendecomposition of the similarity matrices, and use the entries of the second and third eigenvectors as stimulus features. I only include abstract features to get a 6-dimensional feature vector in task A for each stimulus and a 4-dimensional feature vector in task B (which has no object features). Then, I take the neural responses to stimulus 1 to 7 in the first block of task A, and fit a GLM with the corresponding features as regressors. The resulting parameter estimates are feature weights that can be used to predict the neural response to features of the held-out stimulus 8. I repeat the same fit and prediction, holding out each stimulus once, and end up with an 8-dimensional vector of predicted responses. Finally, I regress these predicted responses from the first block on the measured responses from the second block. That produces a coefficient that expresses how well I can predict a voxel's response in a different block to a stimulus with features I did not fit on. I then predict the other way around, fitting on the second block and predicting the first block, and repeat the process for task B. To narrow down which features enable accurate predictions, I repeat the analysis while only including the 2-dimensional feature vector for each of the similarities.

Group statistics

The final result of the three analyses described above is a set of brain maps consisting of regression coefficients in every voxel for each subject. To assess

the reliability of these results across the population of participants, I then calculate statistical maps on the group level using FEAT (Woolrich et al. 2004). After making sure that all subject-level maps are in the common MNI standard space, using the mappings calculated in preprocessing, I fit a group-level GLM. The group-level GLM contains a constant mean effect regressor, and a number of regressors modelling the subject-specific variations around the mean: the order of scan tasks, the performance on scan task catch trials and the number of time-out on catch trials, and the performance on the final training day task.

However, simply fitting a group-level GLM does not control the family-wise error rate. To account for the fact that, by fitting models in every single voxel in the brain, I have a huge multiple comparisons problem, I carry out max-statistic permutation tests (Nichols and Holmes 2001) in PALM (Winkler et al. 2014). I randomly multiply the subject-level coefficients of each participant by -1 or 1 before fitting the same group-level GLM. Under the null hypothesis that subject-level coefficients are symmetric around zero, this sign-flipping permutation would not affect the group-level mean. I repeat this procedure 10,000 times to build up a null distribution of permutations. Any test statistic that I calculate on the original, unpermuted data, I also calculate for each of the 10,000 permutations. Finding the fraction of permutations in which the test statistic in a particular voxel exceeds that of the same voxel's unpermuted statistic provides an *uncorrected* p-value for that voxel. By comparing an unpermuted voxel's test statistic to the maximum statistic across the whole brain (or actually: across voxels within the hippocampal-cortical mask as used in RSA) in each permutation, I obtain its *family-wise error corrected* p-value. The same process can be applied to cluster-based analysis: I form clusters by thresholding coefficients, calculated the resulting cluster mass, and compare cluster masses in the unpermuted data to the maximum cluster mass in each of the permutations for a corrected cluster p-value.

Surface

The fMRI data that I collect is a series of three-dimensional volumes, measuring timeseries of brain activity in a regular three-dimensional grid of voxels. However, for studying representations in cerebral cortex, this three-dimensional coordinate system is not necessarily the most natural space to work in. Instead, the cortical surface can be considered as a folded two-dimensional sheet. I extend all analyses described in volumetric space above to the cortical surface. This has several advantages. First, functional regions are anatomically organised as patches on the cortical surface, following its two-dimensional topology (Tootell et al. 1995; Dale et al. 1999; Halgren et al. 1999). Neighbouring voxels, close in volumetric space, might be separated by a large distance on the surface, for example for two voxels on opposite sides of a sulcus. Localisation of functional regions is therefore expected to be more accurate on the cortical surface. Second, to compare effects across participants, the within-subject results need to be registered to a common standard space. Because functional cortical areas follow the surface topology, surface-based registration can lead to more accurate alignment of areas between participant (Fischl et al. 1999; Glasser et al. 2013). Finally, as analysis on the cortical surface only considers a small subset of all voxels measured in fMRI, the correction for multiple comparisions is less severe.

I use the FreeSurfer software (Fischl 2012) for surface-based analysis. FreeSurfer reconstructs the cortical surface from the structural volumetric image. In surface analysis I start from the unsmoothed preprocessed data in subject functional space, because I want the smoothing to reflect the surface topology, and I want to exploit the improved surface-based standard space registration. How I proceed from there is different for the univariate (repetition suppression and encoding) and multivariate (representational similarity) analyses. In the univariate case, I use FEAT to fit the same GLM in functional space as in the volumetric analysis. FreeSurfer then calculates the registration from functional to structural space to map the volumetric GLM results to the subject-specific cortical surface,

and from the subject-specific surface to the standard surface. I then apply smoothing with a 5 mm full width at half maximum Gaussian kernel on the surface. Any further analysis, like held-out stimulus response prediction in the encoding analysis or the group-level permutation analyses, is then identical to its volumetric variant – except for the fact that it applies to the values at each of the vertices of the standard surface maps. For the multivariate RSA, I now define searchlights in each participant's functional space, instead of in MNI standard space. Importantly, I use the RSA toolbox and FreeSurfer's reconstructed subject-specific surfaces to define the 100 nearest voxels included in a voxel's searchlight according to distances *on the surface*, instead of volumetrically. I then calculate the data RDMs in subject volumetric functional space for each searchlight, and fit the hypothesis RDMs. Using FreeSurfer's registration, I map the resulting parameter estimates from subject functional volumetric space to the standard surface. I then smooth on the surface, and do group analysis through permutation testing on standard surface vertex values, correcting for multiple comparisons in each hemisphere.

2.3 Results

This section presents the results of the experiment and analyses described in Section 2.2. After showing evidence of participants successfully learning the task and its underlying regularities, I will proceed to plot neuroimaging results. Only very few results survive family-wise error correction, and agreement between analysis approaches is rare. I will not be selective in presenting results that fit a narrative. Instead, I will report all neural outcomes as transparently as possible, plotting whole-brain maps across analyses. As a consequence, this section reads more like a list than a story, but hopefully it provides an overview of what the data do and do not support.

To give an impression of trends and distributions of responses across the brain, I will show uncorrected p-value maps, for both the volumetric and cortical surface analysis variants. These uncorrected p-values result from 10,000 sign

flip permutations and are plotted as negative base-ten logarithms, thresholded between 2 (red) and 4 (yellow), corresponding to $p = 1 \cdot 10^{-2}$ and $p = 1 \cdot 10^{-4}$ respectively. Because situation similarities can be measured in both task A and task B blocks, I will plot their resulting maps twice: once for the relevant block only (conceptual similarity in task A, sequential similarity in task B), and once for the contrast averaging across both blocks.

2.3.1 Behaviour

Over the three days of training, participants improved on all components of the training task: the dynamics prediction and situation classification and the following which-object and which-situation questions of task A, and the story prediction of task B (Figure 2.13a). They performed well above chance on each on the final testing day (one-tailed t-test, all $p < 0.001$).

I then analyse the participants' errors to get confusion matrices: when they made a mistake, which object or situation did they respond instead of the correct one? I compare the measured confusion matrices to the perceptual similarity matrices (Figure 2.13b). To compare confusion matrices to conceptual similarity matrices pooled across participants, I need to 'un-randomise' objects and situations, so that the abstract similarities can be aligned. On the first training day, the confusion matrix for situation classification in task A across participants was significantly correlated with perceptual similarity (Pearson's $r = 0.39$, one-tailed t-test $p = 0.0015$), but not conceptual similarity (Figure 2.13c). On the third day of training (Figure 2.13d), the confusion matrix in task A correlates with conceptual similarity in addition to perceptual similarity (Pearson's $r = 0.27$, one-tailed t-test $p = 0.02$). In other words, participants confused situations that share objects, even when the dynamics of these situations are different across participants. Moreover, the better participants are at the task, the stronger their confusion correlates with the conceptual similarities (Figure 2.13e; Pearson's $r = -0.26$, one-tailed t-test $p = 0.027$), and the weaker the confusion correlates with perceptual similarities (Figure 2.13f; Pearson's $r = 0.26$, one-tailed

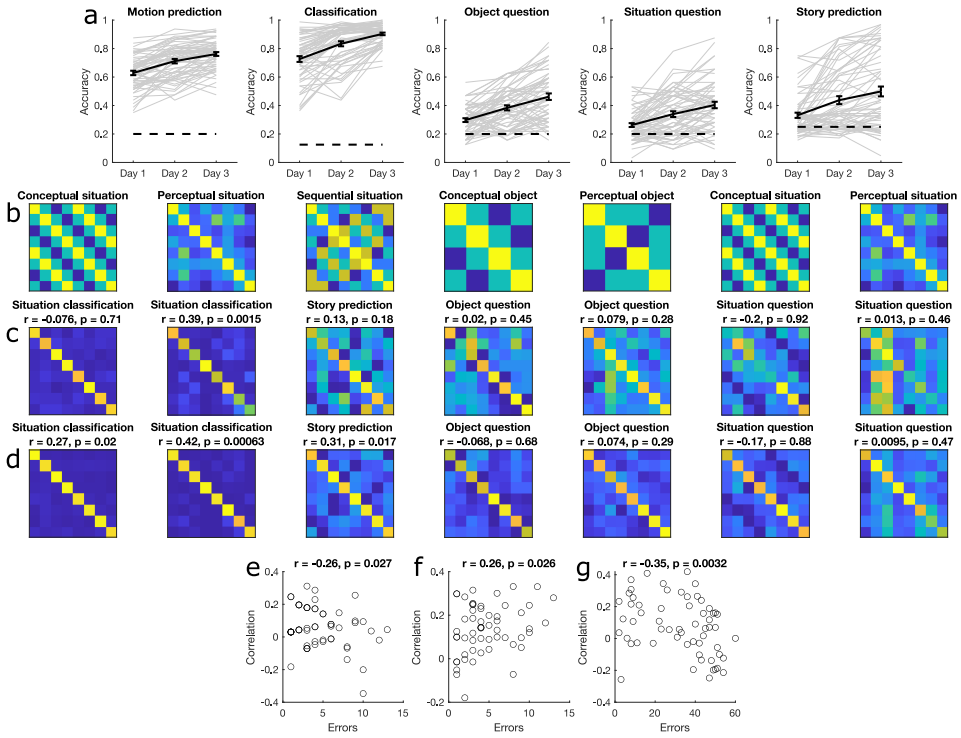


Figure 2.13: Behavioural results. a. Accuracy on each of the different tasks across training days, from left to right: Task A: motion prediction, situation classification, which-object question, which-situation question; Task B: story prediction. Each grey line shows one participant's accuracy, the black line plots mean $\pm$ s.e.m., and the dashed black line represents chance level accuracy. b. Hypothesised similarities between objects and situations for comparison to confusion matrices of responses in the different tasks. From left to right: conceptual then perceptual situation similarity for task A classification, sequential situation similarity for task B prediction, conceptual then perceptual object similarity for task A which-object question, conceptual then perceptual situation similarity for task A which-situation question. c. Corresponding confusion matrices pooled across participants on day 1. The perceptual situation similarity correlates with confusion in task A's classification task. d. Confusion matrices pooled across participants on day 3. Now the conceptual similarity correlates with task A classification confusion, and the sequential similarity with task B prediction confusion. e. The errors of better performing participants are more correlated with conceptual similarity in task A classification, f. and less with perceptual similarity. g. In task B prediction, better performing participants confuse more sequentially similar situations. A caveat for e-g: confusion correlations (on the y-axis) are not meaningful for very low numbers of errors (on the lower x-axis).

t-test $p = 0.026$). This suggests that participants rely on object occurrence regularities for situation judgements in task A, and better participants use this information more. Task B shows a similar pattern. On the first day, the confusion matrix for story prediction in task B was not correlated with sequential similarity (Figure 2.13c). On the third day, the confusion matrix pooled across participant

was significantly correlated with sequential similarity (Figure 2.13e; Pearson's $r = 0.31$, one-tailed t-test $p = 0.017$), and between participants this correlation was higher for participants with fewer errors (Figure 2.13g; Pearson's $r = -0.35$, one-tailed t-test $p = 0.0032$). The which-object and which-relation questions at the end of task A blocks did not show correlations between confusion and similarity matrices on any day.

2.3.2 Repetition suppression

Figure 2.14 plots uncorrected p-values for perceptual similarities measured by repetition suppression. The perceptual sanity check, modelling suppression for subsequent stimuli with similar pixel values, shows strong activation in early visual cortex (Figure 2.14a). There are no clear responses to the perceptual similarity for object shapes (Figure 2.14b) and situation dynamics (Figure 2.14c).

Repetition suppression for conceptual similarity is plotted in Figure 2.15. Conceptual similarity between objects has its strongest activation in right lingual gyrus, near the posterior end of cingulate cortex and hippocampus, in the volumetric analysis but less clearly so on the surface. Responses to conceptual situation similarity are also located in lingual gyrus but more posterior, and more bilateral in the contrast across task A and task B.

Figure 2.16 shows repetition suppression results for sequential similarity. There are some hints of activation in entorhinal cortex, and a bit stronger activation, particularly across both tasks in Figure 2.16b, in lateral orbitofronal cortex.

So far, the repetition suppression maps showed uncorrected p-values. In Figure 2.17, I plot all suppression results that survive cluster-based family-wise error correction. This correction is carried out separately for volumetric and cortical surface results, so it is possible that one is significant while the other is not. The cluster formation threshold is $p = 0.01$ and I plot clusters with corrected $p < 0.05$ on a negative base-10 log colour scale, from 1.3 (red) to 4 (yellow). I find a significant cluster for stimulus pixel similarity in early visual cortex in the volumetric analysis (corrected $p = 0.0016$) and in both

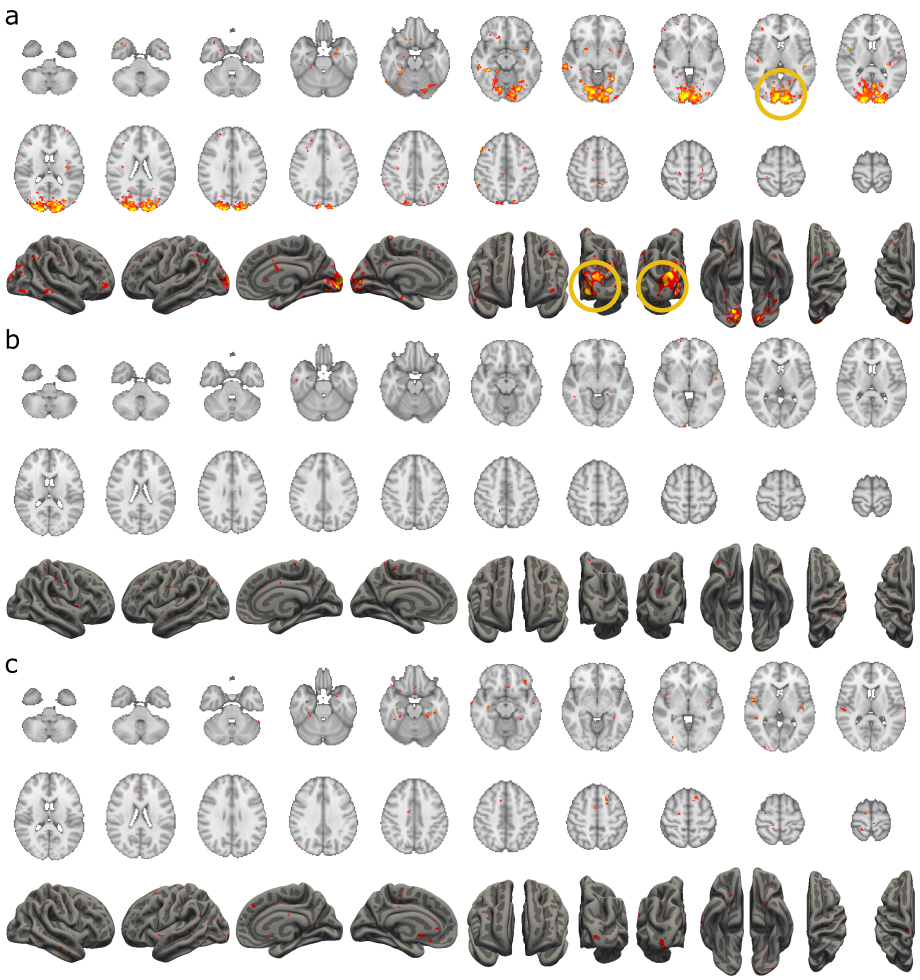


Figure 2.14: Perceptual similarity suppression. a. Negative base-ten logarithm maps of uncorrected p-values between 2 (red) and 4 (yellow) for suppression for the similarity in pixel values of subsequent stimuli, with strong activation in early visual cortex. b. Suppression for object shape similarity. c. Suppression for situation dynamics similarity.

hemispheres of the surface analysis (corrected $p < 0.001$ and $p = 0.027$ on the left, $p < 0.001$ on the right). Finally, the surface analysis reveals a significant cluster for conceptual situation similarity suppression in the left hemisphere's lingual gyrus (corrected $p = 0.043$).

2.3.3 Representational similarity

The responses to perceptual similarity in representational similarity analysis are plotted in Figure 2.18. Perceptual object similarity in RSA (Figure 2.18a) shows responses in lateral occipital cortex, inferotemporal cortex, and – particularly in

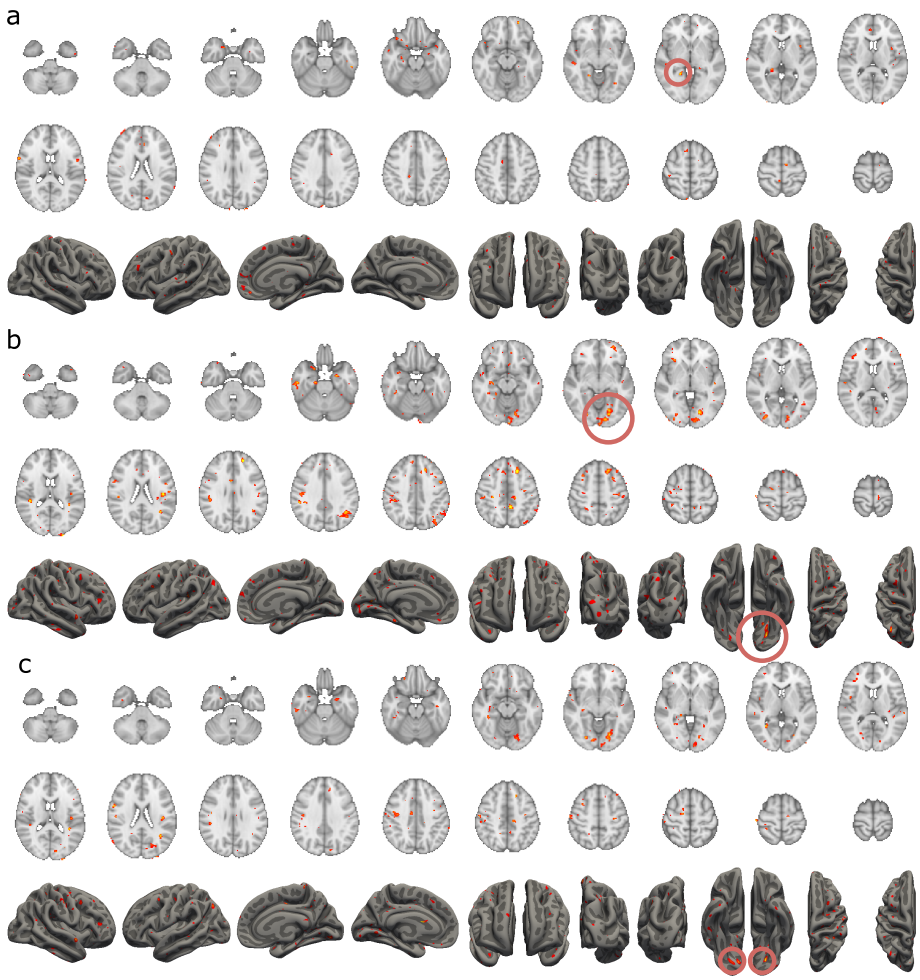


Figure 2.15: Conceptual similarity suppression. a. Negative base-ten logarithm maps of uncorrected p-values between 2 (red) and 4 (yellow) of suppression for conceptual object similarity, with peak activation in lingual gyrus. b. Suppression for conceptual situation similarity in task A, with indicated responses in lingual gyrus. c. Suppression for conceptual situation similarity averaged across blocks, with bilateral posterior lingual gyrus activation.

volumetric analysis – strong bilateral activation in inferior frontal sulcus. Perceptual situation similarity in Figure 2.18b shows strong responses in RSA in middle temporal visual area/V5, and also hints of activation in superior temporal sulcus.

Figure 2.19 plots RSA responses for conceptual similarity. There are responses in temporal pole and entorhinal cortex for conceptual object similarity (Figure 2.19a), but only in the volumetric analysis, not on the surface. Conceptual situation similarity (Figure 2.19b) shows RSA responses in perirhinal cortex, just lateral from the object entorhinal activation, and in parahippocampal gyrus.

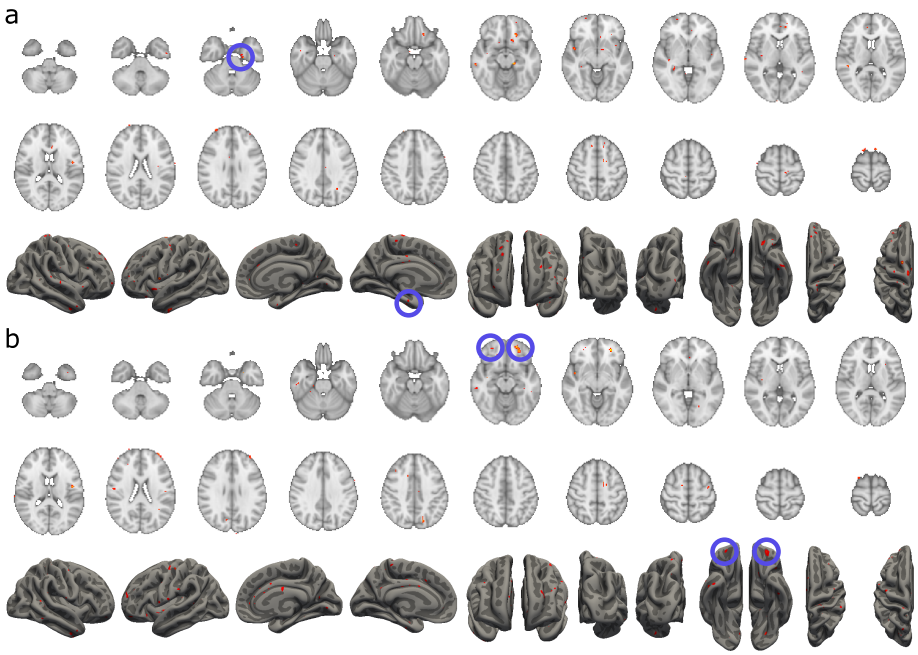


Figure 2.16: Sequential similarity suppression. a. Negative base-ten logarithm maps of uncorrected p-values between 2 (red) and 4 (yellow) of suppression for sequential situation similarity in task B, with indicated entorhinal cortex activation. b. Suppression for sequential situation similarity across blocks, with responses in lateral orbitofrontal cortex.

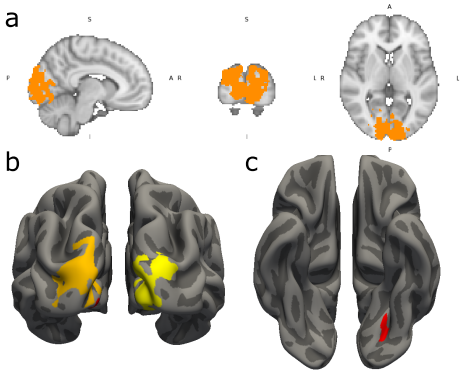


Figure 2.17: Family-wise error corrected suppression. a. Negative base-ten logarithm maps of family-wise error corrected cluster p-values between 1.3 (red) and 4 (yellow). The volumetric analysis shows a cluster with corrected $p = 0.0016$ in early visual cortex for pixel similarity suppression. b. Pixel similarity suppression also produces family-wise error corrected clusters in visual cortex on the left (corrected $p < 0.001$ and $p = 0.027$) and right ($p < 0.001$) hemisphere in surface analysis. c. Conceptual situation similarity cluster (corrected $p = 0.043$) in lingual gyrus of left hemisphere in surface analysis.

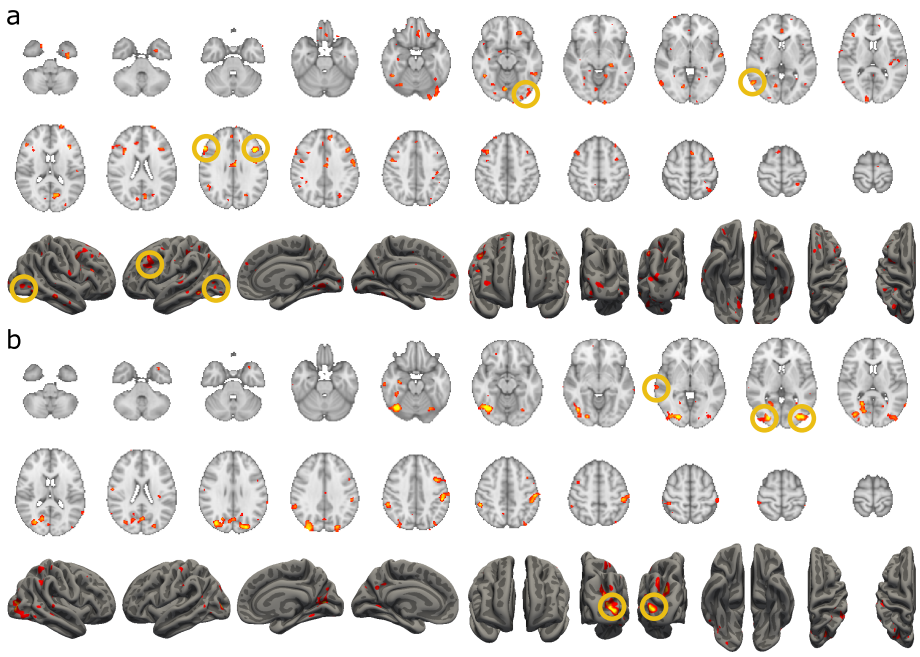


Figure 2.18: Perceptual similarity RSA. a. Negative base-ten logarithm maps of uncorrected p-values between 2 (red) and 4 (yellow) of RSA for perceptual object similarity, with indicated responses in lateral occipital cortex, inferotemporal cortex, and inferior frontal sulcus. b. RSA for perceptual situation similarity, with strong activation in V5 and smaller responses in superior temporal sulcus.

Additionally, there is a temporal occipital fusiform cortex response that also appears on the surface. But these results are less clear when considering the average across task A and task B blocks, as shown in Figure 2.19c.

Sequential situation similarity, shown in Figure 2.20, does not produce clear responses in RSA. In task B blocks, Figure 2.20a shows an RSA response that appears to be orbitofrontal in volumetric analysis, but in insular cortex on the surface. Figure 2.20b shows weak hints of parahippocampal responses and an unexpected early visual activation across scanner blocks.

The only similarity that survives family-wise error correction in RSA is the perceptual situation similarity. I plot the resulting clusters, with cluster formation threshold $p = 0.01$, by their negative base-ten log corrected p-value from 1.3 (red) to 4 (yellow) in Figure 2.21. The volumetric analysis in Figure 2.21 shows a single cluster in the right hemisphere with corrected $p = 0.0058$ that stretches across V4, V5, and both the superior and inferior parts of lateral occipital cortex.

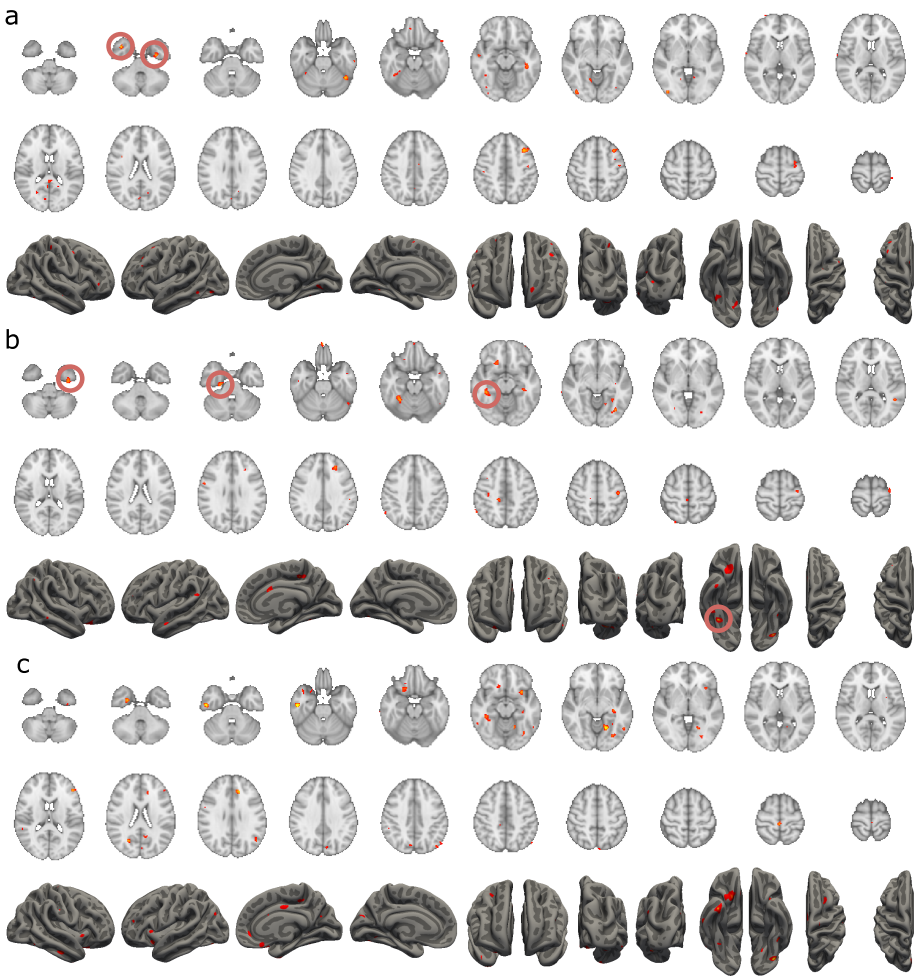


Figure 2.19: Conceptual similarity RSA. a. Negative base-ten logarithm maps of uncorrected p-values between 2 (red) and 4 (yellow) of RSA for conceptual object similarity, with activation in temporal pole and entorhinal cortex in the volumetric analysis. b. RSA for conceptual situation similarity in task A, with indicated responses in perirhinal cortex, parahippocampal gyrus, and temporal occipital fusiform cortex. c. RSA for conceptual similarity across task A and task B blocks.

There is also a more medial part of the cluster in pericalcarine cortex. In the surface analysis, this cluster is split up in a lateral occipital cluster with corrected $p = 0.012$ and a medial pericalcarine cluster with corrected $p = 0.018$, with only family-wise error corrected results in the right hemisphere.

2.3.4 Encoding

In the encoding analysis, I fit neural responses to features for a set of stimuli, and then predict the response to a held-out stimulus from its features across

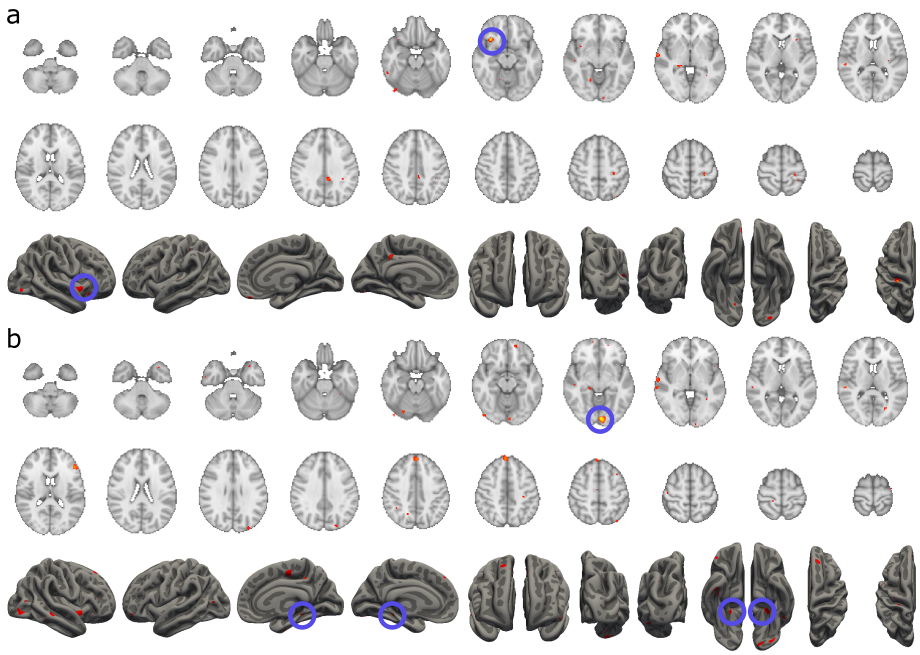


Figure 2.20: Sequential similarity RSA. a. Negative base-ten logarithm maps of uncorrected p-values between 2 (red) and 4 (yellow) of RSA for sequential situation similarity in task B blocks. The indicated response appears orbitofrontal in the volumetric analysis, but in insular cortex on the surface. b. RSA for sequential situation similarity across blocks, with unexpected activation in early visual cortex, and weak hints of responses in parahippocampal gyrus.

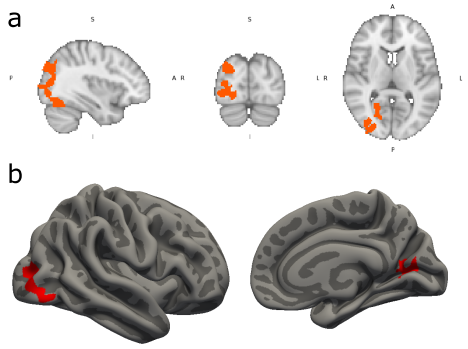


Figure 2.21: Family-wise error corrected RSA. a. Negative base-ten logarithm maps of family-wise error corrected cluster p-values between 1.3 (red) and 4 (yellow) for perceptual situation similarity. The volumetric analysis shows a cluster with corrected $p = 0.0058$ that includes parts of superior and inferior lateral occipital cortex, and a medial region in pericalcarine cortex. b. In cortical surface analysis, I find two separate clusters in the right hemisphere: one in lateral occipital cortex with corrected $p = 0.012$ and one in pericalcarine cortex with corrected $p = 0.018$. There are no family-wise error corrected clusters in the left hemisphere.

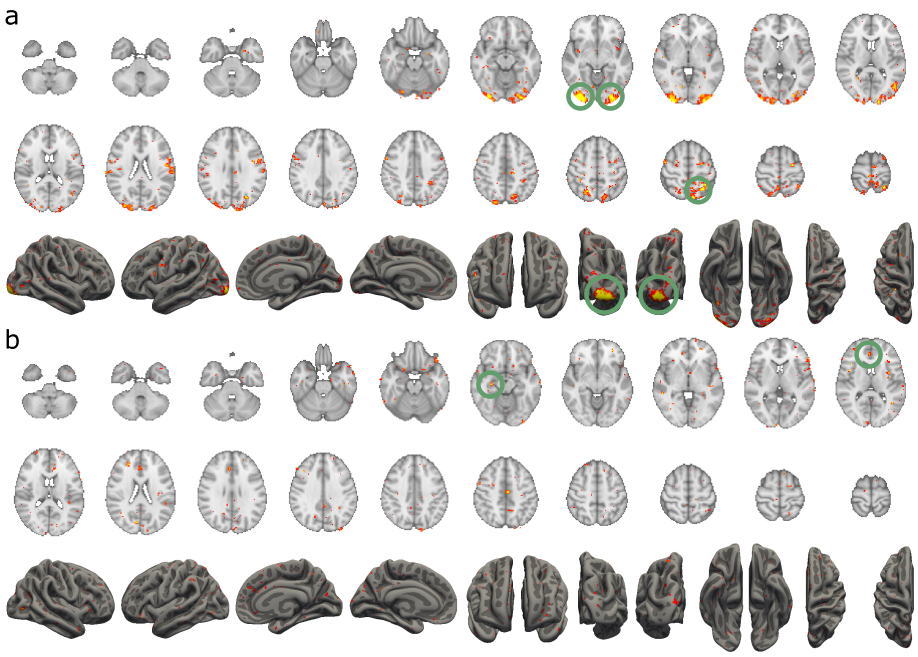


Figure 2.22: Encoding with all features. a. Negative base-ten logarithm maps of uncorrected p-values between 2 (red) and 4 (yellow) of encoding prediction fit in task A based on all stimulus features, with strong activations in early visual cortex. b. Encoding prediction fit in task B based on all stimulus features, with hints of responses in hippocampus and medial prefrontal cortex.

blocks. Figure 2.22 plots where predicted and measured responses match, across task A blocks (Figure 2.22a) and across task B blocks (Figure 2.22b). The task A results show strong activations in early visual areas, and particularly in volumetric analyses these extend up along superior lateral occipital cortex into the parietal lobe up to the central sulcus. These strong visual responses are absent in task B, where the volumetric analysis shows hints of activation in hippocampus and medial prefrontal cortex.

When I predict responses based on only a subset of features, corresponding to one similarity only, I find the results shown in Figure 2.23. Figure 2.23a plots voxels and vertices where only conceptual object similarity features predict activity across task A blocks, Figure 2.23b plots the same for only conceptual situation similarity features across task A blocks, and Figure 2.23c for only sequential situation similarity features across task B blocks. There are hints of bilateral temporal pole responses for conceptual object similarity features in

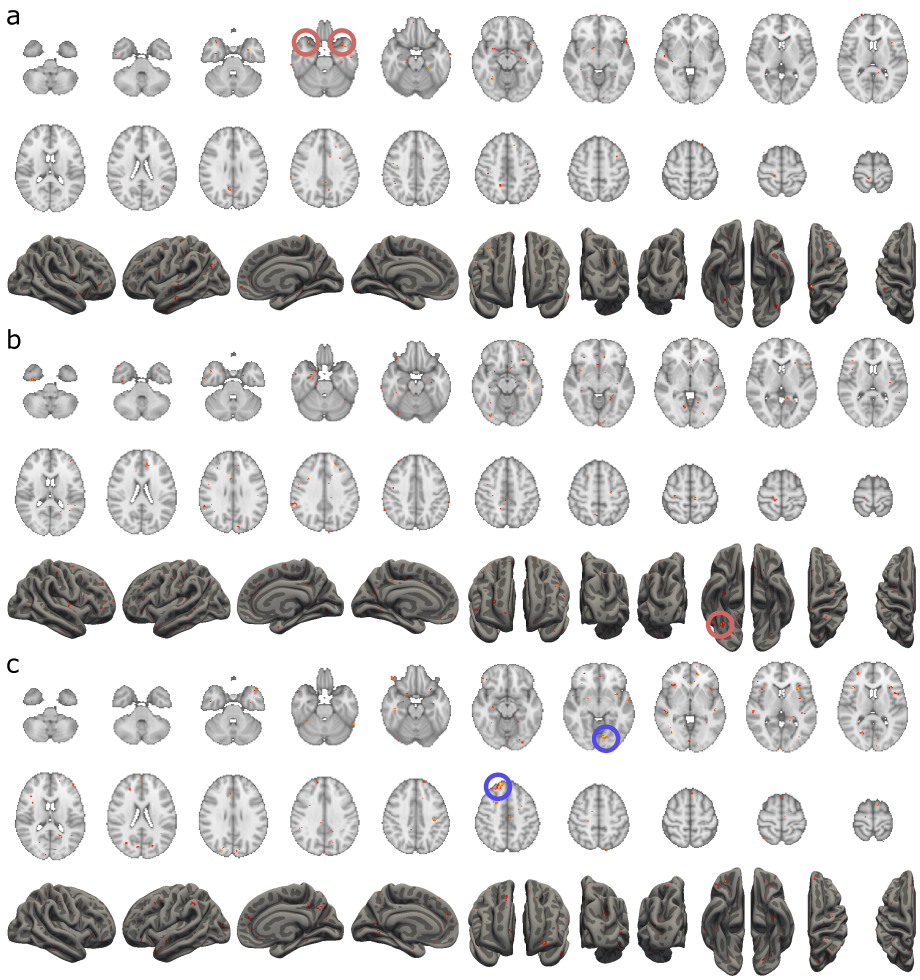


Figure 2.23: Separate component encoding. a. Negative base-ten logarithm maps of uncorrected p-values between 2 (red) and 4 (yellow) of encoding prediction fit in task A based on conceptual object features only, with indicated bilateral temporal pole responses. b. Encoding prediction fit in task A based on conceptual situation features only, with activity in temporal occipital fusiform cortex. c. Encoding prediction fit in task B based on sequential situation features only, with responses shown in lingual gyrus and frontal pole.

Figure 2.23a, mostly so in volumetric analysis. For conceptual situation similarity, Figure 2.23b shows activity in temporal occipital fusiform cortex. There is lingual gyrus as well as frontal pole activity for sequential situation similarity in Figure 2.23c.

Figure 2.24 plots the clusters in the encoding analysis that are significant after family-wise error correction. In the volume, there are three significant clusters for prediction across task A blocks based on all similarity features (Figure 2.24a). Two are located in early visual cortex on either hemisphere, with

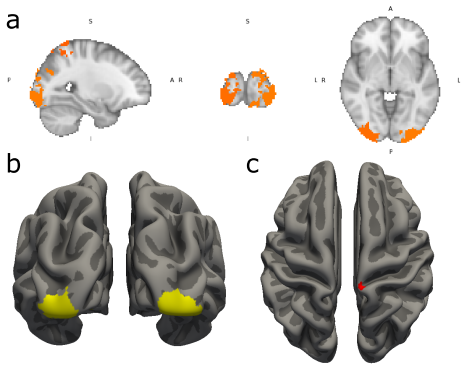


Figure 2.24: Family-wise error corrected encoding. a. Negative base-ten logarithm maps of family-wise error corrected cluster p-values between 1.3 (red) and 4 (yellow) for encoding prediction fit in task A based on all features. The volumetric analysis shows clusters in early visual cortex in both hemispheres, with corrected $p = 0.0024$ on the left and $p = 0.0035$ on the right, and a more dorsal cluster with corrected $p = 0.005$. b. In cortical surface analysis, I find the same early visual clusters for encoding prediction fit in task A based on all features: corrected $p < 0.001$ in both hemispheres. c. Encoding prediction fit in task A for conceptual object features only, with a corrected $p = 0.039$ right hemisphere central sulcus cluster.

corrected $p = 0.0024$ on the left and $p = 0.0035$ on the right, and the third is a more dorsal cluster that stretches along superior lateral occipital cortex into superior parietal cortex, with corrected $p = 0.005$. I find the same visual clusters for prediction using all features across task A blocks on the surface, with corrected $p < 0.001$ in both hemispheres (Figure 2.24b). Finally, there is a significant cluster for prediction using conceptual object similarity features in the central sulcus of the right hemisphere in the surface analysis with corrected $p = 0.039$ (Figure 2.24c).

2.4 Discussion

The brain represents the world at different levels of abstraction, with representations that range from sensory to highly abstract. I have conducted an neuroimaging experiment that aims to identify this hierarchy of abstraction. Over three days, participants learn about procedurally generated video stimuli and their regularities. These stimuli allow me to define precise similarity hypotheses for which I examine the BOLD signal measured in fMRI on the third

day. I find responses to *perceptual* similarities between object shapes and situation dynamics, *conceptual* similarities between objects that share situations and situations that share objects, and *sequential* similarity between situations that link together in stories.

Across the results presented in Section 2.3, the perceptual ones are most convincing. The perceptual similarities, particularly the pixel similarity and situation dynamics situation, reliably activate visual cortex and neighbouring areas. The strong response for pixel similarity in Figure 2.17a,b serves as an important sanity check, implying that basic stimulus timings and similarity modelling has worked. Pixel similarity mostly results from whether the grey floor square is aligned between stimuli, which depends on the camera perspective of the three-dimensional rendering. The camera perspective is sampled randomly for every single video stimulus, independent of objects and situations and different for each participant. Therefore when timing errors would misalign the stimulus presentation and modelled pixel similarity, the pixel similarity regressor would be useless. The representation of situation dynamics similarity found in Figure 2.21 provides evidence that I can measure brain responses to not just the static background pixels, but also to dynamic movies, and that the representational similarity analysis is set up correctly. Representing similarity of dynamics already requires some degree of abstraction: the actual object trajectories are different for each repetition of for example the 'follow' situation, and the random camera perspective creates further distortions in the trajectories of pixels on the screen.

However, it is harder to draw conclusions about any of the learned abstract representations. Overall, there are hints of responses along the temporal lobe, more posterior around lingual, fusiform, and parahippocampal cortex (Figure 2.15a,b, Figure 2.19b, Figure 2.23b,c), and more anterior around entorhinal and perirhinal cortex and temporal pole (Figure 2.16a , Figure 2.19a, Figure 2.19b). When prefrontal cortex is activated, the responses are usually located around orbitofrontal cortex (Figure 2.16b, Figure 2.19c). In general, these results are not consistent between analyses, and often not even across blocks of the same analysis. The

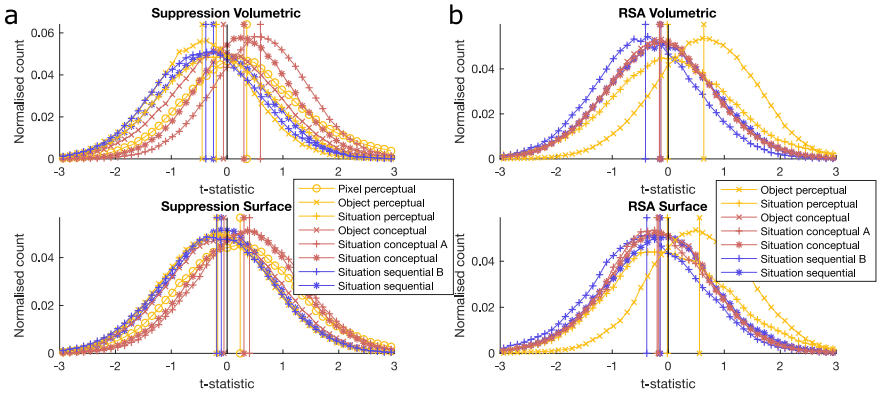


Figure 2.25: Whole-brain statistics histograms. a. Histogram of t-statistics across the whole brain in the unpermuted group map of the repetition suppression analysis, with the bias of each distribution given by the mean indicated as a vertical line, in the volume (top) and on the surface (bottom). b. Whole-brain group-level histogram with biases for representational similarity analysis in the volume (top) and on the surface (bottom).

fact that I find a family-wise error corrected cluster for situation conceptual similarity in repetition suppression (Figure 2.17c) seems promising, but should be interpreted with caution. My reason for suspicion is the whole-brain group-level t-statistic histogram for this result. I have plotted the whole-brain histograms for all RSA and suppression results of Section 2.3 in Figure 2.25. The histogram for conceptual similarity in suppression on the surface (Figure 2.25a, bottom) has a considerable positive bias.

These biases in the histogram of group-level whole-brain statistical maps may cause unreliable permutation test results. An artefact-induced bias invalidates the sign-flipping permutation tests. If a particular result has on average positive parameter estimates across participants, sign-flipping the values for some participants will likely result in statistical maps with smaller values than the original unpermuted map. Positive biases make it easier to survive family-wise error corrections, and negative biases make surviving them harder. Indeed, the group-level biases of Figure 2.25 are reflected in the maps plotted in Section 2.3. For example, there are many strong responses in Figure 2.18a and very few in Figure 2.14b. What is the source of these biases? In principle the brain can respond positively or negatively on average, but that is unlikely here.

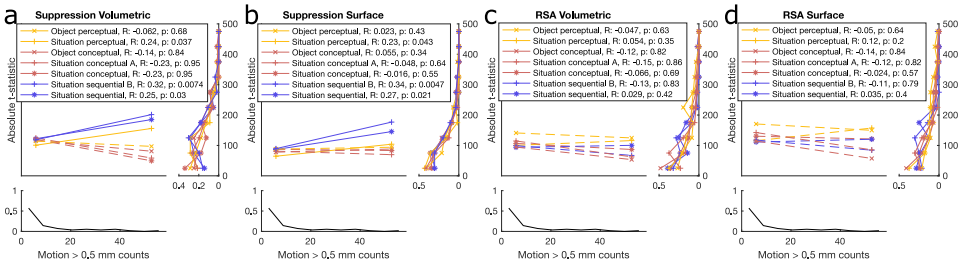


Figure 2.26: Bias-motion correlation. a. Correlation between the amount of motion of a participant and their absolute whole-brain t-statistic for each result in volumetric repetition suppression. The amount of motion, on the x-axis, is quantified as the number of fMRI volumes with $>$ 0.5 mm instantaneous motion, which is the same measure used to exclude subjects for excessive motion. The absolute whole-brain participant t-statistic, on the y-axis, measures how far activations are from zero-centred across the brain. I plot the histogram of motion and t-statistics across subjects next to the axes, and a linear regression with a solid line for significant correlations and a dashed line otherwise. b. Correlation between participant motion and absolute whole-brain t-statistic for each result in surface repetition suppression. c. Correlation between participant motion and absolute whole-brain t-statistic for each result in volumetric representational similarity analysis. d. Correlation between participant motion and absolute whole-brain t-statistic for each result in surface representational similarity analysis.

Motion artefacts may cause biases, because they induce large BOLD signal changes – although it is improbable these would have the same effect for each participant. Many steps of my preprocessing pipeline are aimed at reducing motion artefacts, including ICA denoising, regressing out motion timeseries and their derivatives, and the inclusion of cerebro-spinal fluid regressors and motion outlier regressors. If motion is really the cause of the biases, then participants that moved a lot may have stronger biases. I plot correlations between participant motion and the whole-brain absolute t-statistic for all analyses in Figure 2.26. Except for situation perceptual similarity in suppression, I do not find bias-motion correlations across subjects.

RSA and repetition suppression are orthogonal methods, operating on different properties of the BOLD signal. Repetition suppression is univariate, relying on single-voxel tuning, while multivariate RSA probes microstructures of selectivity across voxels in a searchlight. Nevertheless, they measure the same similarities in my experiment. I would therefore expect them to show overlapping results, but do not find that reliably in Section 2.3. To get an idea of the

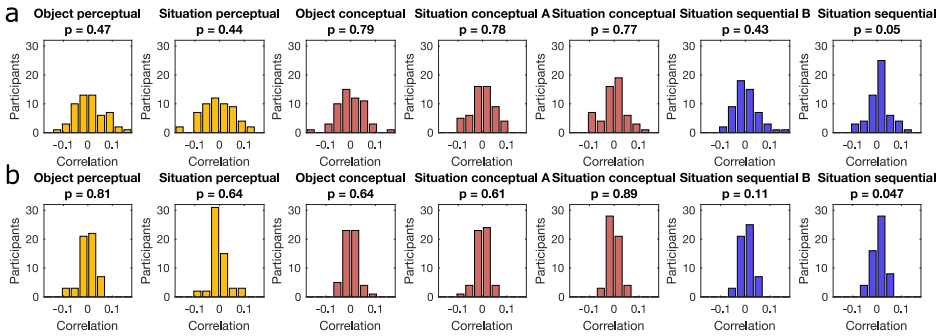


Figure 2.27: RSA-suppression correlation. a. Histogram of Fisher-transformed whole-brain correlations between repetition suppression and representational similarity analyses across participants, in volumetric analysis, with p-value for significant positive correlation across partipants. b. Correlation between repetition suppression and representational similarity analyses across participants, in surface analysis.

correspondence between RSA and repetition suppression, I calculate the whole-brain correlation between RSA and suppression for each analysis in each participant, and plot the distribution of correlations across participant in Figure 2.27. I do not find strong correspondence between RSA and suppression. This does not mean it does not exist: if similarities are only represented in small fractions of the brain, whole-brain correlations are not necessarily expected. Repetition suppression and multivariate pattern methods have been found to correlate in early visual areas (Sapountzis et al. 2010) and lateral occipital cortex (Drucker and Aguirre 2009), possibly modulated by attention (Moore et al. 2013).

The encoding analysis meant to identify regions that respond to *any* kind of abstract similarity. Bao et al. 2020 find repeated 'islands' of shape representation along inferotemporal cortex, with the same anatomical organisation of shapes (e.g. from body to face to stubby) within each island. If similarly organised representations exist for the abstract similarities here, the encoding analysis could identify robust islands across participants, and then for each participant analyse the order of representations within islands – that way, I do not require representations at identical locations within islands across participants. However, the encoding results of Section 2.3 highlight two problems with the method. First, it is prone to overfit responses to stimulus features, as I have six (scanner task A) or four (scanner task B, without objects) features for only eight

stimuli. Overfitting leads to inaccurate held-out stimulus response predictions, but may be reduced by some form of regularisation. I suspect that the strong visual encoding responses in task A (Figure 2.22a and Figure 2.24a,b) occur for the same reason: If the dimensionality of the feature space is close to the number of stimuli, response prediction will be good in regions that encode stimulus identity consistently across blocks. But a different set of features may work equally well. To prove that only the hypothesis stimulus embedding allows for accurate prediction, I could compare the prediction accuracy to predictions based on feature vectors shuffled across stimuli. Second, the resulting maps are particularly spotty because they plot parameter estimates of a regression where both the data (measured responses) and regressor (predicted responses) are noisy. Smoothness is important if I want to use the resulting maps as regions of interest for other analyses.

So what are possible reasons for the lack of robust neural results in this study? One issue that affects all analysis methods is that many of the regions where responses are expected (and sometimes found) are notoriously hard to image in MRI. Motion, physiological and susceptibility artefacts, and signal dropout are particularly prominent in medial temporal and prefrontal regions. Especially in medial temporal lobe, registration from subject functional space to standard space is difficult. Cortical surface analysis can improve registration, and for further improvements multimodal surface matching may be beneficial (Robinson et al. 2014, 2018). Multimodal surface matching uses not only anatomical features to align participants to a standard surface, but matches other measures too. For example, resting-state networks provide common functional features across participants. I find these networks by independent component analysis on the concatenated functional data across participants, then dual regression to obtain participant-level functional networks from the resulting group components. Then after registration to the standard surface, I use multimodal surface matching to calculate an additional transformation that aligns the functional networks on the surface, and apply that alignment transformation to

the participant-level parameter estimates. The resulting group-level responses are sometimes shifted, sometimes weaker, sometimes stronger but not more convincing compared to those presented in Section 2.3.

To find representations of abstract similarities, it is crucial that participants have learned the corresponding stimulus regularities well. Figure 2.13a shows that individual participants respond above chance when asked about these regularities, while Figure 2.13d shows that when pooling across participants, situations with abstract similarity are confused more. However, I do not have a clear behavioural readout of whether an individual participant has really embedded the objects and relations in their abstract similarity spaces. The within-participant confusion correlation on the y-axis of Figure 2.13e-g is not meaningful for a participant who makes very few errors. If each participant would have a clear signature in their behaviour of organising stimuli by their abstract regularities, it would be unlikely not to find neural correlates. But without evidence of this organisation of objects and situations in individual participants, I cannot be certain that the training was sufficient to consistently induce the hypothesised similarities.

Registration inaccuracy and learning variability increase group-level variance. This may be counteracted by increasing the number of stimulus presentations during the scan task to improve parameter estimates, or by increasing the sample size. Power calculations for fMRI are difficult, especially in a new paradigm, because effect sizes are hard to estimate – what would the induced percentages of BOLD increases and decreases be? – and their consequences for final outcome measures are not obvious – how do these BOLD responses result in p-values after, in RSA for example, calculating stimulus representational dissimilarity matrices, fitting hypotheses RDMs, and group-level permutation testing? The number of stimulus repeats in the scan task, four blocks with each pair of subsequent stimuli repeated three times, or each individual stimulus repeated 21 times, was therefore mostly based on similar previous studies. For example, Garvert et al. 2017 measure repetition suppression in hippocampus for seven

stimuli, repeating each pair ten times in three blocks, and Baram et al. 2021 apply RSA in entorhinal cortex for eight stimuli repeating each stimulus ten times in eight blocks. It is possible that more repeats are required to improve parameter estimates, but longer scans do put a strain on participants' attention span. I use an eye tracking camera to monitor participants' eyes in the scanner and many start losing focus near the end of the scan. The number of participants is also based on previous studies, and I increased the sample size from 34 in the initial round of data collection to the final 60 scans (before any exclusion) to obtain a larger-than-usual sample size (Szucs and Ioannidis 2020).

In summary, this experiment did not reliably demonstrate abstract representations of objects and situations. Sensory properties of stimuli evoked robust responses in visual areas, but I found only weak hints of representation of abstract similarities along temporal lobe and prefrontal cortex. The conclusion is that I did not find evidence of a hierarchy of abstraction, as an anatomically organised gradient with progressively more abstract representations. MRI data are extremely high-dimensional; other modalities in neuroscience are rapidly approaching the same regime (Steinmetz et al. 2019)[†]. In my opinion, effective use of such data requires tightly controlled experiments with precise quantitative predictions, answering yes-or-no questions. Without these, there is a risk of fitting hypotheses to data, instead of the other way around, and finding nonsense correlations – like mouse neurons predicting cryptocurrency price (Meijer 2021). However, clean yes-or-no questions also come with a risk: that the answer is no. Of course, absence of evidence is not evidence of absence. In fact, I would be surprised if the brain really does not represent the abstract similarities from statistical regularities as emulated by the stimuli in this experiment. But I have not been able to identify the hypothesised representations. I still think there is value in reporting this null result. Unlike null results in randomised controlled trials, I do not think there are obvious implications. The task is too specific and the scope is too limited for drawing strong conclusions. But I do hope that

[†]poem in Appendix

the questions asked here, and the approach for answering them, can serve as inspiration for the questions, hypotheses, and designs of future experiments on the hierarchy of abstraction.

3
Planning through composition

When James Cook was exploring Tahiti in 1769, he took a local guide named Tupaia aboard the HMS Endeavour. Tupaia was a priest, scientist, and master navigator. As part of his education, he had memorised knowledge of the islands in the region, like their size and their chiefs, but importantly also the bearing of each island, and the time to get there. Together with Cook and his officers, Tupaia started to work on a map of the Pacific Ocean. But the resulting map looked rubbish. Islands in the north appeared in the south, far away islands were located right next to each other... "It is therefore very probable that the vanity of appearing more intelligent than he really was, had prompted him to produce this fancied chart of the South Sea, and perhaps to invent many of the names of islands in it", wrote Georg Forster, one of Cook's crew, in 1777. The map remained a mystery, and evidence of the limited navigation skills of the Polynesians for some, until in 2019 two German scientists found a dramatically different interpretation of the map.

Lars Eckstein and Anja Schwartz realised that Tupaia, having learned about Cook's Cartesian maps, invented a entirely new cartographic model. He placed e-avatea, Tahitian for 'noon', at the center of the chart to mark bearing to the north. To navigate from one island to another, the navigator pictures themselves on the starting island and draws a line towards the center and towards the



Adapted from Eckstein and Schwarz 2019

goal island. The angle between the lines produces the compass direction for travel. These angles turn out to be spectacularly accurate: the bearings from the most remote islands on the map, at 4000 kilometers, read 335° – and 330° on our modern maps. Tupaia had built a system of island positions relative to the traveller, to translate the Polynesian way of reckoning and wayfinding for Cook's map and compass.

In this Chapter I will present a model of navigation in the brain. I will show how to find goals in new environments without having to relearn good routes. I propose that the brain achieves that by using maps like Tupaia's for making decisions, instead of Cook's maps. It keeps track of relative positions of objects, adding them in one by one, so that it can immediately read out the right directions. And to fill in the map of relative positions in far-away places, the brain imagines going there.

Reinforcement learning (RL) is concerned with the problem of choosing actions that maximise long-term reward (Sutton and Barto 2018). The goal is learning a *policy* $\pi(a|s)$ that provides the appropriate *action* a given the current *state* of the world s. The policy $\pi(a|s)$ can thus be written as a mapping $f(\vec{s}) = \vec{a}$ from state representation $\vec{s}$ to a probability distribution $\vec{a}$ over available actions. Learning this mapping is hard. It involves assigning credit to actions in states

that may be rewarding only in the distant future (Minsky 1961). Moreover, changes in environment or reward may completely invalidate learned mappings. Algorithmic solutions to the credit assignment problem require extensive learning experience (Watkins and Dayan 1992; Mnih et al. 2015) or sophisticated planning at decision time (Kocsis and Szepesvári 2006; Silver et al. 2016). However, animals can make accurate decisions quickly even in novel situations (Tolman 1948; Harlow 1949). Here I demonstrate how such rapid optimal behaviour can be constructed in new environments without any additional learning. Instead of relearning f each time, I compose state representation $\vec{s}$ out of general building blocks, that come 'pre-credit-assigned': a single, pre-learned state-action mapping directly applies to all building blocks.

Fast or flexible decisions, but not both

Temporal difference (TD) learning (Sutton 1988; Montague et al. 1996) is an RL algorithm with particular relevance to neuroscience due to its correspondence with midbrain dopamine signalling (Schultz et al. 1997). It assigns credit to actions by comparing their expected and observed long-term reward (or *value*), then updating these expectations according to the reward prediction error. Long-range credit assignment takes many of such updates, and would therefore require experiencing actions many times. To alleviate this experience requirement, algorithms like DYNA (Sutton 1991) carry out value updates offline. After value estimates have converged, f can simply select the highest value action in a state.

However, TD-learning is slow to adapt to environment changes. For example, a change of reward location requires a complete reconfiguration of values over many TD updates. To account for flexible behaviours that immediately respond to such changes, the existence of a *cognitive map* was proposed (Tolman 1948): a mental representation of states linked by their transitions (Behrens et al. 2018). Equipped with a cognitive map, f is obtained through simulation or *planning* of action sequences, and selecting the action that leads to the

most favourable long-term outcome (Kocsis and Szepesvári 2006; Botvinick and Toussaint 2012). Planning on cognitive maps can flexibly take environment changes into account, but is computationally costly and error-prone (Daw et al. 2005). Extensive mental simulation is not feasible when a quick decision is required, for example during escape responses (Shamash et al. 2020).

State representations for generalisation

The robust and rapid decision making from TD-learned values can be combined with the flexibility provided by planning through *generalisation*. A clever state representation $\vec{s}$ allows for the transfer of knowledge from one situation to another, instead of relearning f from scratch (Ponsen et al. 2010; Li et al. 2016). An example is the *successor representation* (SR) (Dayan 1993). Instead of long-term expected reward, the SR expresses long-term expected occupation of future states. Credit assignment becomes a matter of multiplying SR and immediate reward, which easily accounts for new situations with different reward locations. The SR thus generalises across reward locations in the same environment, but is limited by its policy dependence and it needs relearning in new environments. Hippocampal place cells have been proposed to represent each state's SR, and entorhinal grid cells the SR eigenvectors across states (Stachenfeld et al. 2017).

Here, I exploit the power of generalisation to construct state representations $\vec{s}$ for which a single f applies across environments, whatever shape, size, and configuration of rewards and obstacles they have. This state representation $\vec{s}$ is composed from building blocks that flexibly recombine into arbitrary environment layouts. These building blocks have two crucial properties: they path-integrate, so once a representation is initialised in one place it is accessible in any other place simply by navigating there, and they imply actions, so given a representation the optimal action is immediately available. Object-vector representations, as found in entorhinal cortex, satisfy both.

Credit assignment through composition

I will show how states consisting of object-vector representations can provide optimal actions in new environments without additional learning of f. The intuition is that a wall to the east, and a goal to the south-east behind that wall, always require going south around the wall. In other words, the relative configuration of objects is what determines a policy – so that is exactly what state representation $\vec{s}$ should express. Once learned over many environments, the mapping $f(\vec{s}) = \vec{a}$ then immediately applies in a new environment. For that to work, the state representation must contain information about all elements in the new environment. Therefore the only learning needed upon entering a new environment and discovering new objects or rewards, is the integration of all objects into a compositional cognitive map. This map ties the object-vector representations to sensory experience of a location in memory. I suggest a new function for replay: composing the cognitive map.

Replay has previously been implicated as a neural substrate for computations in both TD-learning and planning. Replay is the phenomenon of the reinstatement of remote memories, often in rapid sequences (Foster and Wilson 2006; Gupta et al. 2010). It would thus be well-suited for offline remote value updates for TD-learning. Indeed, the hypothesis that replay carries out those value updates that have the biggest impact on future behaviour explains many observed patterns of replay (Mattar and Daw 2017). Additionally, replay has been proposed to support planning through forward sweeps (also called *preplay*) (Johnson and Redish 2007; Dragoi and Tonegawa 2011). These preplay sequences play out possible future trajectories around decision time (Kay et al. 2020) [†] and could thus implement mental simulation for finding f.

Here, the role of replay is constructive. Replay carries around cortical object-vector representations and binds them to environment locations in hippocampal memory. Through offline path integration it makes available object-vector representations in remote locations without having to physically navigate there.

[†] poem in Appendix

Because the generalisable f operates on object-vector representations, constructing a state-space implicitly performs credit assignment. Since the combined object-vector representations directly provide optimal actions, replay essentially lays down memories of future behaviour.

Chapter outline

This chapter presents, implements, and analyses the proposed model for constructing behaviour. Section 3.1 describes the setup of model and task. It shows how for object-vector state representations f generalises across environments, yielding optimal actions in environment configurations it was not trained on. Composing state spaces through latent learning or replay enables optimal decision making even on the very first encounter of a state. Finally, using replay to bind object-vector representations to locations in memory effectively suppresses path integration errors. Section 3.2 extends the model to support hierarchical representations. For example, navigating a building involves a high-level route between rooms, and low-level actions to get to the correct door within rooms.

The final sections provide support for some important model assumptions. The model heavily relies on object-vector representations, but how are these learned in the first place? Section 3.3 shows how object-vector representations emerge in a neural network model trained on predicting actions for goal approach from new locations. Finally, although demonstrated in physical space, the same model approach will work in any structured space where actions have consistent meaning. For example, in a regular square grid a go-north action has the same consequence everywhere, and is always the opposite of a go-south action. But in abstract state spaces, consistent actions are not necessarily available *a priori*. Section 3.4 proposes a method for finding low-dimensional action embeddings in arbitrary state space structures.

3.1 Constructing behaviour in structured environments

For f to work in any environment, state representations $\vec{s}$ should imply actions independent of environment specifics. Object-centred representations, like object-vector (Høydal et al. 2019) or boundary-vector cells (Lever et al. 2009) found in the hippocampal formation, are capable of that. An object-vector cell (OVC) has a firing field at a fixed distance and allocentric angle to an object, regardless of the absolute location of the object. A population of OVCs therefore encodes the agent's relative position to an object. An OVC representation centred at reward thus immediately enables approach: if the agent is two steps to the east of a reward, they will need to take a step west – independent of the actual location of the reward in the current environment. More complex worlds require more complex state representations, containing vector representations to all objects present. If there is a wall two steps to the east of the agent, and a reward four steps to the east, the agent will need to go around the wall to get to the reward.

Path integration for building maps

So given the full $\vec{s}$, with object-vector representations for relevant objects, $\mathrm{f}(\vec{s}) = \vec{a}$ provides optimal actions. But on entering a new environment, the agent does not yet know where objects are. How do they get $\vec{s}$? Crucially, object-centred representations path-integrate. Once initialised they carry over to new locations through accumulation of self-movement vectors. For instance, an agent that is one step east of a reward and takes a step east is now two steps east of the reward – again, independent of environment specifics. Therefore they can, after initialising an object's representation on first encountering it, keep track of that representation while exploring the environment. By exploring the environment, the agent collects an increasingly complete $\vec{s}$, incorporating objects as they come along.

But path integration as the only mechanism for obtaining $\vec{s}$ is impractical for multiple reasons. It means a location's full $\vec{s}$ is only accessible through first discovering each object from that location, then navigating back – even in familiar environments. In complex environments, it means continuously tracking vector representations of many objects. And path integration accumulates errors the longer it goes on for. All these issues can be overcome by equipping the agent with *memory*. Object vector representations are bound to their corresponding locations in hippocampal memory. Hippocampal memories thus constitute a compositional map, tying the structural building blocks to physical locations. Importantly, creating a new memory of an object vector representation at a location does not even require going there: *replay* enables remote, offline memory encoding. During replay the agent imagines a walk while path-integrating $\vec{s}$, and binds it to the imagined location in memory. Now any time the agent returns to that location (even by teleport), they can retrieve that location's $\vec{s}$. Here replay constructs state spaces in memory – a fundamentally different interpretation from using replay for credit assignment in RL.

Model overview

Figure 3.1 summarises the model and its components. During exploration (orange arrows in Figure 3.1), the agent observes locations (labeled by black numbers) and discovers objects (like walls, in green and blue, and reward, in red). Initially, they do not know where the objects are, so object vector representations are empty (Figure 3.1b, step A). Upon object discovery (for example the green wall, Figure 3.1b, step B) the agent initialises the corresponding object vector representation so it can be path integrated. Learning f as a mapping from the collection of object vector representations to actions generalises to different environments; learning f through reward expectations of locations as in TD learning will not. Then during replay (purple arrows in Figure 3.1) after finding reward (Figure 3.1b, step D), the agent imagines path integration of $\vec{s}$ to remote locations (Figure 3.1c, step E and F), binding $\vec{s}$ to the corresponding location at

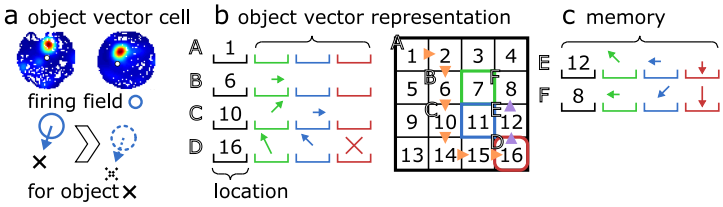


Figure 3.1: Model overview. a. Example object vector cell ratemap, adapted from (Høydal et al. 2019). This object vector cell has a firing field to the north of the object. When the object moves, the firing field moves correspondingly. b. During exploration (orange arrows), the model observes locations (numbers, black slots) and tracks object vector representations (arrows in green, blue, and red slots) through path integration. Object vector representations start empty (step A), but are initialised on object discovery (like the green wall, step B). While path integrating the green wall, the model finds the blue wall (step C), and finally encounters the red reward (step D) to obtain the full state representation. c. In replay, the model carries state representation to remote locations without physical displacement (purple arrows). It then binds the object vector representations to locations in memory (step E, F) for retrieval at that location.

each step. In contrast to its credit assignment role in TD learning, replay now construct the map of state space in memory. On future encounters of these locations the agent can retrieve that $\vec{s}$ indexed by the observed location.

3.1.1 Methods

I have implemented the proposed model in two different flavours: discrete and continuous space. The discrete space model lives on a graph, whereas the continuous space model experiences arenas of two-dimensional Euclidean space. The former has the advantage that it automatically extends to non-spatial state spaces, but the latter compares more naturally to well-studied spatial behaviour, and offers more realistic path integration noise. In principle they are very similar, but they differ slightly in their practical implementations.

Environment structure and representation

The discrete space model considers locations $s \in \mathcal{S}$ on graphs, connected by actions $a \in \mathcal{A}$, as in a deterministic Markov decision process. All results here assume regular square grid worlds, with actions 'go-north', 'go-east', 'go-south', and 'go-west', but that is not a fundamental limitation: the only requirement is

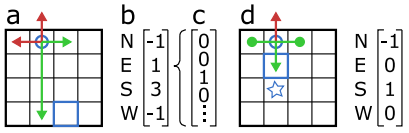


Figure 3.2: Discrete object vector representations. a. Example object vector representation at the location indicated with the blue circle for the blue wall object. I calculate distances along each action to the object, plotted by green arrows; red arrows indicate actions that go in opposite direction. b. The full object vector representation is a vector of distances along each action, here 'go-north' (N), 'go-east' (E), 'go-south' (S), and 'go-west' (W). Opposite actions are indicated by a -1. c. The full state representation that serves as input for f is a one-hot encoding of the distances in b. d. Illustration of why representing actions as opposite is useful: the reward (blue star) is immediately to the south of both the blue wall and blue circle. Distances in east, west, and north direction are all zero, but only east and west are optimal for moving towards the reward.

that the graph can be path integrated. I generate environments by adding objects such as rewards and walls on top of square grids with sides of l_d nodes, making sure walls do not overlap rewards or completely block parts of the environment. Rewards are represented by single objects, walls are represented by a pair of objects: one at each end of the wall, in the same row or column, so that walls can only be horizontal or vertical.

Given an environment populated with objects, I calculate object-vector representations for each location as illustrated in Figure 3.2. The object-vector representation $\vec{s}_o(s)$ at location s for object o consists of a set of distances to o, one along each action a. Starting from s, I repeatedly move along a until the graph distance to o stops decreasing. The number of repetitions is the distance along a. If the distance along a is zero, so a single step along a does not end up closer to o, I additionally check whether action a is in the opposite direction of o: whether taking a from o decreases the distance to s. I convert the integer distance along an action to one-hot coding, where a one in the first entry indicates an opposite action and the remaining entries encode distance. I concatenate distance vectors along all actions to obtain the vector representation $\vec{s}_o(s)$ for object o, and concatenate the vector representations for all objects to get the full location object-vector representation $\vec{s}$ for location s.

Learning a policy for any environment

The model learns how to act through a function $f(\vec{s}) = \vec{a}$ that maps these object-vector representations to policy. This mapping is learned across many environments, so that it generalises to new configurations of objects on first encounter. I implement f as a multi-layer perceptron, consisting of an input layer of the same dimension as $\vec{s}$, a hidden layer of twice that dimension with rectified linear activation, and an output layer of the same dimension as $\vec{a}$. As training data, I generate environments and specify a representation $\vec{s}$ and an optimal policy $\vec{a}$ for each state. Dijkstra's algorithm provides shortest-path distances from each location to the reward, taking walls into account; the optimal policy then assigns equal probability for all actions that decrease the graph distance to the reward object, and zero probability for all others. I train f using supervised learning on batches of n_b input $\vec{s}$s and output $\vec{a}$s, each sampled from a different environment. Because f is trained on multiple environments in parallel, it needs to learn a mapping that works across environment configurations. After a n_s number of samples I generate new training environments, n_e times. Pytorch calculates gradients to update f's parameters through backpropragation using the ADAM optimiser with learning rate 10^{-3}.

Building a map of the current environment

Once the mapping $f(\vec{s}) = \vec{a}$ is learned successfully, it can be applied in any new environment configuration, without additional credit assignment. A simulated episode in such a new environment then runs as follows. Initially, the modeled agent does not know where objects are in the current environment, so their object-vector representations start empty. They explore the environment by observing their location, selecting an action, and transitioning to the next location. Discovering an object, at a distance of one step for wall ends or zero steps for reward, initialises the object vector representation $\vec{s}$ of that object. For easier notation I will write the *represented location* s instead of the full *representation* $\vec{s}$; either works, because there is a one-to-one correspondence

between the two in a given environment. From then on, the agent needs to update that object vector representation to reflect the action taken at each step through path integration. However, path integration is noisy. Errors might cause the *represented* object vector to diverge from the true vector relation. I model path integration noise as a distribution over transitioned locations s_{PI} given action a in location s:

$$p_{PI}(s_{PI}) = \begin{cases} \frac{1-\epsilon}{|S'|} & \text{if } s_{PI} \in S' \\ \frac{\epsilon}{|S''|} & \text{if } s_{PI} \in S'' \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

where

$$S' = \{s' \in \mathcal{S} \mid p(s'|s,a) > 0\} \tag{3.2}$$

is the set of locations that taking a from s can transition to – $p(s'|s,a)$ is defined by the environment; $|s'| = 1$ for deterministic transitions – and

$$S'' = \{s'' \in \mathcal{S} \mid s'' \notin S' \text{ and } \sum_{a'}^{\mathcal{A}} p(s''|s',a') > 0\} \tag{3.3}$$

the set of neighbours of s'. Path integration thus updates the representation to $\vec{s}(s_{PI})$, corresponding to the correct location with probability $1-\epsilon$ or any of its neighbours with total probability ϵ.

In addition to path integration, the agent relies on *memory* to update representations after an action. I model memory as a dictionary $M(s) = [s_1, ...]$ with locations as keys and multisets of representations (or, again for ease of notation: the location corresponding to a representation) as values. In each step the agent appends their current representation to their memories at the current location. On arrival at location s, they retrieve all previously encoded representations for that location $M(s)$ to obtain a representation probability distribution:

$$p_M(s_M) = \frac{|[s_{M(s)} \in M(s) \mid s_{M(s)} = s_M]|}{|M(s)|} \tag{3.4}$$

Now they can make use of the *combined* distribution to update their representation: a weighted sum over the path-integrated representation (weight w),

and the distribution over representations on earlier encounters of this location (weight $1 - w$).

$$p(s) = w p_{PI}(s) + (1 - w) p_M(s) \tag{3.5}$$

Then sampling s from Equation (3.5) produces the updated state representation $\vec{s}(s)$.

Replay in this model allows the agent to encode remote memories offline. Given an initial location and representation, they *imagine* transitions through the environment. After each transition, they update their representations and append them to the current location's memories, as during online navigation. There is one difference: Since transitions are now imagined, the current location cannot be observed from the environment. Instead, it needs to be path integrated, just like the object-vector representations. Again, I model path integration noise by sampling from a distribution over correct and neighbour locations. The agent performs r_n replays of r_l steps long, after every r_f steps of exploration.

Goal-directed navigation in model and controls

After exploring the environment, having discovered all objects, the agent can 'exploit': they can use their object vector representation $\vec{s}$ as input to f and select the highest probability action in $\vec{a}$. They take the resulting action, transition to a new location, update their representation $\vec{s}$ and repeat the same procedure until arriving at reward. To compare the model's performance on navigating towards reward, I implement two control models. The first, a 'path integrator', relies only on path integration to update object-vector representations, without access to memory. There is no role for replay in the 'path integrator'. The second control model, a 'TD learner', relies on observed locations only and learns reward expectations for actions using TD updates. The model calculates these reward expectations, or q-values, by multiplying state representation $\vec{s}$

by $|\mathcal{A}| \times |\mathcal{S}|$ weight matrix W:

$$\vec{q}(\vec{s}) = W\vec{s} \tag{3.6}$$

where for the 'TD-learner' agent the state representation $\vec{s}$ is a one-hot encoding of the observed location. Then the TD update for W, on taking action a from location s, resulting in a transition to s' and reward r, becomes

$$W_a \leftarrow W_a + \alpha\vec{s}(r + \gamma \max_{a'} W_{a'}\vec{s}' - W_a\vec{s}) \tag{3.7}$$

where γ is the temporal discounting factor and α the learning rate. The 'TD-learned' agent does not track object vector representations, so does not need to path integrate representations during navigation. They use replay for additional offline remote TD updates, by applying Equation (3.7) to imagined transitions. That does require path integration of $\vec{s}$, since transitioned locations are not observed but imagined during replay.

Environment structure and representation

The continuous space model experiences two-dimensional Euclidian space instead of graph environments. That changes a few things. Instead of graph nodes, locations are now defined by (x, y)-coordinates in an l_c by l_c arena. Object-vector representations consist of populations of n_c object-vector cells, with spatially tuned firing rates. I distribute the firing fields of the OVC population evenly on a square grid that spans twice the arena length in each direction, with the population's center aligned on the object it represents. This configuration makes sure that the whole arena is covered by OVCs for any possible object location. I model a single OVC firing field as a two-dimensional multivariate Gaussian, with a covariance matrix that has firing field width Σ on the diagonal and zeros elsewhere. The representation $\vec{s}$ at location (x, y) for object o is thus the n_c-dimensional vector of multivariate Gaussians centred around o evaluated at (x, y). Figure 3.3 plots populations of object vector cells that represent objects and place cells that represent locations in different environments.

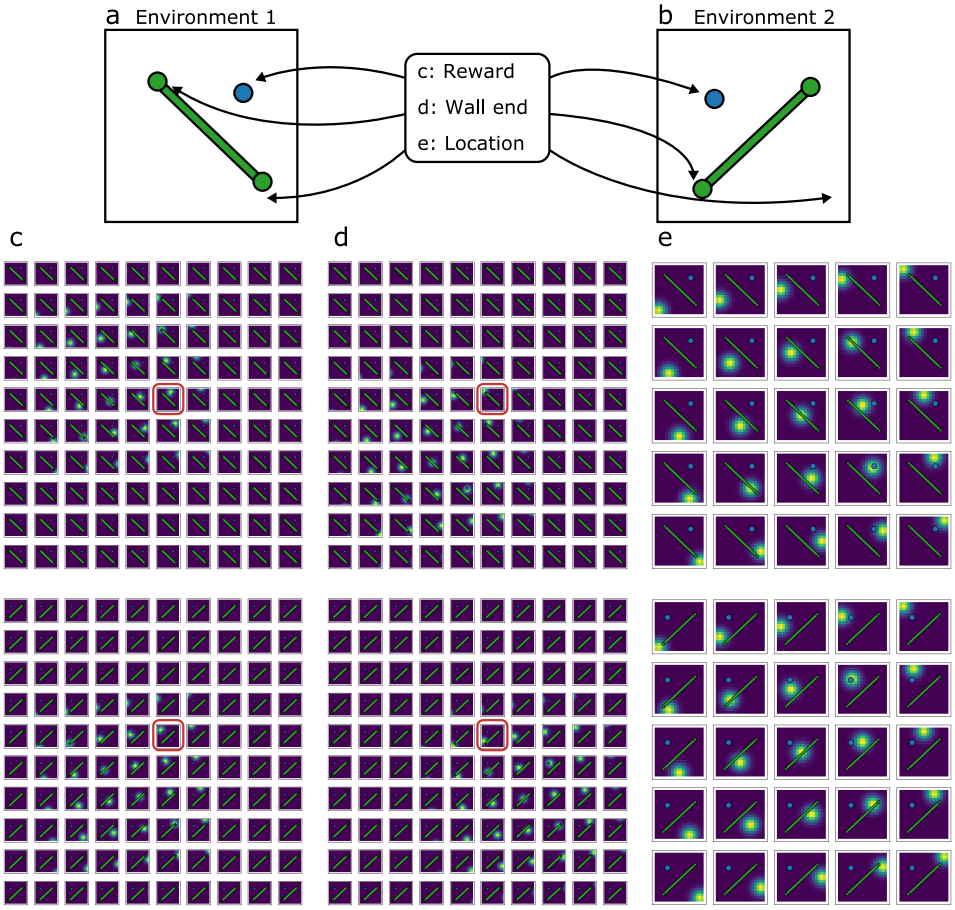


Figure 3.3: Continuous object vector representations. a. Continuous environment with a reward (blue circle) and a wall (green bar), represented by two wall ends (green circles). b. Continuous environment with different configuration. c. Object vector cell population for the reward object. Each square is a rate map for one cell, the top row for the environment in *a* and the bottom row for the environment in *b*. The rate map circled in red, for example, shows a cell that fires just west of the reward in both environments. d. Object vector cell rate maps for one wall end, with the cell circled in red firing just west of the wall end in both environments. e. Place cell population that represents location in both environments, independent of object orientation.

Learning a policy for any environment

Again, I train multi-layer perceptron f to map representation $\vec{s}$ to action $\vec{a}$. In continuous space I define actions as directions. Because direction wraps around (from 0 to 2π), f learns a two-dimensional output $\vec{a}$ given the input of object-vector population firing rates concatenated for all objects $\vec{s}$: the sine and the cosine of the optimal direction. I can then recover the optimal direction α from

$$\alpha = \text{atan2}(a_y, a_x) \tag{3.8}$$

To generate training examples of input $\vec{s}$ and output $\vec{a}$, I create empty arenas and populate them with walls, again represented by an object at each end, and reward objects. For supervised learning I need the optimal direction towards reward at any given (x, y) location $\vec{x}$. I get the optimal direction as follows. First, I construct a graph of the environment, creating an edge between each pair of objects weighted by their Euclidian distance – but only if the line connecting that object pair in the arena does not cross any walls. Then I use Dijkstra's algorithm to calculate the graph distance to the nearest reward for each object. Then for a given location $\vec{x}$, the optimal direction points towards the object o, reachable in a straight line from $\vec{x}$ without crossing walls, for which the distance from $\vec{x}$ to o together with the graph distance from o to reward is smallest. Provided with pairs of $\vec{s}$ and optimal $\vec{a}$, training f proceeds in the same way as in the discrete model.

Building a map of the current environment

During simulated exploration, the continuous space agent takes steps $\vec{d}$ of length d_r (capped at d_R) in direction d_θ, by adding normal random variation with standard deviation σ_r, σ_θ to the previous length and direction for smooth trajectories. An additional complication of the continuous world compared to its discrete counterpart is the agent's interaction with walls. The agent cannot pass through walls, but should still be able to slide along walls if they are not moving perpendicularly towards the wall, unless there is another wall preventing that (as in corners). Algorithm 1 resolves an attempted step $\vec{d}$ from location $\vec{x}$ to $\vec{x}'$ in the presence of walls, with wall normal vectors pointing from $\vec{x}$ orthogonally into the wall.

The agent thus explores the environment while discovering objects (Figure 3.4). On object discovery, when the agent is within a radius r_o of object o, they initialise the object vector representation for that object. Again, I will write *represented locations* $\vec{x}_s$ instead of *representations* $\vec{s}$ for easier notation. From then on, they track the object's representation through path integration

Algorithm 1 Transition with walls

1. Is $\vec{x}$ within distance l_w of a wall?

Yes Is $\vec{x}$ within distance l_w of multiple walls?

Yes Calculate the dot products of $\vec{d}$ and the normal vectors of the walls within l_w

Remove the positive component along the normal vector of the wall with the largest dot product from $\vec{d}$.

Recalculate the dot product of $\vec{d}$ and the normal vector of the wall that had the second-largest dot product. Is it greater than zero?

Yes Set $\vec{d}$ to $\vec{0}$; *continue to 2*.

No *continue to 2*.

No Remove the positive component along the normal vector of the wall within l_w from $\vec{d}$; *continue to 2*.

No *continue to 2*.

2. Does the line from $\vec{x}$ to $\vec{x} + \vec{d}$ cross any walls?

Yes Set new location $\vec{x}'$ to wall contact location, l_w away from the wall crossed first; *exit*.

No Set new location $\vec{x}'$ to $\vec{x}' = \vec{x} + \vec{d}$; *exit*.

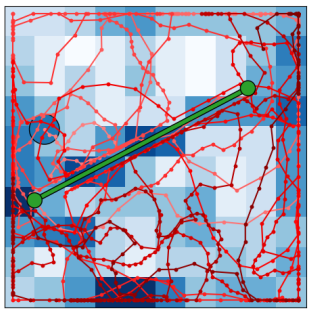


Figure 3.4: Continuous behaviour. Example walk of agent in a continuous environment, each step marked with a red dot connected to the previous step with a red line, progressing from light red to dark red over the course of the walk. Blue patches indicate binned location occupancy. Interactions with the green wall and arena boundaries demonstrate how the agent slides along blockades without crossing them.

and memory retrieval. Path integration updates represented locations to reflect step $\vec{d}$, with path integration error modelled as Gaussian noise with standard deviation ϵ_r and ϵ_θ on step length d_r and direction d_θ respectively:

$$\begin{aligned}
\vec{x}_{PI} &= \vec{x}_s + \vec{d}_{PI} \\
\vec{d}_{PI} &= d_{r,PI}\cos(d_{\theta,PI})\hat{x} + d_{r,PI}\sin(d_{\theta,PI})\hat{y} \\
d_{r,PI} &= d_r + e_r, e_r \sim \mathcal{N}(0, \epsilon_r) \\
d_{\theta,PI} &= d_\theta + e_\theta, e_\theta \sim \mathcal{N}(0, \epsilon_\theta)
\end{aligned} \tag{3.9}$$

Note that Equation (3.9) follows the location update algorithm, taking walls into account. I encode a new memory as a location-representation pair (again, for ease of notation: I store represented locations instead of actual representations) at every step, so that the memory bank $M = [\{\vec{x}_1, \vec{x}_{s1}\}, ...]$ becomes a multi-set of pairs. Then to retrieve representation $\vec{x}_M$ at location $\vec{x}$ from all previous experience, I calculate a weighted average across represented locations $\vec{x}_{si}$ in memory. The weights are given by the corresponding memory locations $\vec{x}_i$: memories created near $\vec{x}$ contribute stronger to the retrieved representation. More specifically, I exclude memories created beyond distance threshold ξ_M, and then weight remaining memories by the softmax of dot products between place-representations of $\vec{x}_i$ and $\vec{x}$:

$$\vec{x}_M = \frac{\sum_{\{\vec{x}_i, \vec{x}_{si}\} \in V} e^{\beta_M \vec{p}(\vec{x}) \cdot \vec{p}(\vec{x}_i)} \vec{x}_{si}}{\sum_{\{\vec{x}_i, \vec{x}_{si}\} \in V} e^{\beta_M \vec{p}(\vec{x}) \cdot \vec{p}(\vec{x}_i)}} \tag{3.10}$$

where β_M is the softmax inverse temperature,

$$V = [\{\vec{x}_i, \vec{x}_{si}\} \in M \mid \sqrt{(x_x - x_{i,x})^2 + (x_y - x_{i,y})^2} < \xi_M] \tag{3.11}$$

contains only the included memories, and $\vec{p}(\vec{x})$ translates a location to a place-representation – a vector of population firing rates very much like the OVCs, but with firing fields on a grid of absolute arena locations, instead of relative to objects (Figure 3.3e). The final updated representation combines the path integration and retrieved represented location:

$$\vec{x}_s = w\vec{x}_{PI} + (1 - w)\vec{x}_M \tag{3.12}$$

from which I calculate the OVC population firing rate $\vec{s}(\vec{x}_s)$. The replay implementation in the continuous model is very similar to the discrete case. The agent imagines transitions and makes a new memory at each replay step, path-integrating both location $\vec{x}$ and represented location $\vec{x}_s$ – in contrast to physical navigation, where the location $\vec{x}$ is observed.

Goal-directed navigation in model and controls

Once all objects have been found and exploration has ended, exploitation is similar to the discrete case. Plugging the full object vector representation $\vec{s}$, containing vector relations for all objects, into f produces optimal direction a (Equation (3.8)). The agent takes a step of size d_R in direction a, updates their representation $\vec{s}$, obtains a new a through f and continues until within r_o from reward. The 'path-integrator' control model only relies on the continuous path integration of Equation (3.9) to update representations. The 'TD-learner' control model uses the same state-action value weight update of Equation (3.7), both during physical navigation and imagined transitions in replay. The states for which it learns weights are the place representations $\vec{p}(\vec{x}_s)$. Because Equation (3.7) assumes discrete actions, I discretise direction into n_d steps for the TD-learner.

Table Table 3.1 lists all parameters that appeared in this Section with a short description and default values.

3.1.2 Results

Object centred state representations support generalisable learning

The mapping $f(\vec{s}) = \vec{a}$ successfully learned optimal actions in discrete space. It achieved high policy accuracy both in a simple environment that is empty except for a single reward (Figure 3.5), and in a more complex environment that includes one or more walls (Figure 3.6). I evaluate the accuracy of f by, starting from each environment location, following the maximum probability action from $\vec{a}$ given by f, and then the new $\vec{a}$ in the new location, and so on.

Name	Value	Description
l_d	7	Number of nodes along one side of square grid graph
n_b	25	Number of training samples in batch, each from a different environment
n_s	200	Number of training samples from the same environment
n_e	20	Number of times new training environments are generated
ϵ	0	Path integration error probability
w	0.2	Weight of path integrated representation (versus memory retrieved representation) in representation update
r_n	5	Number of replays repeats at one step
r_l	15	Replay length: number of steps in one replay
r_f	4	Replay interval: number of physical steps after which to replay
γ	0.7	TD learner temporal discounting factor
α	0.8	TD learner learning rate
l_c	1	Length of continuous two-dimensional arena side
n_c	100	Number of object vector cells in object vector representation of one object
d_R	0.1	Maximum step size
σ_r	0.01	Step size variation standard deviation
σ_{theta}	$0.1 \cdot 2\pi$	Step direction variation standard deviation
l_w	0.025	Wall radius (or half the wall thickness)
r_o	0.05	Radius for perceiving objects
ϵ_r	0.025	Path integration step size noise standard deviation
ϵ_θ	$0.25 \cdot 2\pi$	Path integration step direction noise standard deviation
ξ_M	0.05	Distance cutoff for including memories in retrieval
β_M	1	Memory retrieval weight softmax inverse temperature
n_d	16	Number of discretisation steps for direction in TD learner

Table 3.1: Model parameters. List of parameters that appear in Section 3.1

I then count the fraction of initial locations from which f successfully navigates to the goal. I contrast the performance of f when trained on *location* state representations $\vec{s}$, which is a one-hot location encoding, to *object vector* state representations $\vec{s}$. When f is trained on a single environment, and evaluated on that same environment, f achieved high accuracy both for types of representation (Figures Figure 3.5c, Figure 3.6c). However, when trained on multiple environments in parallel by providing batches of $\vec{s}$, $\vec{a}$ training pairs taken from different environments, f's accuracy for location state representations dropped dramatically. A policy based on object vector state representations remained

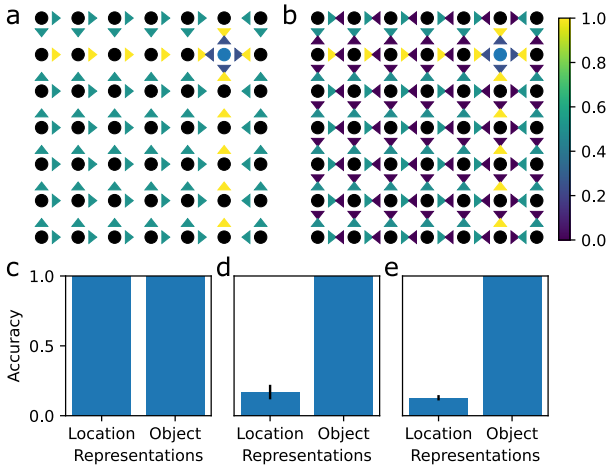


Figure 3.5: Simple environment. a. Optimal policy for regular square grid graph with single reward (blue). Locations are marked by circles, actions by triangles, the triangle colour indicates the probability under the optimal policy (zero-probability actions not shown). b. Learned policy obtained from $f(\vec{s}) = \vec{a}$, again with actions colour indicating probability. c. Policy accuracy $\pm$ s.e.m., measured as the fraction of locations from which following the policy successfully leads to reward, for one-hot location representations ('Location') and object vector representations ('Object'). Here I train f and test the policy accuracy in the same single environment. d. Policy accuracy $\pm$ s.e.m. when I train f and test the policy accuracy in multiple environments in parallel. e. Policy accuracy $\pm$ s.e.m. when I train f in multiple environments in parallel and test the policy accuracy in a new set of environments.

highly accurate (Figures Figure 3.5d, Figure 3.6d). The reason for this discrepancy is that locations are not predictive of correct actions across environments, whereas vector relationships between objects are. This generalising property of object vector representations is further demonstrated by evaluating f, trained on many environments, in a new environment without any further training (Figures Figure 3.5e, Figure 3.6e). Again, location representations did not provide accurate actions in this new environment, but object vector representations did: the latter allows for optimal policies in new environments *without* any additional learning. The same applies in environments where optimal paths take twists and turns, like an S-maze (Figure 3.7). Trained on different S-maze configurations, f supplied with object-vector representation $\vec{s}$ produces optimal actions in a never-seen-before S-maze.

The continuous space model showed the same pattern of f accuracies (Figure 3.8). Now f maps population firing rates $\vec{s}$ to optimal continuous directions a

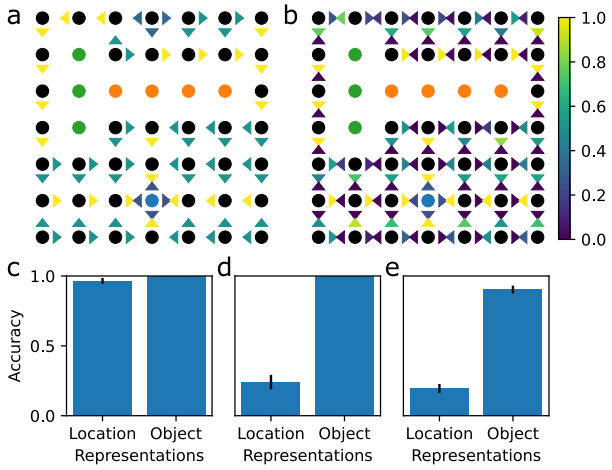


Figure 3.6: Complex environment. a. Like Figure 3.5a, but with walls. Optimal policy for regular square grid graph with two walls (orange and green) and a single reward (blue), again action colour indicating probability. b. Learned policy obtained from $f(\vec{s}) = \vec{a}$. c. Policy accuracy $\pm$ s.e.m. for one-hot location representations ('Location') and object vector representations ('Object'), with f trained and tested in the same single environment. d. Policy accuracy $\pm$ s.e.m. when I train f and test the policy accuracy in multiple environments in parallel. e. Policy accuracy $\pm$ s.e.m. when I train f in multiple environments in parallel and test the policy accuracy in a new set of environments.

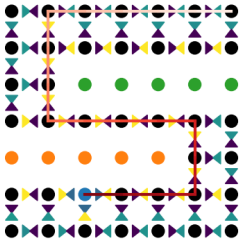


Figure 3.7: S-maze. Learned policy for a new S-maze after training on different S-mazes, with action colour indicating probability, and an example walk from light to dark red following the policy from the top of the S-maze to the reward.

(Figure 3.8a-d). I evaluate its accuracy by considering a grid of starting locations, then taking a step in direction a provided by f, then f's output direction a at the new location, and so on. Again, I count the fraction of initial locations from which the goal is reached as accuracy for f. I compare f trained on *place cell* codes, where cells have firing fields at fixed locations in the environment, to f trained on *object vector cell* codes, where cells have firing fields at locations relative to objects. As in discrete space, single environment policies were equally accurate (Figure 3.8e), but object vector cell representations afford generalis-

able mappings (Figure 3.8f). Again, f learned for OVC representations produce accurate policies in new environments without additional learning (Figure 3.8g).

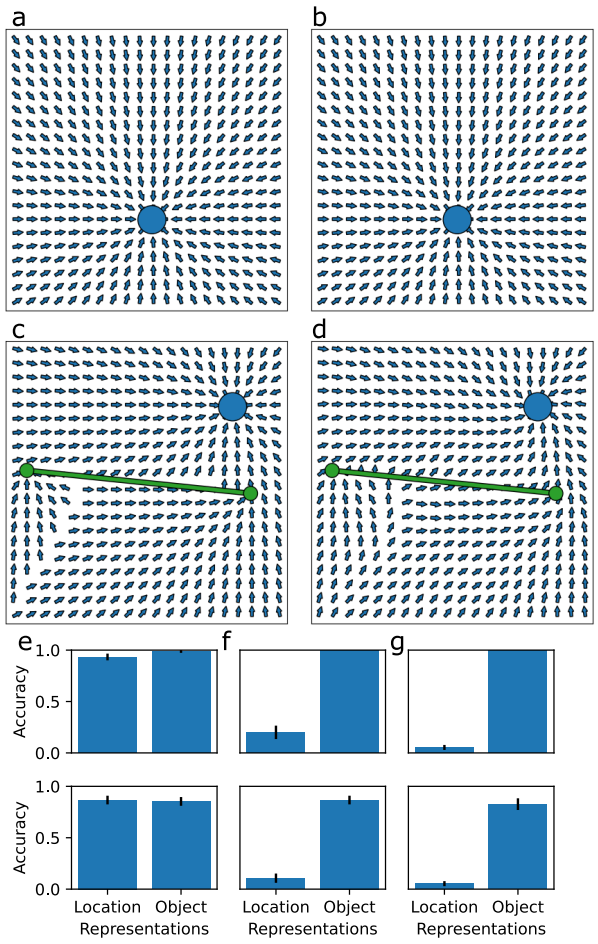


Figure 3.8: Continuous policy. a. Optimal policy for continuous arena with single reward (blue circle). I plot policy as directions on a grid of locations. b. Learned policy obtained from $f(\vec{s}) = \vec{a}$. c. Like *a*, but with a wall. Optimal policy for arena with a wall and a reward. d. Learned policy from $f(\vec{s}) = \vec{a}$. e. Like Figure 3.5c and Figure 3.6c, but in continuous space. Policy accuracy $\pm$ s.e.m., measured as the fraction of locations from which following the policy successfully leads to reward, for place representations ('Location') and object vector representations ('Object'). Here I train f and test the policy accuracy in the same single environment: with reward only (top) or reward and wall (bottom) f. Policy accuracy $\pm$ s.e.m. when I train f and test the policy accuracy in multiple environments in parallel, with reward only (top) or reward and wall (bottom). g. Policy accuracy $\pm$ s.e.m. when I train f in multiple environments in parallel and test the policy accuracy in a new set of environments, with reward only (top) or reward and wall (bottom).

Latent learning as binding representations in place

Given a pre-learned f, the agent needs the OVC representation $\vec{s}$ of a location to calculate $f(\vec{s}) = \vec{a}$ in a new environment. Latent learning of $\vec{s}$ allowed for optimal actions in distant locations on first encounter after reward discovery (Figure 3.9). I simulate an agent exploring the discrete environment, discovering objects on their way, initialising and path integrating object vector representations until finally ending up at reward. Through latent learning they have composed $\vec{s}$ object by object, and upon arrival at reward $\vec{s}$ is complete. From then on, the agent continues exploring the environment, while I measure whether a predicted by $f(\vec{s})$ is indeed optimal. I contrast this to a similar agent but without latent learning, so that they need to discover all objects *after* finding reward. The former agent had access to optimal actions immediately on the first visit to a location after reward; the latter required multiple visits (Figure 3.9a). The equivalent simulation in continuous space showed the locations where the policy fails on first visit without latent learning (Figure 3.9c). These locations required knowledge of all objects, not just reward, for correct actions.

Replay for memory is efficient

Replay provided the agent access to OVC representations in remote locations without physically navigating there. I simulate an agent that has discovered all objects in an environment and arrives at reward with a complete object vector representation $\vec{s}$. They then use replay to distribute the full representation across the environment. Here replay consists of imagined transitions through the environment, where at each step the agent encodes a memory as a location-representation pair. Replay thus constructs a map of representational building blocks in memory. On future visits to that location, the representation can be retrieved so that the agent immediately knows how to act. As in physical path integration, replay afforded optimal actions after a single (imagined) visit to a location. I compare this replay-for-construction to replay-for-credit-assignment

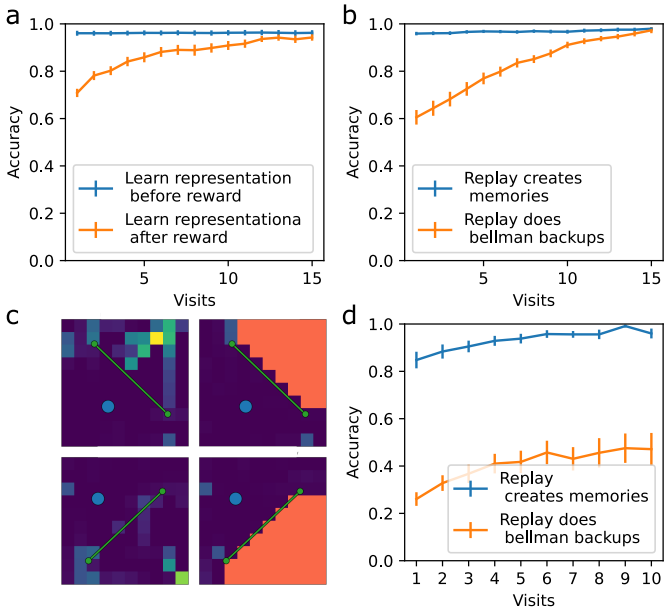


Figure 3.9: Latent learning and replay. a. Policy accuracy ± s.e.m. when learning representations before reward discovery (*latent learning*) in a discrete environment with two walls versus only learning representations after reward discovery, plotted against the number of visits to a location. b. Policy accuracy ± s.e.m. when replay creates memories by binding object vector representations to locations versus replay that does Bellman backups, as a function of replayed location visits. c. Heatmap that plots the additional distance toward the reward (blue circle) when following $f(\vec{s}) = \vec{a}$ compared to the optimal policy – red colours mean $f(\vec{s}) = \vec{a}$ does not lead to reward – in two different continuous environments (top and bottom row). The left column plots this heatmap for $f(\vec{s}) = \vec{a}$ when $\vec{s}$ holds the full representation, including walls, as learned before reward discovery (latent learning); the right column for when only the reward object vector representation is available. d. Like *b*, but in continuous space. Policy accuracy ± s.e.m. in continuous space when replay creates memories by binding object vector representations to locations versus replay that does Bellman backups, as a function of replayed location visits. I discretise continuous locations to count visits and I discretise continuous action directions for Bellman backups.

in a 'TD-learner' agent. They use the imagined transitions in replay to perform credit assignment by performing reward expectation updates (or *Bellman backups*). This kind of replay required more (imagined) experience to provide accurate actions (Figure 3.9b). Continuous space simulations produced a similar pattern of results, albeit noisier due to discretisation of location for evaluation and of direction for Bellman backups, with a higher amount of replay needed for replay-for-credit-assignment than replay-for-construction (Figure 3.9d).

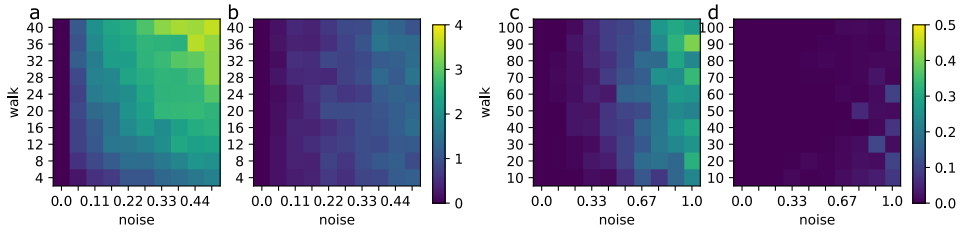


Figure 3.10: Replay suppresses noise effectively. a. Heatmap of final distance from home in the noisy homing task (distance in discrete locations, same colour scale as in b) for the 'path integrator' agent that relies on path integration only, with walk length on the y-axis and path integration noise on the x-axis. b. Distance from home in noisy homing task for agent that uses memory for path integration error correction. c. Like *a*, but in continuous space. Final distance from home in the noisy homing task (Euclidian distance in square continuous arena with sides of length 1, same colour scale as in d) for the 'path integrator' agent, with walk length on the y-axis and path integration noise on the x-axis. d. Distance from home in noisy homing task for continuous-space agent that uses memory for path integration error correction.

Replay suppresses path integration noise

Tracking object vector representations, during navigation or replay, is very effective in principle, but has serious issues in practice: path integration is noisy, and errors accumulate over time. Encoding location-representation memories in replay solved this problem effectively and efficiently (Figure 3.10, Figure 3.11). During exploration after finding an object, the agent does not only path integrate $\vec{s}$, but also replays. They imagine path integrating $\vec{s}$ towards different locations while making memories of that imagined $\vec{s}$ at each location. If the agent replays repeatedly, path integration errors will be independent between replays, and cancel out on average. I demonstrate the error-correcting power of replay and memory in a homing task. The agent starts at their home location, explores the arena, until they suddenly have to escape towards home as quickly as possible. I measure how far away from home they end up by following $f(\vec{s}) = \vec{a}$ on escape. Repeated replay for memory construction strongly improved homing behaviour accuracy compared to a 'path integrator' agent without memory, particularly for long exploration and high path integration noise scenarios (Figure 3.10a-b). The same applied in the continuous case, where the implementation of path integration noise is more realistic (Figure 3.10c-d).

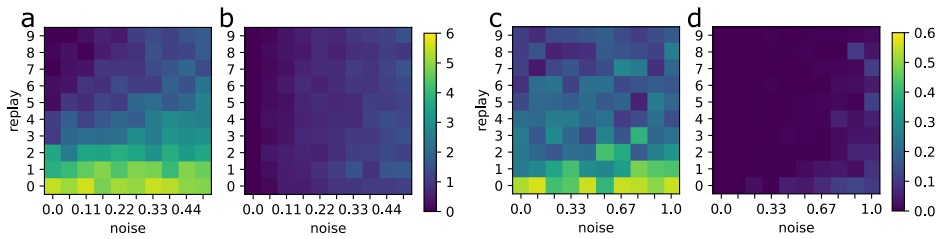


Figure 3.11: Replay suppresses noise efficiently. a. Heatmap of final distance from home in the noisy homing task (distance in discrete locations, same colour scale as in b) for the 'TD learner' agent that uses replay for location credit assignment, with the number of replays on the y-axis and path integration noise on the x-axis. b. Distance from home in noisy homing task for agent that uses replay to construct location-representation memories. c. Like *a*, but in continuous space. Final distance from home in the noisy homing task (Euclidian distance in square continuous arena with sides of length 1, same colour scale as in d) for the 'TD learner' agent, with the number of replays on the y-axis and path integration noise on the x-axis. d. Distance from home in noisy homing task for continuous-space agent that uses replay to construct location-representation memories.

In order to suppress path integration noise, replay-for-construction now requires multiple repeated replays, just like replay-for-credit-assignment needed repeated replay to propagate credit along long distances. How does the required amount of replay compare? I simulate homing behaviour with replay-for-construction for different amounts of replay, and contrast it to a 'TD-learner' agent that performs replay-for-credit-assignment to learn the homing policy (Figure 3.11). Replay-for-construction achieved higher accuracy with fewer replays, both in the discrete (Figure 3.11a-b) and the continuous setting (Figure 3.11c-d).

3.1.3 Discussion

I have shown how state spaces can be constructed from building blocks that directly imply optimal behaviour. Policies trained on object vector representations generalise to new environment configurations without additional learning. I interpret latent learning and replay as constructive processes: Both support the composition of the current environment from its constituent objects. Replay constructs a cognitive map of the current environment by tying vector representations to locations in memory – a fundamentally different interpretation from

replay for credit assignment in RL or forward sweeps in planning. Combining latent learning and replay also avoids tracking many different objects at the same time in complex environments: on object discovery, replay lays down object vector representations in memory, so they are available though memory retrieval instead of ongoing path integration. Repeated replay proves an effective and efficient way of suppressing path integration noise.

The accuracy of a generalisable policy is determined by state-action mapping f. Currently f is implemented as an arbitrary non-linear function in the form of an MLP. This particular choice is sufficiently capable in the environments presented here, but may require increasing effort to train as environments grow in complexity. Supplying f with appropriate inductive biases can alleviate these training difficulties. For example, *hierarchical* neural network architectures encourage combining low-level features into common feature patterns. For example, the two wall end object vector codes can be combined into a single wall representation, and multiple walls into corners or corridors. Alternatively, *graph networks* are particularly well suited for solving problems that can be cast into graph form. Mapping object vector representations to actions is such a problem: it requires reasoning over objects, like walls and rewards, and their specific relations, like whether the wall is blocking the reward or not. In fact, finding the optimal solution for generating training examples in continuous space environments involves constructing a graph of objects and their shortest distance to reward while avoiding walls.

In addition to inductive biases, another way of reducing the amount of required training data is to *imagine* instead of experience new environments. As representations are compositional, new environments can be generated by construction from building blocks in imagination. Using such an imagined environment for training f requires having output solutions (optimal actions) as well as input configurations (object-vector representations). Replay in its hypothesised function of forward sweeps for planning could provide those solutions. Given an imagined environment and state representation $\vec{s}$, replay would through search

find the corresponding optimal action $\vec{a}$. The fact that search is time-consuming and error-prone is not as problematic when performed off-line during rest for training f, compared to during online behaviour.

The presented model relies on two types of representations: the object-vector representations, used for policy, and representations of location, used for memory. I have described their algorithmic application; what is their implementation on a more mechanistic level? The location representation can be obtained from observation in the environment, for encoding and retrieving memories. But it must also be accessible without observation, when imagining transitions during replay. I suggest grid cells as substrate for location representation that meets these requirements. A population of grid cells encodes the agent's location in the environment, in a way that supports path integration and location memory. Because grid cells path integrate, they support imagined transitions during replay. Because of their role in location memory, grid cells support obtaining the location representation from observation. The Tolman-Eichenbaum Machine (Whittington et al. 2020) proposes that hippocampal place cells implement location memory through *conjunction* of sensory observations and abstract location, provided by grid cells, in an attractor network. In such a setup, an observation input to the attractor pattern-completes to the place representation, from which location's grid cell code can be retrieved.

So I propose that just like the conjunctive binding of grid representations to observations, object vector representations are tied to grid representations in memory through conjunction between the two in hippocampus. But then tracking many objects requires many object vector representations, with a population of OVCs specific to each object. I suggest an alternative implementation: binding *object identity*, the OVC code corresponding to that object, and the grid code in a three-way conjunction. In that case, a single population of OVCs suffices to track many objects. That means that the full object vector code, that serves as input to state-action mapping f, ends up in hippocampus instead of entorhinal cortex. The main disadvantage is that path integrating multiple

object vector representations at the same time becomes cumbersome, because each needs to be reinstated from memory separately. However, maintaining many object vector representations at the same time is undesirable in the first place. Laying down an object's vector representation map in memory on its discovery for subsequent retrieval avoids ongoing path integration of multiple objects. The object identity part of the conjunction could also function as a context signal, for example to solve a task in which one location is rewarded when a green light is on and another location is rewarded when a red light is on. If external stimuli determine the object identity code used for retrieving the object vector representation that is fed into f, the agent will approach the appropriate reward location depending on the context.

Recently discovered properties of replay can be interpreted in light of its proposed function of constructing a cognitive map that implies a policy in memory. In a many-armed maze with dynamic reward, replayed locations are not predictive of upcoming behaviour; instead, locations with (past or present) reward delivery and locations not recently visited dominate replay, suggesting replay's involvement in maintaining memory instead of planning (Gillespie et al. 2021). When replay is silenced, hippocampal place map stability is impaired (Roux et al. 2017). When the location of an intermediate goal on a trajectory changes, replay events focus around that new waypoint; disrupting replay produces unstable behaviour after the waypoint change (Igata et al. 2021). In an arena with frequent wall configuration changes, replay trajectories rapidly adapt to conform to new barriers, whereas the place cell location representations remain stable (Widloski and Foster 2022). These observations are in accordance with replay contributing to map construction for guiding behaviour. Replay trajectories carry momentum (Krause and Drugowitsch 2022), and in humans sequences that were shuffled during presentation are replayed in unshuffled order (Liu et al. 2019), as expected for replay that path integrates.

Hypothesising that the goal of replay is constructing memories to guide future behaviour raises the question what the content of replays should be

to optimally achieve that goal. A future direction for the current work would be to formulate optimality of replay for memory in a similar fashion to the prioritised replay framework, that derives which experiences should be replayed to update state-action value estimates efficiently (Mattar and Daw 2017). That framework considers for all available updates a 'gain', which reflects how much the value estimate will change at a location, and a 'need', that expresses the relevance of that location for future behaviour. I suspect similar terms exist when replay encodes memories: representation updates (like adding in wall vectors) change policies more in certain locations (like those behind walls) than others (those near reward), and certain locations (like those near wall ends) are encountered more commonly than others (like corners). A theory of what should be replayed when would allow for precise experimental predictions to disambiguate different replay functions.

Another extension of the work presented here is to incorporate hierarchical representation and replay in the same model. Hierarchy simplifies plans over extended state spaces by planning on different levels of granularity. Section 3.2 shows how the same principles of generalising f with object vector representations and building maps in memory can be applied in hierarchical spaces.

3.2 Hierarchical representation and replay

Confronted with complicated tasks that involve long sequences of many actions, humans often exploit hierarchical solutions (Botvinick 2008; Logan and Crump 2011). Cooking multiple courses for a fancy dinner party requires a series of efforts, often with interdependence: frying the onions can only occur after chopping them. Planning each individual action would pose a problem of infeasible complexity. Instead, the cook considers preparing the full meal in temporally abstracted chunks of action sequences. Cooking dinner consists of preparing starter, main, and desert; the main dish requires boiling pasta and making a sauce; cooking a sauce involves chopping vegetables then frying them; et cetera, all the way down to individual physical acts. The cook has

thus successfully overcome the *scaling problem* that many RL algorithms suffer from: their effectiveness deteriorates as the size of the problem grows (Barto and Mahadevan 2003).

As the number of states increases, credit assignment and search in standard RL become acceleratingly difficult (Botvinick et al. 2009). Therefore, just like the cook, the framework of hierarchical RL (Botvinick 2012) introduces temporally abstract actions to learn policies in large state spaces. An hierarchical RL agent can choose between *primitive* actions, the ones available to standard RL algorithms, and *options*, subroutines that group together related actions (Sutton et al. 1999). Each option specifies its own policy towards the subgoal associated with that option. For example, in a big state space that consists of rooms connected by doors, the agent would invoke an option that takes them to a door, until they arrive at the room with reward where they follow primitive actions towards that reward. Options break up learning over many states and steps into smaller and shorter subtasks. The effectiveness of hierarchical RL depends on good subtask definitions, for example those minimising planning description length (McNamee et al. 2016) or those maximising the probability of target behaviour (Solway et al. 2014) or those that form a low-rank approximation of possible tasks (Earle et al. 2017).

However, hierarchical RL (and other hierarchical approaches like Saxe et al. 2017) only partially exploits hierarchical problem structure. It splits up a big problem into smaller problems, but solves each small problem separately. In the rooms example, each room requires learning its own policy. A far more powerful approach generalises across rooms, to realise that the same low-level structure is repeated in a high-level pattern. Then learning in one room immediately applies to every room. Here, I will present an extension of the model of Section 3.1 that incorporates hierarchical representations. Through the same principle of object vector representations, now at different hierarchical levels, this model generalises across the hierarchy within an environment, and across different

hierarchical environments. This is achieved by separating representations between levels while reusing them within levels. Object vector representations are tied to locations in memory on each level of the hierarchy, both during behaviour and replay.

3.2.1 Methods

Environment structure and representation

The environments that the model interacts with consist of deterministic MDPs, as in the discrete model of Section 3.1.1, nested hierarchically. As a recurring example I will use a building-with-rooms environment, illustrated in Figure 3.12. At the low level of the hierarchy this environment consists of square grid rooms, that connect together into a high-level loop. Or, in a top-down rather than bottom-up view, it is a loop environment, where each loop location itself is a square grid room. Actions and locations now exist on both levels of the hierarchy. On the low level, within-room actions on the square grid ('go-north', 'go-east', 'go-south', 'go-west') transition between the locations within the room. On the high level, between-room actions on the loop ('clockwise' and 'anticlockwise') transition between rooms. These between-room transitions occur when the agent encounters a 'door' object within a room, which takes them to an adjacent room. Finally, there is a reward object located in one of the rooms.

Learning a policy for any environment

Given hierarchical environments defined like this, what does representation-action mapping f look like? Its output $\vec{a}$ remains the same: a probability distribution over low-level (or *primitive*) actions. Although hierarchical environments define actions on each level, the agent only has access to the low-level actions. The representation input $\vec{s}$ of f contains object vector representations on all levels. In the building-with-rooms environment, $\vec{s}$ concatenates the high-level reward-room vector representation $\vec{s}^{\mathrm{H}}_{\mathrm{R}}$ and the low-level vector representations for reward $\vec{s}^{\mathrm{L}}_{\mathrm{R}}$, the clockwise door $\vec{s}^{\mathrm{L}}_{CW}$, and the anticlockwise

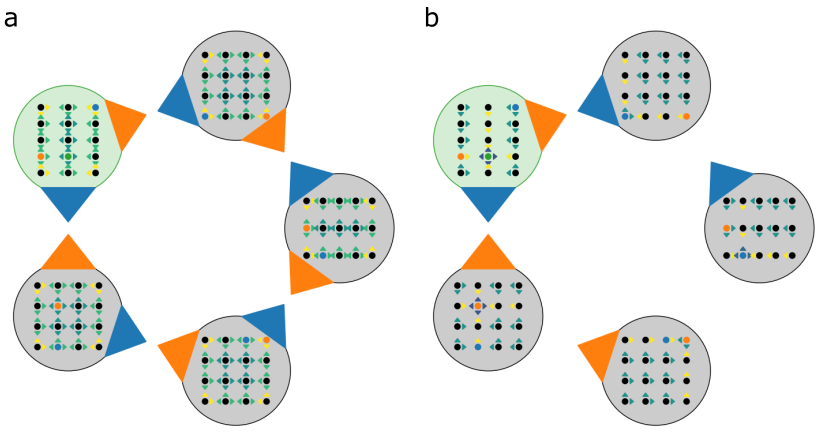


Figure 3.12: Hierarchical rooms. a. Example hierarchical rooms environment, consisting of square grid graph rooms connected together into a high-level loop. Low-level locations are plotted as small circles, and low-level actions as small triangles, with action colour plotting the policy – here for a random policy, where each available action has equal probability. There are three types of low-level objects: a green reward, only in the room at 10 o'clock, and a blue and an orange door in each room. High-level locations are plotted as large circles in grey, except for the reward-containing room in green, connected by high-level actions: blue and orange triangles, corresponding to the low-level doors. Navigating to a low-level orange door results in the high-level orange action, to transition to a different room. b. Optimal policy towards reward in *a*'s hierarchical rooms environment. I only plot low-level actions with non-zero probability under the optimal policy. Optimal low-level actions lead towards the door object towards the high-level room nearest to reward; or directly to reward, if already in the reward-containing room. For illustration purposes, to make it easier to see the correct low-level door in each room, I plot only the high-level actions in the optimal high-level direction, but it is important to realise that the optimal policy (and any policy) is only defined for low-level actions.

door $\vec{s}^{\,L}_{ACW}$ object. Importantly, while object vector representations separate between levels, they generalise within hierarchical levels: the same $\vec{s}^{\,L}_{CW}$ represents a location two steps to the east of the clockwise door in each room. I calculate object vector representations as detailed in Section 3.1.1, so that on the high-level loop the representation contains distances along both clockwise and anticlockwise actions.

Intuitively, the full representation $\vec{s}$ provides all information required for optimal actions towards reward: from the high-level $\vec{s}^{\,H}_{R}$ the agent knows if the reward room is closest in clockwise or anticlockwise direction, and from $\vec{s}^{\,L}_{CW}$ and $\vec{s}^{\,L}_{ACW}$ they know how to get to the appropriate door – or how to get to reward from $\vec{s}^{\,L}_{R}$ if already in the room with reward. I train f like the discrete

model of Section 3.1.1, through supervised learning with training examples $\vec{s}$, $\vec{a}$ sampled from multiple environments in parallel.

Building a map of the current environment

I make two important assumptions about higher levels of the hierarchy, so that I can apply the same mechanisms of path integration and memory as in Section 3.1. First, I assume that just like the low-level location can be observed from the environment, higher-level locations are observable too. The agent knows from observation which room they are in – but not how far away that room is from the rewarding room. Second, I assume that the agent knows the high-level action they have just taken on a high-level transition, like they do on the lowest level. Within a room they know when they have taken 'go-north' action; similarly, between rooms they know whether they went clockwise or anticlockwise.

The second assumption is required for path integration on all levels. At the start of exploration, the agent does not know any of the vector representations that constitute $\vec{s}$. As they move around, they encounter objects like doors and rewards and initialise the corresponding object vector representations. Finding a reward initialises the object vector representation both within that room, and the higher level room vector representation between rooms. Within the room the agent updates their within-room, lowest-level representations through path integration as before. When they traverse a door and transition to the adjacent room, they update their between-room, high-level vector representation according to the corresponding high-level action. On arrival in the new room the low-level object vector representations are reset. If it is the first time the agent encounters this new room, they will have to discover the door configuration of this room to construct $\vec{s}$. If they have been in this room before, they can retrieve their new $\vec{s}$ from memory.

Such memory retrieval, and the preceding memory encoding, require observable locations at all levels, as per the first assumption. The hierarchical

model has a memory bank within each level, and an additional set of memories that links subsequent hierarchical levels. The within-level memories tie object vector representations to locations, just like in the original model of Section 3.1. Specifically, the low level memories $M^L = \{M^L_i, ...\}$, where i counts rooms, contain memories $M^L_i = \{M^L_i(s_j), ...\}$, where j counts locations in that room, which in turn contain memories $M^L_i(s_j) = [s_k, ...]$ of representations s_k (object vectors for doors and rewards) previously encountered at within-room location s_j. The high level memories $M^H = \{M^H(r_i), ...\}$, where i counts rooms, contain memories $M^H(r_i) = [r_k, ...]$ of representations r_k (room vectors towards the reward room) previously encountered at room r_i. Simply put: low-level memory stores at each within-room location the associated within-room object vector representations, and high-level memory stores for each room the associated between-room room vector representations. These memories need locations as indices, and therefore I assume observable locations across hierarchical levels.

But in addition to the within-level memories, the model has an additional memory bank for between-level memories. The reason for these additional memories is that *a priori* an agent cannot know where a door will take them. When the agent discovers the first door, they keep track of its within-room object vector representation through path integration. But they only know whether this is a clockwise or anticlockwise door on the between-room level after traversing it. In order to feed into f it is necessary to know which door is which: f expects object vector representations in order, so that when the rewarding room is in clockwise between-room direction, the agent navigates to the clockwise door within-room. Therefore on a between-room transition from room r_i, they encode a memory $M^{HL}(r_i) = \{a, o\}$ that records which low-level object corresponds o to that high-level action a.

The role of replay, as before, is to construct the cognitive map in memory by tying representations to locations. Replay operates separately on each level of the hierarchy. On the low level, replay path integrates within-room object vector representations and ties them to within-room locations, populating M^L_i

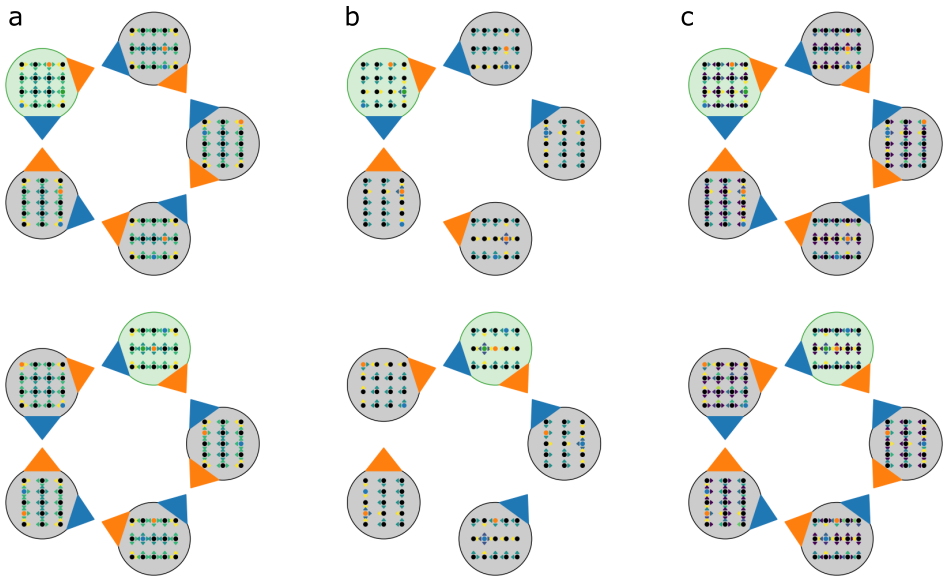


Figure 3.13: Hierarchical policy. a. Two example hierarchical room environments (top and bottom row), with actions plotted for a random policy. b. Optimal policy in hierarchical room environments. For visualisation, I only plot high-level actions in the correct high-level direction, but policies are only defined over low-level actions. c. Learned policy on hierarchical room environments, obtained from $f(\vec{s}) = \vec{a}$ after training f on many such environments in parallel.

in room i. On the high level, replay populates M^H by path integrating across rooms, for example to tie reward room vector to remote rooms without physically navigating there.

3.2.2 Results

Hierarchical object vector policies generalise across rooms

Provided with a hierarchical object vector representation $\vec{s}$, mapping $f(\vec{s}) = \vec{a}$ provided optimal primitive actions (Figure 3.13). I train f on different hierarchical environments, all consisting of low-level grid graph rooms connected together on a high-level loop, with different room dimensions and object locations across environments. Optimal actions then take the agent to the appropriate low-level door, that leads towards the room that is nearest to the room with reward – or immediately to the reward if already in that room.

One high-level replay updates all low-level actions

During exploration and replay, the agent encodes within-room and between-room object vector representations in memory. They replay the different hierarchical levels separately. High-level, across-room replay can be extremely powerful to immediately update policies in locations that would be very far away via primitive actions. If clockwise and counterclockwise within-room door locations are known for rooms far away, and the agent finds a reward in the current room, it only takes a single across-room replay sequence to assign appropriate between-room vectors to remote rooms in memory – and when the agent returns to that room, f produces optimal within-room policies from the full $\vec{s}$ retrieved from memory.

I demonstrate how high-level replay immediately changes low-level policy in the situation where one pair of doors gets blocked, turning the high-level loop into a high-level line (Figure 3.14). Before the blockade is discovered, the optimal policy leads towards whichever door is closest to the reward in clockwise or anticlockwise direction along the loop. Then, when the agent encounters the blockade, the optimal policy changes dramatically. All actions leading to doors in the direction of the blockade are no longer optimal; instead they should guide the agent towards the opposite door. A single high-level replay sequence carried the blockade vector representation to every room, yielding immediately updated optimal policies in each.

3.2.3 Discussion

I have extended the model of Section 3.1 to employ hierarchical object vector representations. Hierarchy is powerful in general to split up big problems into smaller, more tractable subproblems, but the particular implementation here goes beyond that. I separate object vector representations between hierarchical levels, and generalise within hierarchical levels. In the recurring example of rooms-on-a-loop here, that means that there is a low-level within-room object

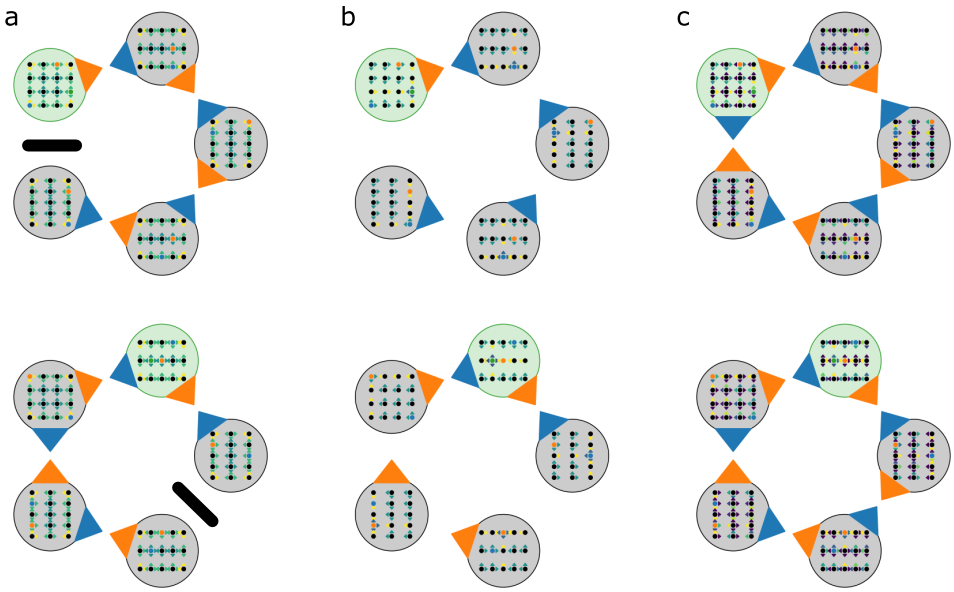


Figure 3.14: Hierarchical replay. a. The same hierarchical rooms environments as in Figure 3.12, but now I introduce a blockade between one pair of rooms, which disables the doors between that pair. b. The optimal policy changes drastically: in the top row environment, optimal actions in the 8 o'clock and 6 o'clock rooms now lead towards the blue door instead of the orange door, and vice versa in the 6 o'clock room of the bottom row environment. c. After discovering the high-level blockade, a single replay that propagates the high-level blockade object vector representation to other high-level rooms suffices to immediately change the learned policy in each room.

vector representation that is reused across rooms, and a high-level between-room object vector representation. Such representations have three important advantages. First, they turn what would be an enormous state space when considering all lowest-level locations individually into something much smaller. The dimension of representations grows as the sum of the number of locations of a typical unit on each level instead of the product. Second, because the same representation is shared within hierarchical layers, policies generalise. What is learned in one room directly applies in a different room. Third, separating representations between hierarchical layers allows high-level updates to immediately affect all low-level locations. The agent can build up a complete low-level object-vector map of both doors for all locations in a room, and then a single change in high-level between-room representation provides optimal policies in the whole room.

The same problem definition here used for hierarchically structured environments could also accommodate sequential tasks. For example, consider a task where an animal needs to first press a lever in one corner of an arena, then in another corner, before receiving reward in the center, and going back to the first corner again (a bit like Boccara et al. 2019) – or a barista preparing a big order of cappuccinos. The essence of these tasks is identical to the rooms-on-a-loop example, except that each element of the loop is now an of identical copy of the same room. The high-level representation of the hierarchy tracks the current progress on the sequence, the low-level the object vector representations.

On the lowest level of the hierarchy, assuming observable actions and locations is justifiable. Agents know how they act on the environment, and the environment usually provides enough clues as to where they are. However, on higher levels of the hierarchy, such assumptions are not as easily justified. For example, in sequential tasks as just mentioned, the environment does not necessarily change at every step, turning high-level location unobservable. Similarly, it is not always obvious how high-level actions transition high-level locations. In general high-level configurations, the agent would have to understand an action in terms of its consequences. I will return to this problem of action discovery from state space structure in Section 3.4.

3.3 Action prediction yields vector representations

While looking for neural substrates of the cognitive map in the brain, neuroscientists have found a variety of spatially selective cells in the rodent hippocampal-entorhinal formation. Examples include place cells, grid cells, border cells, object vector cells, and reward cells, and it is not unlikely that more discoveries are on the way. The fact that these different cell types exist poses the question whether each is a phenomenon on its own, or whether they emerge from the same underlying principle. Is there a common mechanism that explains the

function of the variety of spatially tuned cells, and how these different representations are learned?

The Tolman Eichenbaum Machine (TEM) proposes sensory prediction combined with generalisation as such a mechanism. TEM is an artificial neural network model trained to predict upcoming sensory stimuli associated with locations on a graph that the model navigates. Provided with sequences of stimuli and the actions taken for graph transitions, TEM builds a map of stimulus locations. Importantly, that requires tracking a *latent state*: since multiple locations may have identical stimuli (called *aliasing* (George et al. 2021)), TEM cannot rely on sensory information alone to know where it is on the map. To achieve this, TEM learns a representation of abstracted location in medial entorhinal cortex, and binds that through conjunction in an attractor network in hippocampus to sensory stimuli from lateral entorhinal cortex. After training on stimulus prediction, TEM develops grid-like firing patterns in entorhinal cortex and place cells in hippocampus. Place cells and grid cells, but also splitter cells and lap cells, are examples of latent state representations: they disambiguate states with similar sensory experience but different futures.

TEM's separation (or *factorisation*) of abstract location and sensory experience additionally affords generalisation of structural knowledge to new environments. In new environments with similar structure but different stimuli, the entorhinal representation of abstract location simply needs to be tied to new sensory experiences in hippocampus – but prior knowledge about how to update (or *path integrate*) abstract locations still applies. For even greater generalisation, maps of abstract locations themselves can be composed from common structural elements (or *bases*) across tasks. Grid cells represent a basis of two-dimensional space, but other basis representations include object-vector cells, border-vector cells, and goal-direction cells. Because structural bases imply behaviour and behaviour determines sensory experience, TEM learns such representations as part of sensory prediction.

Can the same TEM framework provide a mechanistic account of learning and using object vector representations in replay, as required for the model of Section 3.1? Here, I implement the conceptual principles of the model, namely maps of locations and of object vector representations that predict actions, tied together in memory, in an actual neural learning system. I demonstrate how two adaptations to TEM produce two results that underlie the ideas of Section 3.1. First, I train TEM to predict *actions* in addition to observations, and second, I train TEM not just while exploring an environment but also in *replay* episodes. A replay episode involves finding reward, then replaying to a remote location, and then teleporting to that remote location and navigating to reward. I find that first, replay is sufficient for TEM to learn object vector representations, and second, replay allows TEM to accurately predict actions towards the reward location from remote locations.

3.3.1 Methods

I build on the existing Tolman Eichenbaum machine (TEM), described extensively in the supplementary material of Whittington et al. 2020. I will not describe the model in full implementational detail, but I do need to introduce some of its notation and terminology here, illustrated in Figure 3.15, so that I can refer to it later. The medial entorhinal (grid) representations of abstract location are denoted by $\vec{g}$, the lateral entorhinal (sensory) representations of observation input by $\vec{x}$, and the hippocampal (place) representations of the conjunction of these two by $\vec{p}$ (Figure 3.15b). At every step, TEM receives a location's stimulus $\vec{x}_t$ and the action a that took it there. To figure out its new abstract location, encoded in $\vec{g}_t$, TEM relies on a generative and an inference model (Figure 3.15c). The generative model uses a to path integrate previous grid representation $\vec{g}_{t-1}$ to obtain $\vec{g}_{gen}$. The inference model uses $\vec{x}_t$ to retrieve previously encountered grid representations for that stimulus from memory. Memory M is implemented as Hebbian weights between hippocampal cells, so that a conjunction $\vec{p}$ can be retrieved from its constituents through attractor dynamics. The inference

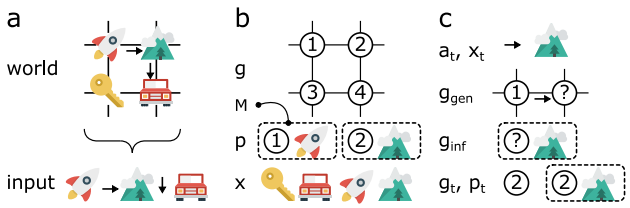


Figure 3.15: Notation and terminology. a. TEM experiences environments with an observation at each location. A walk in such an environment produces a series of observations and actions; TEM receives the observation and action as input, and predicts the next observation. b. The medial entorhinal code $\vec{g}$ represents the abstract location, the lateral entorhinal code $\vec{x}$ represents the sensory observation. Hippocampal place cells $\vec{p}$ form a conjunctive memory that binds observations to abstract locations; this memory is stored in weights M of an attractor network. c. To figure out its abstract location, TEM relies on a generative and an inference model. As input, TEM receives an action a_t and observation x_t. The generative uses the action to update the previous location through path integration. The inference model uses the observation to retrieve locations associated with that observation. TEM combines g_{gen} and g_{inf} to estimate the current location g_t, and creates a new conjunctive memory p_t.

model thus pattern completes the current observation $\vec{x}_t$ to retrieve $\vec{p}_x$ from memory, which provides the grid representation $\vec{g}_{inf}$ part of the conjunction $\vec{p}_x$. Finally, TEM combines the generative path-integrated and inference pattern-completed grid representations $\vec{g}_{gen}$ and $\vec{g}_{inf}$ into current grid representation $\vec{g}_t$. The corresponding place representation $\vec{p}_t$ is the conjunction of grid $\vec{g}_t$ and sensory $\vec{x}_t$ codes, which is added to memory M.

Object vector cells for action prediction

So how are object vector cells incorporated? Place and grid cells are organised into modules to allow for representations at different spatial and temporal scales. TEM implements object vector cells as just another grid module, and I will do the same. More specifically, $\vec{g}$ then contains grid cells and object vector cells, and therefore $\vec{p}$ contains the corresponding conjunctions with sensory input $\vec{x}$: place cells and *landmark* cells, respectively. Path integration weights between grid modules and object vector cell modules are set to zero, so that they update independently. For memory retrieval, connections between place and landmark modules are allowed, so that grid and object vector representations can retrieve each other. I make one important addition to TEM's original

object vector cell implementation: at every step, I multiply the object vector part of $\vec{g}$ by a matrix of learnable weights to predict the upcoming action at the next step, and I introduce another set of weights to predict reward.

The optimisation of TEM's parameters over training remains largely unchanged compared to Whittington et al. 2020. At every step, I calculate a number of losses as in original TEM to minimise the mismatch between inferred and generated variables, and the error between observations and predictions. I introduce two additional prediction errors: a cross-entropy loss between the predicted and taken action, and a cross-entropy loss between predicted and observed reward signal. I then collect the losses over a block of n_{bp} time steps, and backpropagate losses through time to update all parameters along the gradients that minimise the loss. TEM is trained on batches of n_{env} multiple environments in parallel, and optimised using the ADAM optimiser with a decaying learning rate. Apart from the additional prediction losses, I do not change details and parameters of neural network fitting from the original procedures.

Backpropagation through replay

What I do change, however, is the task that TEM is trained on. I alternate backpropagation blocks of exploration, where TEM follows a random policy on a graph, with backpropagation blocks of replay (Figure 3.16). By backpropagating through a replay episode, TEM learns to employ replay to generate representations and memories that support prediction. A replay block consists of three phases. First, TEM continues to randomly explore for n_e steps, and discovers reward on the final exploration step. Then, TEM replays towards a remote location for n_r steps, and encodes a new memory there on the final replay step. Finally, TEM is teleported to that remote location and follows a goal-directed policy for n_g steps until it arrives at the reward location, where it remains. In order to successfully predict actions towards reward during goal-directed behaviour, TEM needs to initialise an object vector representation on reward discovery, path integrate that representation during replay, encode the

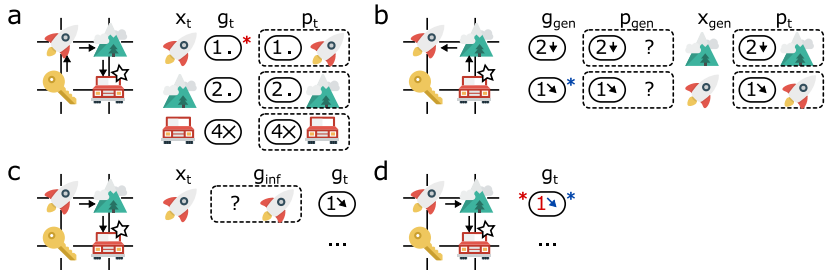


Figure 3.16: Replay backpropagation block. a. A replay path integration block starts with exploration, with standard TEM behaviour. At each step TEM x_t represents the observation, g_t the abstract location, and p_t the conjunctive memory. The object vector part of g_t is initially marked with ., because before finding reward, the object vector code is unknown. Then at the final step of exploration, TEM finds reward at the car observation, and initialises the object vector code as centred on the current location. b. During replay, TEM imagines a walk through the environment. Since it does not receive observation input, it needs to rely on its generative model. At each step, TEM path integrates the location and object vector code g_{gen}, then retrieves the observation x_{gen} for that imagined location from memory p_{gen}, and creates a new memory that includes the object vector code. c. After replay, TEM is teleported from the final explored location (the car) to the final replayed location (the rocket). It then navigates towards the goal location, while predicting actions and observations. To make such predictions, TEM needs access to the object vector code. Therefore on the very first step, it relies on its inference model to retrieve the object vector code from the memory that was created on the final step of replay. d. In practice, I take a shortcut around the memory retrieval of c: I copy in the abstract location from the initial step of a, and the object vector code at the final step of b, as the initial g_t after replay.

representation in memory at the final replay step, retrieve it after teleportation, and path integrate it until arriving at reward.

I will now describe the implementation of each phase of the replay block in more detail. The initial exploration (Figure 3.16a) is identical to exploration the original TEM framework, in which it receives each step's observation and action, and I additionally provide a binary reward signal at every step. I make sure that reward is encountered for the very first time at the end of exploration, so that reward information can only reach other locations through replay.

Then replay starts (Figure 3.16b). During replay, the only input is the imagined replay action without any sensory experience, so TEM needs to rely only on its generative model. It path integrates to update $\vec{g}_{gen}$, which includes grid and object vector representations, from the previous step's $\vec{g}$. Given grid code $\vec{g}_{gen}$, it can pattern complete to obtain place representation $\vec{p}_{gen}$ from

memory, and predicted sensory experience $\vec{x}_{gen}$ at the imagined location from that. In the final replay step, TEM encodes a new memory that ties the path integrated $\vec{g}$, which includes the reward-centred object vector representation, to the remote location. In principle I could use the conjunction of $\vec{x}_{gen}$ and $\vec{g}_{gen}$ for this purpose. If TEM accurately path integrates during replay, it can imagine the sensory stimulus $\vec{x}_{gen}$ at the final location for memory encoding – and when it returns to that location and observes the stimulus, it can use that stimulus to retrieve the replayed $\vec{g}_{gen}$ from memory (Figure 3.16c). However, in practice I take a shortcut that achieves the same result (Figure 3.16d). After teleportation, I initialise the *grid* modules of $\vec{g}$ to be the last grid representation recorded at that location, and the *object vector* modules of $\vec{g}$ to be the representation at the final step of replay.

Equipped with the object vector representation obtained through replay, TEM then navigates towards the reward location. This final phase produces the crucial teaching signals for learning how to use object vector representations and replay. I therefore repeat the post-replay goal-directed behaviour n_n times, teleporting TEM back to the final replay location each time after arriving at reward. To avoid early iterations from influencing subsequent ones, I do not create any new memories in this phase. Additionally, TEM uses different path integration weights to update $\vec{g}$ without action input, because now that behaviour is goal-directed, actions are predictable – that is, with appropriate replay and representations. To further encourage TEM to develop representations that predict actions and path integrate, I switch off the inference model for updating $\vec{g}$ in the final phase.

Table Table 3.2 lists all parameters that appeared in this Section with a short description and default values.

Name	Value	Description
n_{bp}	50	Number of time steps for backpropagation through time
n_{env}	16	Number of environments in batch
n_e	12	Number of explore steps in replay backpropagation block
n_r	12	Number of replay steps in replay backpropagation block
n_g	7	Number of steps in goal directed walk of replay backpropagation block
n_n	5	Number of goal-directed walks in replay backpropagation block

Table 3.2: Model parameters. List of parameters that appear in Section 3.3.

3.3.2 Results

Object vector cells are learned from replay

To investigate the representations that TEM learns for prediction in goal-directed behaviour after replay, I plot rate maps for TEM's artifical neurons in different environments in Figure 3.17. These rate maps revealed object vector cells in entorhinal cortex (Figure 3.17a,c) and landmark cells that respond to one particular object in hippocampus (Figure 3.17b,d). I generate ratemaps by scattering between two and four rewards in different environments, and then having a trained instance of TEM explore around the rewards in these environments while I record neuronal activity at each step. For each neuron I average its activity across all visits to a location. The extended goal-directed behaviour around rewards during recording serves to measure stable rate maps; during training, reward is only encountered as part of replay blocks. The object vector representations can therefore only result from replay training.

Object vector cells path integrate

Figure 3.18 illustrates how the full representation across the population of grid cells and object vector cells evolves over an example replay block. TEM path integrated object vector representations during replay and subsequent goal-directed behaviour to support prediction after reward discovery. Before finding reward, the object vector cells retain their initial state. Then, upon reward discovery, the object vector representation initialises (Figure 3.18a, red represen-

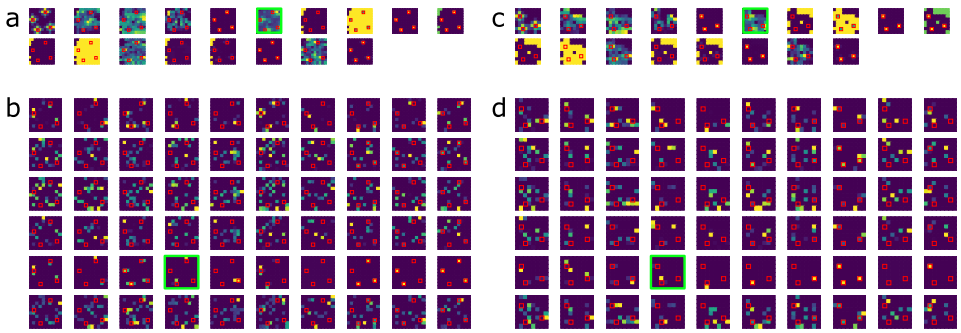


Figure 3.17: Object vector cells. a. Ratemaps of artificial entorhinal object vector cells in TEM after training on action prediction through replay, in an square grid environment with four reward objects at location with red outlines. The example cell with green outline has a firing field to the east of each object. b. Ratemaps of artificial hippocampal landmark cells in TEM. Landmark cells are conjunctions of object vector cells and observation codes, so they are more specific than object vector cells. The example cell with green outline has a firing field to the north of two out of the four rewards. c. Like *a*, but in a different environment, now with three rewards. The same entorhinal object vector cell, again with green outline, also fires to the east of the objects in this environment. d. Like *b*, but in *c*'s environment. The same hippocampal landmark cell now has a firing field to the north of one out of three rewards.

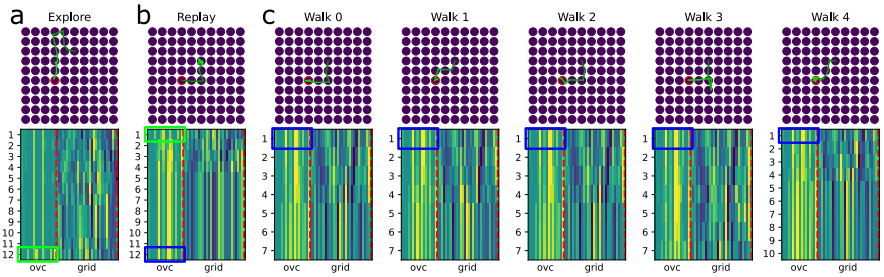


Figure 3.18: Path integration. a. Behaviour (top row) and entorhinal representations (bottom row) during a replay block. TEM explores around the environment (green walk) and finds the reward on the final step (red outline location). On reward discovery, TEM initialises the object vector representation (green outline representation). b. During replay, TEM imagines navigating to a remote location (green walk) while path integrating the grid and object vector representations. The final representation (with blue outline) is encoded in memory. c. After replay, TEM starts from the final replay location and retrieves the corresponding representation (blue outline) from the memory encoded in replay. Then follow multiple repeats of goal-directed behaviour in which TEM predicts action and reward signal.

tation). The object vector representation is then path integrated in replay to a remote location, and the resulting representation (Figure 3.18b, blue representation) retrieved after teleportation on the first step of goal-directed behaviour.

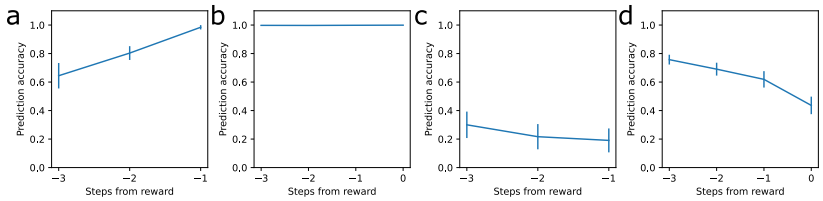


Figure 3.19: Prediction accuracy. a. Direction prediction accuracy ± s.e.m. in steps leading up to reward during goal-directed behaviour after replay. The closer to reward, the more predictable the direction is, until only one action leads to reward at only one step away. b. Reward signal prediction accuracy ± s.e.m. in steps leading up to reward. Both absence and presence of reward are predicted accurately. c. Like *a*, but without replay. Direction prediction accuracy ± s.e.m. during goal-directed behaviour from a remote location without prior replay. d. Like *b*, but without replay. Reward signal prediction accuracy ± s.e.m. during goal-directed behaviour from a remote location without prior replay.

Replay supports action and reward prediction

Having learned these representations, TEM accurately predicted actions and reward signal in replay blocks, but only when using replay (Figure 3.19). I align the post-replay goal-directed behaviour across environments so that the arrival at reward occurs at the same time in each of them. The action prediction accuracy peaks at the step right before arrival at reward (Figure 3.19a). That is expected: in many situations on a square grid there are two possible optimal actions towards reward, but at the final step there is only one. The reward signal prediction accuracy is high throughout (Figure 3.19b), which means that TEM accurately predicts both absence and presence of reward. When replay is disabled by discarding the representation at the final replay step instead of retrieving it after teleportation, both the action and reward signal prediction drop to chance level (Figure 3.19c-d).

3.3.3 Discussion

The Tolman Eichenbaum Machine provided a starting point to implement the ideas of Section 3.1 in a learning system. I presented how extending TEM for action and reward prediction combined with backpropagation through replay supported essential components of the model in Section 3.1. In particular, I

showed that the object vector representations that the model heavily relies upon can be learned from replay alone. These object vector representations predict goal-directed action and reward signal, but the right object vectors need to be available at the right places. Replay achieves such map composition by tying object vector representations to the current arena. Grid representations map out general two-dimensional space; object vector representations implement object-centred coordinates; constructing a particular environment is like shifting the two coordinate systems on top of another and binding them together at the appropriate locations.

Here, the composition of the two coordinate systems into one map takes a bit of a detour: through sensory experience. The entorhinal representation essentially concatenates grid and object vector representations. This combined representation is tied to sensory observation by conjunction in hippocampus. For grid representations the conjunction produces place cells, and landmark cells for object vector representations. In Section 3.1.3 I proposed a different architecture: one conjunction of grid and sensory codes, and another of grid and (one or multiple) object vector codes. The first conjunction thus maps out the sensory aspects of the different coordinates, while the second conjunction flexibly combines separate coordinate systems into an abstract map, that does not involve sensory experience. That implementation explicitly codes the composition of coordinate systems through conjunction, instead of indirectly via sensory experience, and does not have aliasing issues. It would therefore likely be preferable, but requires greater adaptations to the original TEM framework as used in the proof-of-principle here.

It is important to realise that despite the presence of rewards and goal-directed behaviour in this section, there is no actual reinforcement learning taking place. TEM receives sensory stimuli and actions as input, and learns by comparing predicted to observed stimuli and actions. There is no decision making within TEM, and no evaluation of long-term reward expectations. A mechanism as presented here would require a separate reinforcement learning

module that through experience of reward signal produces reward-optimising actions, such as striatal temporal difference learning. After many instances of successful goal-directed behaviour, achieved through reinforcement learning, TEM's predicted actions can provide optimal actions instead. This presents a complementary approach to the reinforcement learning problem: instead of learning what do in a situation from reward, TEM learns to predict what it would *usually* do in situations *like the current one* – and the representations that support making such predictions.

3.4 Action embedding from structure

During navigation of physical space, it is usually obvious what an action is and does. The outcome of a step in a certain direction, which can be determined from environmental and proprioceptive cues and encoded in head direction cells, is a new location along that direction. Such action outcomes are consistent across the environment – a step to the east produces the same location update, whatever the starting location. The models presented in this Chapter heavily rely on consistent, observable actions. To allow for path integration, the models need to receive labelled actions (such as 'go-north' or 'go-clockwise') with consistent outcomes and learn how actions relate to each other (like 'go-north' does the opposite of 'go-south'). However, in tasks that play out in abstract state spaces rather than physical space, these action labels cannot be observed directly from the environment. The physical action in such tasks, like a button press or a nose poke, are not informative of the abstract actions that determine state transitions. For example, in Samborska et al. 2021 mice perform a reversal learning task that defines a abstract task space, but the physical action (the nose poke port) corresponding to task state transitions (for example, the initiation of a trial) varies. Similarly, human participants in Mark et al. 2020 navigate abstract graphs by selecting stimuli on a screen. Abstract actions can only be understood from their state-space consequences. Is it possible to find out how such actions relate to each other from purely structural considerations?

Here, I present a method of *action embedding* that assigns low-dimensional vector representations to actions that express their structural consequences on state spaces. It is driven by the intuition that certain actions do something similar across state space, like going 'north' on a grid or going 'clockwise' on a ring, but extends this idea to arbitrary structures. More precisely, embedding actions by their structural consequences means that actions that have different outcomes, should be represented as differently as possible. At the same time, for consistent representations across state space long-distance navigation should repeat similar actions. I quantify these two desired properties of action representations, and cast them in an optimisation framework to find *those* action embeddings that maximally satisfy short-term discrimination and long-term consistence.

3.4.1 Methods

I consider deterministic MDPs as task state spaces, so each action leads to a certain state with probability 1. Then the state space can be represented by a graph where each node is a state and each edge is an action. Each action a is embedded by a D-dimensional unit norm vector $\vec{v}_a$, whose entries are determined by optimising a cost function that trades off *controllability* and *simplicity*. I will further define them below, but the intuition is as follows. To maximise controllability, actions that have different consequences should be represented as differently as possible. To maximise simplicity, action along paths between states should be as represented as similarly as possible.

Action SR quantifies action consequences

To quantify structural consequences of actions I define an action successor representation (action SR), analogous to the well-known successor representation for states. It calculates the expected discounted number of times that action a' is taken starting from action a while following policy π:

$$N(a, a') = E[\Sigma_{t=0}^{\infty} \gamma^t \delta_{a_t, a'} | a_0 = a] \tag{3.13}$$

where γ is the discount factor, and $\delta_{i,j} = 1$ if $i = j$, 0 else. It can be calculated directly from

$$N = (I - \gamma T)^{-1}, \quad T(a, a') = \Sigma_s \Sigma_{s'} p(s|a) p(s'|s, a) \pi(a'|s) \tag{3.14}$$

where T is the action transition matrix for policy π. Then the ath row of N, $N(a, :)$, contains how often each action follows a (Figure 3.20a), and the ath column of N, $N(:, a)$, contains how often each action precedes a. The discounted number of times action i follows from state s (Figure 3.20b) is obtained by summing over the actions available in state s:

$$\Psi_s(i) = \Sigma_{\{a \in \mathcal{A} | p(s|a) > 0\}} N(a, i) \tag{3.15}$$

and similarly, the discounted number of times action i precedes state g (Figure 3.20c) is obtained by summing over the actions leading to state s':

$$\Phi_g(i) = \Sigma_{\{a \in \mathcal{A} | (\exists s | p(g|a,s) > 0)\}} N(i, a) \tag{3.16}$$

These definitions allow for the calculation of a measure of 'on-pathness' for action i given a start state s and and goal state g from the product $\Psi_s(i)\Phi_g(i)$, expressing how often that action is on a path between s and g (Figure 3.20d-e). Clearly N, and therefore Ψ and Φ, depends on the policy. I will assume a random policy, so that

$$\pi(a|s) = \frac{1}{|\mathcal{A}(s)|} \tag{3.17}$$

Different actions for different outcomes

The first term in my action embedding cost function is controllability. Controllability implements the idea that for a given combination of start and goal state, high control corresponds to a clear difference between actions toward that goal versus actions away from that goal. For actions embedded by normalised vectors, this similarity can be expressed by the dot product: similar (parallel) vectors have a high dot product, $\vec{v}_1 \cdot \vec{v}_2 \approx 1$, while dissimilar (antiparallel) vectors

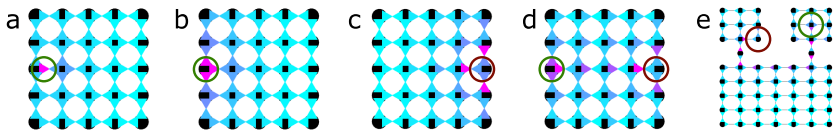


Figure 3.20: Action SR. a. Square grid graph with states as black circles and actions as triangles, with colour reflecting their action SR $N(i,:)$ with action i highlighted by the green circle. b. Action colour now indicates Ψ_s: how often an action follows from state s highlighted by the green circle. c. Action precedence Φ_g, for state g highlighted by the red circle. d. On-pathness $\Psi_s\Phi_g$, with s the state in the green circle and g the state in the red circle. e. On-pathness for start state s in green and goal state g in red for a more complex graph of two grid rooms connected by a central grid room.

have a low dot product, $\vec{v}_1 \cdot \vec{v}_2 \approx -1$. Therefore, for each pair of start state s and goal state g, the controllability is maximised by minimising the dot products of action pairs at state s, weighted by the importance of that pair being different:

$$C(s,g) = -\Sigma_{\{a_1,a_2\},a_i \in \mathcal{A}(s)} \Delta^g_{a_1,a_2} \cdot (\vec{v}_{a_1} \cdot \vec{v}_{a_2}) \tag{3.18}$$

where the importance $\Delta^g_{a_1,a_2}$ of action a_1 and a_2 being different is determined by there absolute difference in Φ_g:

$$\Delta^g_{a_i,a_j} = |\Phi_g(i) - \Phi_g(j)| \tag{3.19}$$

So the importance between a_1 and a_2 is large when a_1 is much more likely to be on a path from s to g than a_2 (or vice versa). That means a_1 and a_2 have different consequences, and therefore for optimal controllability their corresponding vectors should be highly anti-parallel –- so a strongly negative dot product. At the same time, the importance between a_1 and a_2 is small when both are about equally likely to be on a path from s to g. In that case both actions have similar consequences, and and so the dot product between their corresponding vectors need not be negative. The total controllability sums Equation (3.18) across all pairs of start-goal states:

$$C = \Sigma_{s \in \mathcal{S}} \Sigma_{g \neq s \in \mathcal{S}} C(s,g) \tag{3.20}$$

Long-range action consistency

The second term in the cost function is simplicity, to implement large-scale consistency. Simplicity encourages paths between distant states to have simple descriptions in terms of action sequences; here 'simple' means 'consisting of similar actions'. Because I use normalised action vectors, I can evaluate the norm of the vector sum of actions along a path from start state s and goal state g. If the vectors corresponding to actions on that path are mostly have parallel orientation, the norm of the sum vector will be larger than for divergent vectors. The simplicity for a pair of start state s and goal state g is thus computed by adding up all action vectors weighted by their 'on-pathness', and calculating the norm of the resulting vector:

$$S(s, g) = |\Sigma_{i=1}^{|\mathcal{A}|} \Psi_s(i)\Phi_g(i) \cdot \vec{v}_{a_i}| \tag{3.21}$$

where $\Psi_s(i)\Phi_g(i)$ measures on-pathness as described before. This definition has full rotational invariance: as long as vectors along the path are parallel, the simplicity is maximised, whether those vectors (in a two-dimensional example) are all $[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]$ or $[1, 0]$. In order to separate features of the state space structure into separate dimensions, which makes it easier to interpret action vectors, I actually calculate the *maximum entry* of the sum vector instead of its norm for simplicity:

$$S(s, g) = \log(\Sigma_{d=1}^{D} \exp\{\text{abs}[\Sigma_{i=1}^{|\mathcal{A}|} \Psi_s(i)\Phi_g(i) \cdot \vec{v}_{a_i}]_d\}) \tag{3.22}$$

The logarithm of the exponential of the sum vector implements a 'smooth max' function, that approximately return the largest vector entry but in a differentiable way. The total simplicity sums Equation (3.22) by summing across all start and goal state pairs:

$$S = \Sigma_{s \in \mathcal{S}} \Sigma_{g \neq s \in \mathcal{S}} S(s, g) \tag{3.23}$$

Having defined controllability in Equation (3.20) and simplicity in Equation (3.23), I add the two together for the full cost function that I will minimise to obtain optimal action embeddings:

$$S = -(C + \alpha S) \tag{3.24}$$

were α weights controllability against simplicity. Controllability and simplicity pulls representation in different directions: controllability encourages *differences*, simplicity *similarities*. The cost function of Equation (3.24) comprises the negative sum of both and therefore finds a compromise between contrast and consistency, taking long-range information into account. Because the cost function is differentiable, I can provide Matlab's *fmincon* optimiser with analytical gradients for fast minimisation with respect to all action vector entries.

3.4.2 Results

Regular graphs produce regular actions

On a square grid graph, optimising controllability and simplicity produced action embeddings that have north-versus-south (one positive, one negative) in one dimension and east-versus-west in another dimension (Figure 3.21a-c). On small graphs there are some boundary effects, but larger grid graphs have clear north-versus-south and east-versus-west dimensions at their center. Regular triangular graphs produced the same pattern of action embedding, reflecting similar two-dimensional topology (Figure 3.21d).

Irregular features are hard to interpret

I then test a more irregular graph: a square grid rectangular 'room' with a line-structure attached as a 'hallway' (Figure 3.22). Action vectors then reflect the change in graph topology with embedding dimensions representing actions leading into the hallway (Figure 3.22a). However, these results are difficult to assess because outcomes depend on weight factor α in Equation (3.24). For example, a weight factor that heavily emphasises simplicity results in identical action vectors everywhere, which clearly do not carry any information about action outcomes (Figure 3.22b).

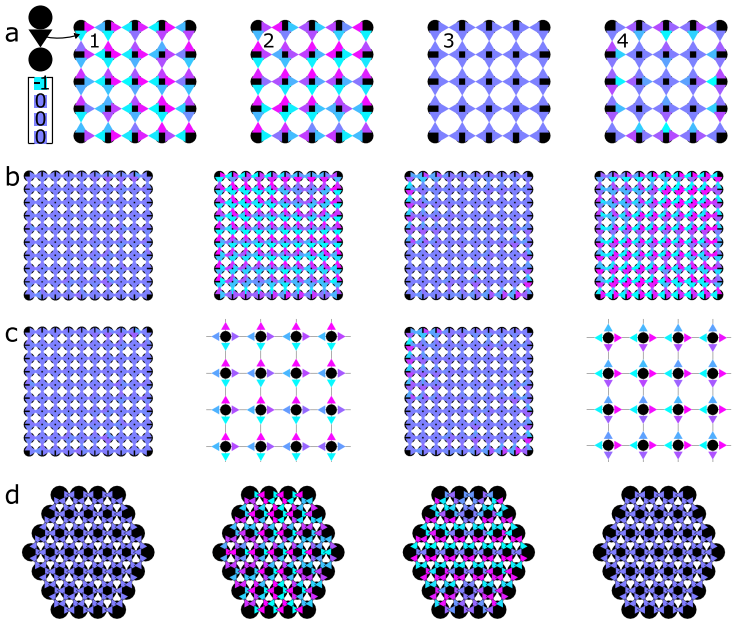


Figure 3.21: Regular grids. a. Four-dimensional action embedding on five by five square grid graph, and how to read them. Each triangle is an action, and each action has a four-dimensional representation vector associated to it. The value in the first vector dimension is plotted in the first column, the second dimension in the second column, and so on. Here, the first dimension contains some west-versus-east action embeddings, the second north-versus-south, the third is empty, and the forth shows a border-to-center pattern. b. Four-dimensional action embedding on a ten by ten square grid graph. The second and fourth dimension show north-versus-south and west-versus-east patterns, respectively. c. Zoomed-in version of four by four central states in the second and fourth dimension of *b*. d. Regular triangular graph with east-versus-west in the second and north-versus-south in the third dimension.

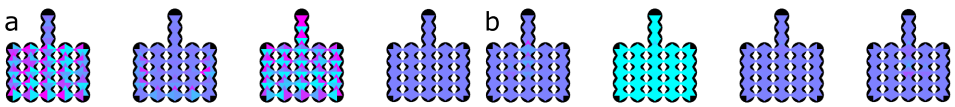


Figure 3.22: Hallway a. Four-dimensional action embedding on five by five square grid graph with hallway of length three attached. The first dimension shows an east-versus-west pattern, the second a weak border-to-center pattern, the third an into-hallway pattern, and the fourth is empty. b. Interpreting embeddings on irregular graphs is hard because of the lack of a ground truth. Here, strongly emphasising simplicity over controllability through a large weight factor α produces identical action vectors for each action. Clearly that is not right – but what choice of α is?

3.4.3 Discussion

Here I have presented a method for action embedding from state space structure to represent actions by their consequences with long-range consistency. I quantify the simplicity and the controllability of a given action embedding on a graph, and optimise these to obtain optimal representations. Because I turn action embedding into an optimisation problem, the method suffers from the same limitations as any optimisation problem. In particular because I am optimising over a large number of parameters, $\mathrm{D}|\mathcal{A}|$, initial guesses matter and it is easy to get stuck in local minima.

However, there is a more general issue with the approach presented here. The lack of a ground truth makes it difficult to evualuate embeddings produced by this method. Hyperparameters like the weighting between simplicity and controllability, or the dimensionality of action representations, cannot be validated because there is no way to quantify the quality of a resulting embedding outside of the framework. The same problem hinders comparisons to other methods for action embedding from structure. For example, calculating eigenvectors of the graph Laplacian, then projecting these eigenvectors on states as state representations, and representing actions by state representation differences results in qualitatively similar action vectors. Which approach is better? A clean definition of what action embeddings need to achieve, or an algorithm they need to support, is necessary to further motivate the method and choices presented in this section. Given the models of previous sections, path integration on arbitrary graphs may be a good candidate mechanism to evaluate action embeddings.

4
Conclusion

I started this thesis by asking two questions about the use of relational and compositional representations of states and spaces in the brain. The first concerns the representation of knowledge: How does the brain organise experience to make sense of the world? The second question is about representation of plans: How can the brain make good decisions, even in environments it has not encountered before? I hypothesised a central role for composition through relational representations in both of these. Composition within states, the construction of situations from related objects and stories from related situations, underlies understanding of objects and situations. Composition between states, the construction of state spaces from related structural elements, allows for generalisation of learned actions to new environments. I tested these two hypotheses in different but complementary ways, through experiment and simulation. Chapter 2 described a neuroimaging experiment that investigates representation of abstract meaning in the brain. To probe the hypothesised organisation of knowledge, the experiment aimed to identify neural responses to the similarities of objects and situations induced by their compositional regularities. Chapter 3 presented a model of state space navigation based on object vector representations. Simulations showed that policies learned on maps com-

posed of such relational representations apply in new configurations without additional learning, and I suggested replay as a way to construct those maps.

No conclusive evidence for understanding through composition

At the core of the neuroimaging experiment presented in Chapter 2 lies a set of video stimuli that emulates the regularities of situation and story composition of the real world, but in a controlled and randomisable way. The video stimuli depict situations of objects in dynamic relations. Certain objects only occur in certain situations, and stories consist of situations in a particular order. I familiarised participants with these regularities though prediction tasks on three subsequent days. The hypothesis is that with experience, the regularities induce abstract similarities between objects and situations. I expect that objects that share situations and situations that share objects become conceptually similar, and temporally proximal situations in stories become sequentially similar. Thus, in addition to the perceptual similarities of objects and situations, I look for neural representations of abstract conceptual and sequential similarities. I applied repetition suppression, representational similarity, and encoding analyses. Across the three, visual areas represented perceptual stimulus features, but I did not find significant or consistent responses for abstract similarities. The first question, about the representation of knowledge, remains unanswered: I do not find evidence for the hypothesised understanding through composition.

Compositional maps support generalisable policies

The model of Chapter 3 considers the problem of learning how to act to reach a goal as learning a mapping from states to actions. In reinforcement learning, this mapping is usually learned from scratch in a new environment through credit assignment or planning. The former requires large amounts of experience, the latter, computation. I proposed a different approach: changing the state representation so that the same mapping, once learned, can be applied anywhere. I represent states by relative positions of objects in the environment, inspired by

populations of object vector cells. These state representations have two crucial properties: first, they imply behaviour, and second, they path integrate. After object discovery, the model path integrates the corresponding object vector representation, and binds it to observed locations in memory – either physically or in replay. In this interpretation, replay constructs a compositional map of behavioural building blocks. The resulting map accurately provided optimal actions, and laying it down in replay makes it robust against path integration noise. I then showed a hierarchical extension to the model, and justified some of the model's assumptions: I showed how object vector cells can be learned through action prediction, and suggested a method of action embedding from structure. Together, the findings of Chapter 3 support the hypothesis related to the second question, about the representation of plans. The composition of spaces from structural building blocks allows for fast and flexible decision making, and there are hints of the proposed mechanism in recent replay research.

Future directions for building meaning and maps

I can see multiple future directions of the work presented here, and I will highlight a few. First, it would be valuable to refine the behavioural paradigm of Chapter 2 to get a within-participant measure of abstract embedding. The across-participant confusion matrices at the final day of behavioural training suggest that on average, participants organise stimuli by their abstract regularities. But to have better chances of finding that effect in the brain, it is important to quantify the embedding space within single subjects. The same measure would provide an inclusion criterion, to boost the neural signal by only including participants with clear embeddings, and a group-level regressor, to probe whether embedding strength correlates with neural responses.

Armed with a behavioural signature of abstract representation, the paradigm allows for further investigation of its neural implementation. One option would be to investigate the temporal dynamics of abstraction. Zhan et al. 2019 use magnetoencephalography (MEG) to study how an ambiguous scene is perceived

and classified, and show how high-dimensional stimulus information turns into a low-dimensional behavioural decision variable from occipital to temporal lobe. What are the equivalent spatiotemporal dynamics for representations along the hierarchy of abstraction hypothesised here? Another intriguing question is whether the same types of abstract meaning that Chapter 2 investigated exist in non-human primates. The temporal lobe is unique to primates (Bryant and Preuss 2018), with particular reorganisation of area locations and connections in humans (Eichert et al. 2020). There are two attractive properties of Chapter 2's paradigm that make it suitable for an experiment in non-human primates: it does not rely on language, and it teaches the regularities in a relatively naturalistic way. The motion prediction task is easily adapted for touch screen operation and does not require extensive instruction, and may be installed as a home-room game for automated training (Butler and Kennerley 2019). Do non-human primates represent semantic meaning without language? Finally, although Chapter 2 studies this semantic meaning as understanding resulting from the world's regularities instead of from language, the two are intimately related. How does the language system interact with the semantic system studied here? Bozeat et al. 2000 found that, in addition to the pyramids and palm trees test, semantic dementia patients are also impaired in matching sounds to situations. They conclude that semantic dementia is an impairment to central semantic knowledge, which manifests itself in all modalities that rely thereon – language is simply one of those. But learning the meaning of non-physical concepts, like 'freedom', does require language. Interestingly, the patient described at the beginning of Chapter 2 was able to explain such concepts like *hint* ("When you're given information. What will occur.") or *pact* ("Friendly agreement.") – a striking contrast to their concrete word explanations (Warrington 1975). Other patients did not show this pattern, and were equally impaired on either. Language thus both relies on and contributes to semantic knowledge. How do these systems interact, particularly during early development?

An important extension for the model of Chapter 3 would be a theory of prioritised replay, analogous to Mattar and Daw 2017. Given the hypothesised purpose of replay as the construction of a map for future behaviour through path integration, what would the optimal content of replay be? Precise predictions of what needs to be replayed would allow for testing the proposed model and contrasting it to alternatives through careful experiments. Another question is how the model would deal with structural changes in dynamic environments, for example when a barrier disappears (Shamash et al. 2020). Memories relating to the goal remain relevant, but memories referring to the barrier are now wrong. How is the map updated, and what are the consequences for behaviour? The theory of prioritised replay should accommodate these scenarios as well: what replay content optimally fixes now-obsolete memories to support appropriate behaviour in the new environment configuration?

Different coordinate systems play a central role in the model. Object-centred coordinates, implemented by object vector cells, afford policies that generalise across environments and configurations. Building a map of the current environment is like shifting the different coordinate systems around, then tying them together in memory to fixed world coordinates, implemented by grid cells. To retrieve such memories from observation, they need to be tagged with sensory coordinates, implemented by place cells. The hierarchical extension of the model proposed that these coordinates can exist simultaneously on different hierarchical levels – and suggested these high-level coordinates may not only index physical space, like different rooms, but also task coordinates within the same physical space, like steps of a recipe in the same kitchen. For example, washing vegetables needs to be done at the sink, chopping them at the countertop, and frying them on the hob, all within the same kitchen. Abstracting the task coordinate (washing, chopping, frying) from its spatial coordinates allows for preparing the recipe in kitchens with different layouts, but completing the task also requires grounding these task states in the current kitchen layout (sink, countertop, hob). How does the brain learn abstract task state coordinates

across environments, and apply them in the current environment? Prefrontal cortex may track the location in the task (Koechlin and Summerfield 2007; Wilson et al. 2014; Schuck et al. 2016; Zhou et al. 2021), like a 'frying' code, while entorhinal cortex represents what that means in the current environment, for example with 'hob-vector cells'. How do these different coordinates interact and map onto neural circuits?

Chapter 3's model composes maps from structural elements, like walls and rewards. These structural elements are currently observed from the environment, but in abstract environments (or when walking around with your eyes closed), that information may not be available. How can the brain figure out what the relevant structural building blocks are without an explicit signal? This is the problem of structural inference. It is closely related to the program induction for concept learning from Lake et al. 2015: what are the building blocks that when composed make up a set of observations? Primitives can combine into composite building blocks, growing new concepts on those built previously (Ellis et al. 2020). For example, parallel walls construct corridors, orthogonal walls construct corners, and four corners together build a room. What are the principles and mechanisms of structural inference, to allow finding primitive elements for constructing complex behaviours?

These questions and directions will hopefully serve as inspiration to continue the research from this thesis on how the brain builds maps and meaning.

Appendices

Neuropoetry

Oh wat een lijden is
Oxford bij tijden want
Wil je een prof worden?
Prof van het brein?

Is de conclusie dat
fMRI-studies
Makkelijk lijken maar
Hachelijk zijn.

M.J. Bakermans-Kranenburg

Oh how much suffering
Oxford is offering
When you want expertise,
Knowledge of brain.

You will experience
fMRI-studies
Sound like a pleasure
But end up a pain.

Translation: Jacob Bakermans

My mother wrote the (not-entirely-serious) poem above about my experience as a DPhil student in Neuroscience. It is an ollekebolleke: a form of poetry known as double-dactyl in English and introduced in Dutch – my mother tongue – by

Drs P. Ollekebollekes follow a strict rhythmic structure, consisting of two stanzas, each three lines of dactylic dimeter and a final choriamb line. Ollekebolles are epigrams: they usually contain a punch line. To further complicate things, line 4 and 8 should rhyme, and line 6 should be a single six-syllable double-dactylic word. These requirements make it hard to write an ollekebolleke, but all the more satisfying when successful. Inspired by the poem above, I decided to combine this hobby with my research and in particular my literature study. I have included below a small collection of my ollekebollekes on papers published in scientific journals, old and famous as well as new and exciting, on topics in neuroscience ranging from memory to decision making, and from learning to consciousness. In probably the greatest achievement of my DPhil, one actually found its way into a figure in a scientific journal.

Constant sub-second cycling between representations of possible futures in the hippocampus
Kay et al. 2020

Cognitive faculties such as imagination, planning, and decision-making entail the ability to represent hypothetical experience. Crucially, animal behavior in natural settings implies that the brain can represent hypothetical future experience not only quickly but also constantly over time, as external events continually unfold. To determine how this is possible, we recorded neural activity in the hippocampus of rats navigating a maze with multiple spatial paths. We found neural activity encoding two possible future scenarios (two upcoming maze paths) in constant alternation at 8 Hz: one scenario per 125-ms cycle. Further, we found that the underlying dynamics of cycling (both inter- and intra-cycle dynamics) generalized across qualitatively different representational correlates (location and direction). Notably, cycling occurred across moving behaviors, including during running. These findings identify a general dynamic process capable of quickly and continually representing hypothetical experience, including that of multiple possible futures.

Can you imagine:
Two future scenarios
Taking their turns
on the theta-wave tide.

Sub-second cycles of
generativity:
Daydreaming when you
lack time to decide.

Imaging unconscious semantic priming
Dehaene et al. 1998

Visual words that are masked and presented so briefly that they cannot be seen may nevertheless facilitate the subsequent processing of related words, a phenomenon called masked priming. It has been debated whether masked primes can activate cognitive processes without gaining access to consciousness. Here we use a combination of behavioural and brain-imaging techniques to estimate the depth of processing of masked numerical primes. Our results indicate that masked stimuli have a measurable influence on electrical and haemodynamic measures of brain activity. When subjects engage in an overt semantic comparison task with a clearly visible target numeral, measures of covert motor activity indicate that they also unconsciously apply the task instructions to an unseen masked numeral. A stream of perceptual, semantic and motor processes can therefore occur without awareness.

Flash of a word,
then a number comparison,
altered by something
you don't even see.

Reading and maths happen
involuntarily.
Why then they take so much
effort for me?

Creating a false memory in hippocampus
Ramirez et al. 2013

Memories can be unreliable. We created a false memory in mice by optogenetically manipulating memory engram–bearing cells in the hippocampus. Dentate gyrus (DG) or CA1 neurons activated by exposure to a particular context were labeled with channelrhodopsin-2. These neurons were later optically reactivated during fear conditioning in a different context. The DG experimental group showed increased freezing in the original context, in which a foot shock was never delivered. The recall of this false memory was context-specific, activated similar downstream regions engaged during natural fear memory recall, and was also capable of driving an active fear response. Our data demonstrate that it is possible to generate an internally represented and behaviorally expressed fear memory via artificial means.

Counterfeit memories:
Context B's fear response
Never evoked in place A
Is unlocked.

Trick cells by triggering
Optogenetically.
Do you remember?
You must have been shocked.

Distributed coding of choice, action and engagement across the mouse brain

Steinmetz et al. 2019

Vision, choice, action and behavioural engagement arise from neuronal activity that may be distributed across brain regions. Here we delineate the spatial distribution of neurons underlying these processes. We used Neuropixels probes to record from approximately 30,000 neurons in 42 brain regions of mice performing a visual discrimination task. Neurons in nearly all regions responded non-specifically when the mouse initiated an action. By contrast, neurons encoding visual stimuli and upcoming choices occupied restricted regions in the neocortex, basal ganglia and midbrain. Choice signals were rare and emerged with indistinguishable timing across regions. Midbrain neurons were activated before contralateral choices and were suppressed before ipsilateral choices, whereas forebrain neurons could prefer either side. Brain-wide pre-stimulus activity predicted engagement in individual trials and in the overall task, with enhanced subcortical but suppressed neocortical activity during engagement. These results reveal organizing principles for the distribution of neurons encoding behaviourally relevant variables across the mouse brain.

Thousands of neurons
in forty-two areas!
Choice coding only in
limited space.

Action, engagement both
unlocalisable:
Sometimes your mind is
all over the place.

SUSTAIN: A Network Model of Category Learning

Love et al. 2004

SUSTAIN (Supervised and Unsupervised STratified Adaptive Incremental Network) is a model of how humans learn categories from examples. SUSTAIN initially assumes a simple category structure. If simple solutions prove inadequate and SUSTAIN is confronted with a surprising event (e.g., it is told that a bat is a mammal instead of a bird), SUSTAIN recruits an additional cluster to represent the surprising event. Newly recruited clusters are available to explain future events and can themselves evolve into prototypes–attractors–rules. SUSTAIN's discovery of category substructure is affected not only by the structure of the world but by the nature of the learning task and the learner's goals. SUSTAIN successfully extends category learning models to studies of inference learning, unsupervised learning, category construction, and contexts in which identification learning is faster than classification learning.

Sometimes a concept is
not so straightforward,
a bat looks a bit like
a bird to the brain.

Start small, but when things get
incomprehensible
clusters are added and
sense will sustain.

A distributional code for value in dopamine-based reinforcement learning
Dabney et al. 2020

Since its introduction, the reward prediction error theory of dopamine has explained a wealth of empirical phenomena, providing a unifying framework for understanding the representation of reward and value in the brain. According to the now canonical theory, reward predictions are represented as a single scalar quantity, which supports learning about the expectation, or mean, of stochastic outcomes. Here we propose an account of dopamine-based reinforcement learning inspired by recent artificial intelligence research on distributional reinforcement learning. We hypothesized that the brain represents possible future rewards not as a single mean, but instead as a probability distribution, effectively representing multiple future outcomes simultaneously and in parallel. This idea implies a set of empirical predictions, which we tested using single-unit recordings from mouse ventral tegmental area. Our findings provide strong evidence for a neural realization of distributional reinforcement learning.

Learning the way that
Reward is distributed:
Dopamine neurons
Accomplish this task.

RPE scalings are
Heterogeneous.
Full glass or empty?
Depends who you ask.

Published in Bakermans et al. 2020

Orthogonal representations for robust context dependent task performance in brains and neural networks
Flesch et al. 2022

How do neural populations code for multiple, potentially conflicting tasks? Here we used computational simulations involving neural networks to define 'lazy' and 'rich' coding solutions to this context-dependent decision-making problem, which trade off learning speed for robustness. During lazy learning the input dimensionality is expanded by random projections to the network hidden layer, whereas in rich learning hidden units acquire structured representations that privilege relevant over irrelevant features. For context dependent decision-making, one rich solution is to project task representations onto low-dimensional and orthogonal manifolds. Using behavioral testing and neuroimaging in humans and analysis of neural signals from macaque prefrontal cortex, we report evidence for neural coding patterns in biological brains whose dimensionality and neural geometry are consistent with the rich learning regime.

Growing a forest takes
leaves and takes branches,
so how to select
the appropriate tree?

Learn to respond with low
dimensionality.
Rich and not lazy:
I wish that was me.

Loss of recent memory after bilateral hippocampal lesions

Scoville and Milner 1957

In 1954 Scoville described a grave loss of recent memory which he had observed as a sequel to bilateral medial temporal-lobe resection in one psychotic patient and one patient with intractable seizures. In both cases the operations had been radical ones, undertaken only when more conservative forms of treatment had failed. The removals extended posteriorly along the mesial surface of the temporal lobes for a distance of approximately 8 cm. from the temporal tips and probably destroyed the anterior two-thirds of the hippocampus and hippocampal gyrus bilaterally, as well as the uncus and amygdala. The unexpected and persistent memory deficit which resulted seemed to us to merit further investigation. We have therefore carried out formal memory and intelligence testing of these two patients and also of eight other patients who had undergone similar, but less radical, bilateral medial temporallobe resections. The present paper gives the results of these studies which point to the importance of the hippocampal complex for normal memory function. Whenever the hippocampus and hippocampal gyrus were damaged bilaterally in these operations some memory deficit was found, but not otherwise. We have chosen to report these findings in full, partly for their theoretical significance, and partly as a warning to others of the risk to memory involved in bilateral surgical lesions of the hippocampal region.

Patient forgets things
as fast as observing them.
Old knowledge stays,
but the new fades away.

Only advantage of
hippocampectomy:
Watch Breaking Bad
the first time every day

A map of object space in primate inferotemporal cortex
Bao et al. 2020

The inferotemporal (IT) cortex is responsible for object recognition, but it is unclear how the representation of visual objects is organized in this part of the brain. Areas that are selective for categories such as faces, bodies, and scenes have been found, but large parts of IT cortex lack any known specialization, raising the question of what general principle governs IT organization. Here we used functional MRI, microstimulation, electrophysiology, and deep networks to investigate the organization of macaque IT cortex. We built a low-dimensional object space to describe general objects using a feedforward deep neural network trained on object classification. Responses of IT cells to a large set of objects revealed that single IT cells project incoming objects onto specific axes of this space. Anatomically, cells were clustered into four networks according to the first two components of their preferred axes, forming a map of object space. This map was repeated across three hierarchical stages of increasing view invariance, and cells that comprised these maps collectively harboured sufficient coding capacity to approximately reconstruct objects. These results provide a unified picture of IT organization in which category-selective regions are part of a coarse map of object space whose dimensions can be extracted from a deep network.

Shape recognition has
face patches, body spots:
Islands of knowledge
in oceans of gap.

Representations in
inferotemporal
cortex were no-man's-land,
now there's a map.

References

Aguirre GK (2007). Continuous Carry-over Designs for fMRI. *NeuroImage* 35(4):1480–1494 (page 41).

Aguirre GK, Mattar MG, and Magis-Weinberg L (2011). De Bruijn Cycles for Neural Decoding. *NeuroImage* 56(3):1293–1300 (pages 41, 42).

Akrami A, Kopec CD, Diamond ME, and Brody CD (2018). Posterior Parietal Cortex Represents Sensory History and Mediates Its Effects on Behaviour. *Nature* 554(7692):368–372 (page 41).

Aminoff E and Bar M (2003). Cortical Analysis of Visual Context. *Neuron* 38(2):347–358 (page 39).

Bak TH and Hodges JR (2003). Kissing and Dancing—a Test to Distinguish the Lexical and Conceptual Contributions to Noun/Verb and Action/Object Dissociation. Preliminary Results in Patients with Frontotemporal Dementia. *Journal of Neurolinguistics*. Objects vs. Actions and Nouns vs. Verbs 16(2):169–181 (page 36).

Bakermans JJW, Muller TH, and Behrens TEJ (2020). Reinforcement Learning : Full Glass or Empty — Depends Who You Ask. *Current Biology* 30(7):R321–R324 (page 166).

Baldassano C, Hasson U, and Norman KA (2018). Representation of Real-World Event Schemas during Narrative Perception. *Journal of Neuroscience* 38(45):9689–9699 (pages 37, 39).

Bao P, She L, McGill M, and Tsao DY (2020). A Map of Object Space in Primate Inferotemporal Cortex. *Nature* 583(7814):103–108 (pages 36, 37, 88, 169).

Bar M (2004). Visual Objects in Context. *Nature Reviews Neuroscience* 5(8):617–629 (page 39).

Baram AB (2020). Linking Reinforcement Learning and Cognitive Maps to Understand How the Brain Represents Abstract Tasks. DPhil. Oxford: University of Oxford (page 17).

Baram AB et al. (2021). Entorhinal and Ventromedial Prefrontal Cortices Abstract and Generalize the Structure of Reinforcement Learning Problems. *Neuron* 109(4):713–723.e7 (pages 48, 91).

Barron HC, Dolan RJ, and Behrens TEJ (2013). Online Evaluation of Novel Choices by Simultaneous Representation of Multiple Memories. *Nature Neuroscience* 16(10):1492–1498 (pages 28, 29).

Barron HC, Garvert MM, and Behrens TE (2016). Repetition Suppression: A Means to Index Neural Representations Using BOLD? *Philosophical Transactions of the Royal Society B: Biological Sciences* 371(1705):(page 65).

Barto AG and Mahadevan S (2003). Recent Advances in Hierarchical Reinforcement Learning. *Discrete Event Dynamic Systems: Theory and Applications* 13(12):341–379 (page 124).

Battaglia PW et al. (2018). Relational Inductive Biases, Deep Learning, and Graph Networks. *arXiv:1806.01261 [cs, stat]*:arXiv: 1806.01261 [cs, stat] (page 28).

Beckmann C and Smith S (2004). Probabilistic Independent Component Analysis for Functional Magnetic Resonance Imaging. *IEEE Transactions on Medical Imaging* 23(2):137–152 (page 62).

Behrens TEJ, Woolrich MW, Walton ME, and Rushworth MFS (2007). Learning the Value of Information in an Uncertain World. *Nature Neuroscience* 10(9):1214–1221 (pages 13, 14).

Behrens TE et al. (2018). What Is a Cognitive Map? Organising Knowledge for Flexible Behaviour. *BioRxiv*:(pages 26, 95).

Blodgett HC (1929). *The Effect of the Introduction of Reward upon the Maze Performance of Rats.* University of California (pages 15, 16).

Boccara CN et al. (2019). The Entorhinal Cognitive Map Is Attracted to Goals. *Science* 363(6434):1443–1447 (page 132).

Bonner MF and Epstein RA (2021). Object Representations in the Human Brain Reflect the Co-Occurrence Statistics of Vision and Language. *Nature Communications* 12(1):4081 (page 37).

Botvinick M and Toussaint M (2012). Planning as Inference. *Trends in Cognitive Sciences* 16(10):485–488 (page 96).

Botvinick MM (2008). Hierarchical Models of Behavior and Prefrontal Function. *Trends in Cognitive Sciences* 12(5):201–208 (page 123).

Botvinick MM, Niv Y, and Barto AC (2009). Hierarchically Organized Behavior and Its Neural Foundations: A Reinforcement Learning Perspective. *Cognition* 113(3):262–280 (page 124).

Botvinick MM (2012). Hierarchical Reinforcement Learning and Decision Making. *Current Opinion in Neurobiology* 22(6):956–962 (page 124).

Bozeat S et al. (2000). Non-Verbal Semantic Impairment in Semantic Dementia. *Neuropsychologia* 38(9):1207–1215 (pages 38, 154).

Brady TF and Oliva A (2008). Statistical Learning Using Real-World Scenes: Extracting Categorical Regularities Without Conscious Intent. *Psychological Science* 19(7):678–685 (page 39).

Bryant KL and Preuss TM (2018). A Comparative Perspective on the Human Temporal Lobe. *Digital Endocasts: From Skulls to Brains*. Ed. by Bruner E, Ogihara N, and Tanabe HC. Replacement of Neanderthals by Modern Humans Series. Tokyo: Springer Japan, pp. 239–258 (page 154).

Buračas GT and Boynton GM (2002). Efficient Design of Event-Related fMRI Experiments Using M-Sequences. *NeuroImage* 16(3, Part A):801–813 (page 41).

Bush D, Barry C, Manson D, and Burgess N (2015). Using Grid Cells for Navigation. *Neuron* 87(3):507–520 (page 20).

Butler JL and Kennerley SW (2019). Mymou: A Low-Cost, Wireless Touchscreen System for Automated Training of Nonhuman Primates. *Behavior Research Methods* 51(6):2559–2572 (page 154).

Cai MB, Schuck NW, Pillow JW, and Niv Y (2019). Representational Structure or Task Structure? Bias in Neural Representational Similarity Analysis and a Bayesian Method for Reducing Bias. *PLOS Computational Biology* 15(5):e1006299 (page 46).

Chafee MV, Averbeck BB, and Crowe DA (2007). Representing Spatial Relationships in Posterior Parietal Cortex: Single Neurons Code Object-Referenced Position. *Cerebral Cortex* 17(12):2914–2932 (page 38).

Chomsky N (1965). *Aspects of the Theory of Syntax*. Fiftieth. The MIT Press (page 28).

Clarke A and Tyler LK (2014). Object-Specific Semantic Coding in Human Perirhinal Cortex. *Journal of Neuroscience* 34(14):4766–4775 (pages 37, 45, 46).
Cohen NJ and Squire LR (1980). Preserved Learning and Retention of Pattern-Analyzing Skill in Amnesia: Dissociation of Knowing How and Knowing That. *Science*:(page 24).
Constantinescu AO, O'Reilly JX, and Behrens TEJ (2016). Organizing Conceptual Knowledge in Humans with a Gridlike Code. *Science (New York, N.Y.)* 352(6292):1464–8 (pages 20, 22).
D'Ardenne K, McClure SM, Nystrom LE, and Cohen JD (2008). BOLD Responses Reflecting Dopaminergic Signals in the Human Ventral Tegmental Area. *Science*:(page 12).
Dabney W et al. (2020). A Distributional Code for Value in Dopamine-Based Reinforcement Learning. *Nature* 577(7792):671–675 (page 166).
Dale AM (1999). Optimal Experimental Design for Event-Related fMRI. *Human Brain Mapping* 8(2-3):109–114 (page 40).
Dale AM, Fischl B, and Sereno MI (1999). Cortical Surface-Based Analysis: I. Segmentation and Surface Reconstruction. *NeuroImage* 9(2):179–194 (page 71).
Davachi L (2006). Item, Context and Relational Episodic Encoding in Humans. *Current Opinion in Neurobiology* 16(6):693–700 (page 39).
Daw ND, Niv Y, and Dayan P (2005). Uncertainty-Based Competition between Prefrontal and Dorsolateral Striatal Systems for Behavioral Control. *Nature Neuroscience* 8(12):1704–1711 (pages 19, 96).
Daw ND et al. (2006). Cortical Substrates for Exploratory Decisions in Humans. *Nature* 441(7095):876–879 (pages 13, 14).
Daw ND et al. (2011). Model-Based Influences on Humans' Choices and Striatal Prediction Errors. *Neuron* 69(6):1204–1215 (pages 21, 23).
Dayan P (1993). Improving Generalisation for Temporal Difference Learning: The Successor Representation. *Neural Computation* 5(4):613–624 (page 96).
Dayan P and Niv Y (2008). Reinforcement Learning: The Good, The Bad and The Ugly. *Current Opinion in Neurobiology* 18(2):185–196 (page 17).
Dehaene S et al. (1998). Imaging Unconscious Semantic Priming. *Nature* 395(6702):597–600 (page 162).
Demb JB et al. (1995). Semantic Encoding and Retrieval in the Left Inferior Prefrontal Cortex: A Functional MRI Study of Task Difficulty and Process Specificity. *Journal of Neuroscience* 15(9):5870–5878 (page 65).
Doeller CF, Barry C, and Burgess N (2010). Evidence for Grid Cells in a Human Memory Network. *Nature* 463(7281):657–661 (page 20).
Dolan RJ and Dayan P (2013). Goals and Habits in the Brain. *Neuron* 80(2):312–325 (page 17).
Dragoi G and Tonegawa S (2011). Preplay of Future Place Cell Sequences by Hippocampal Cellular Assemblies. *Nature* 469(7330):397–401 (page 97).
Drucker DM and Aguirre GK (2009). Different Spatial Scales of Shape Similarity Representation in Lateral and Ventral LOC. *Cerebral Cortex* 19(10):2269–2280 (page 88).
Dusek JA and Eichenbaum H (1997). The Hippocampus and Memory for Orderly Stimulus Relations. *Proceedings of the National Academy of Sciences* 94(13):7109–7114 (page 24).

Earle AC, Saxe AM, and Rosman B (2017). Hierarchical Subtask Discovery With Non-Negative Matrix Factorization. *arXiv:1708.00463 [cs]*:arXiv: 1708.00463 [cs] (page 124).

Eckstein L and Schwarz A (2019). The Making of Tupaia's Map: A Story of the Extent and Mastery of Polynesian Navigation, Competing Systems of Wayfinding on James Cook's Endeavour, and the Invention of an Ingenious Cartographic System. *The Journal of Pacific History* 54(1):1–95 (page 94).

Eichenbaum H (1999). The Hippocampus and Mechanisms of Declarative Memory. *Behavioural Brain Research* 103(2):123–133 (page 24).

Eichenbaum H and Cohen NJ (2014). Can We Reconcile the Declarative Memory and Spatial Navigation Views on Hippocampal Function? *Neuron* 83(4):764–770 (pages 24, 25).

Eichenbaum H et al. (1999). The Hippocampus, Memory, and Place Cells: Is It Spatial Memory or a Memory Space? *Neuron* 23(2):209–226 (page 24).

Eichert N et al. (2020). Cross-Species Cortical Alignment Identifies Different Types of Anatomical Reorganization in the Primate Temporal Lobe. *eLife* 9:ed. by Verstynen T, Gold JI, Verstynen T, and Heuer K, e53232 (page 154).

Ellis K et al. (2020). DreamCoder: Growing Generalizable, Interpretable Knowledge with Wake-Sleep Bayesian Program Learning. *arXiv:2006.08381 [cs]*:arXiv: 2006.08381 [cs] (page 156).

Fischl B (2012). FreeSurfer. *NeuroImage*. 20 YEARS OF fMRI 62(2):774–781 (page 71).

Fischl B, Sereno MI, Tootell RB, and Dale AM (1999). High-Resolution Intersubject Averaging and a Coordinate System for the Cortical Surface. *Human Brain Mapping* 8(4):272–284 (page 71).

Flesch T et al. (2022). Orthogonal Representations for Robust Context-Dependent Task Performance in Brains and Neural Networks. *Neuron* 110(7):1258–1270.e11 (page 167).

Foster DJ and Wilson MA (2006). Reverse Replay of Behavioural Sequences in Hippocampal Place Cells during the Awake State. *Nature* 440(7084):680–683 (page 97).

Friston KJ et al. (1994). Statistical Parametric Maps in Functional Imaging: A General Linear Approach. *Human Brain Mapping* 2(4):189–210 (page 67).

Fritsche M, Solomon SG, and Lange FP de (2022). Brief Stimuli Cast a Persistent Long-Term Trace in Visual Cortex. *Journal of Neuroscience* 42(10):1999–2010 (page 41).

Garvert MM, Dolan RJ, and Behrens TE (2017). A Map of Abstract Relational Knowledge in the Human Hippocampal–Entorhinal Cortex. *eLife* 6:1–20 (pages 39, 90).

Gauthier JL and Tank DW (2018). A Dedicated Population for Reward Coding in the Hippocampus. *Neuron* 99(1):179–193.e7 (page 20).

George D et al. (2021). Clone-Structured Graph Representations Enable Flexible Learning and Vicarious Evaluation of Cognitive Maps. *Nature Communications* 12(1):2392 (page 133).

Gillespie AK et al. (2021). Hippocampal Replay Reflects Specific Past Experiences Rather than a Plan for Subsequent Choice. *Neuron* 109(19):3149–3163.e6 (page 122).

Glasser MF et al. (2013). The Minimal Preprocessing Pipelines for the Human Connectome Project. *NeuroImage* 80:105–124 (page 71).

Griffanti L et al. (2017). Hand Classification of fMRI ICA Noise Components. *NeuroImage* 154(June 2016):188–205 (page 62).
Grill-Spector K, Henson R, and Martin A (2006). Repetition and the Brain: Neural Models of Stimulus-Specific Effects. *Trends in Cognitive Sciences* 10(1):14–23 (pages 41, 65).
Grill-Spector K et al. (1999). Differential Processing of Objects under Various Viewing Conditions in the Human Lateral Occipital Complex. *Neuron* 24(1):187–203 (pages 37, 65).
Gross CG, Schiller PH, Wells C, and Gerstein GL (1967). Single-Unit Activity in Temporal Association Cortex of the Monkey. *Journal of Neurophysiology* 30(4):833–843 (pages 41, 65).
Gupta AS, van der Meer MAA, Touretzky DS, and Redish AD (2010). Hippocampal Replay Is Not a Simple Function of Experience. *Neuron* 65(5):695–705 (page 97).
Hafting T et al. (2005). Microstructure of a Spatial Map in the Entorhinal Cortex. *Nature* 436(7052):801–806 (pages 19, 20).
Halgren E et al. (1999). Location of Human Face-Selective Cortex with Respect to Retinotopic Areas. *Human Brain Mapping* 7(1):29–37 (page 71).
Harlow HF (1949). The Formation of Learning Sets. *Psychological Review* 56(1):51–65 (pages 16, 17, 95).
Hassabis D, Kumaran D, Vann SD, and Maguire EA (2007). Patients with Hippocampal Amnesia Cannot Imagine New Experiences. *Proceedings of the National Academy of Sciences* 104(5):1726–1731 (page 29).
Hassabis D and Maguire EA (2007). Deconstructing Episodic Memory with Construction. *Trends in Cognitive Sciences* 11(7):299–306 (page 29).
Herrington JD, Nymberg C, and Schultz RT (2011). Biological Motion Task Performance Predicts Superior Temporal Sulcus Activity. *Brain and Cognition* 77(3):372–381 (page 38).
Hornsby AN et al. (2020). Conceptual Organization Is Revealed by Consumer Activity Patterns. *Computational Brain & Behavior* 3(2):162–173 (page 37).
Howard D and Patterson KE (1992). The Pyramids and Palm Trees Test. *https://eprints.ncl.ac.uk*:(page 38).
Høydal ØA et al. (2019). Object-Vector Coding in the Medial Entorhinal Cortex. *Nature* 568(7752):400–404 (pages 20, 99, 101).
Hunt LT et al. (2018). Triple Dissociation of Attention and Decision Computations across Prefrontal Cortex. *Nature Neuroscience* 21(10):1471–1481 (page 45).
Huth AG, Nishimoto S, Vu AT, and Gallant JL (2012). A Continuous Semantic Space Describes the Representation of Thousands of Object and Action Categories across the Human Brain. *Neuron* 76(6):1210–1224 (page 37).
Igata H, Ikegaya Y, and Sasaki T (2021). Prioritized Experience Replays on a Hippocampal Predictive Map for Learning. *Proceedings of the National Academy of Sciences* 118(1):(page 122).
Iversen I (1992). Skinner's Early Research: From Reflexology to Operant Conditioning. American Psychologist, 47, 1318-1327. *American Psychologist* 47:1318–1328 (pages 5, 6).
Jenkinson M, Bannister P, Brady M, and Smith S (2002). Improved Optimization for the Robust and Accurate Linear Registration and Motion Correction of Brain Images. *NeuroImage* 17(2):825–841 (page 62).

Jenkinson M and Smith S (2001). A Global Optimisation Method for Robust Affine Registration of Brain Images. *Medical Image Analysis* 5(2):143–156 (page 62).
Jenkinson M et al. (2012). FSL. *NeuroImage*. 20 YEARS OF fMRI 62(2):782–790 (page 61).
Johnson A and Redish AD (2007). Neural Ensembles in CA3 Transiently Encode Paths Forward of the Animal at a Decision Point. *Journal of Neuroscience* 27(45):12176–12189 (page 97).
Jones M and Love BC (2007). Beyond Common Features: The Role of Roles in Determining Similarity. *Cognitive Psychology* 55(3):196–231 (page 37).
Kable JW and Glimcher PW (2007). The Neural Correlates of Subjective Value during Intertemporal Choice. *Nature Neuroscience* 10(12):1625–1633 (pages 13, 14).
Kamin LJ (1969). Predictability, Surprise, Attention, and Conditioning. *Punishment and Aversive Behavior.* Ed. by Campbell BA and Church RM. New York: Appleton-Century-Crofts (page 4).
Kandel D, Matias Y, Unger R, and Winkler P (1996). Shuffling Biological Sequences. *Discrete Applied Mathematics* 71(1):171–185 (pages 42–44).
Kay K et al. (2020). Constant Sub-second Cycling between Representations of Possible Futures in the Hippocampus. *Cell*:528976 (pages 97, 161).
Kemp C and Tenenbaum JB (2008). The Discovery of Structural Form. *Proceedings of the National Academy of Sciences* 105(31):10687–10692 (pages 30, 31).
Kocsis L and Szepesvári C (2006). Bandit Based Monte-Carlo Planning. *Machine Learning: ECML 2006*. Ed. by Fürnkranz J, Scheffer T, and Spiliopoulou M. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 282–293 (pages 95, 96).
Koechlin E and Summerfield C (2007). An Information Theoretical Approach to Prefrontal Executive Function. *Trends in Cognitive Sciences* 11(6):229–235 (page 156).
Kourtzi Z and Kanwisher N (2001). Representation of Perceived Object Shape by the Human Lateral Occipital Complex. *Science* 293(5534):1506–1509 (page 37).
Krause EL and Drugowitsch J (2022). A Large Majority of Awake Hippocampal Sharp-Wave Ripples Feature Spatial Trajectories with Momentum. *Neuron* 110(4):722–733.e8 (page 122).
Kriegeskorte N (2008). Representational Similarity Analysis – Connecting the Branches of Systems Neuroscience. *Frontiers in Systems Neuroscience* 2(November):1–28 (page 45).
Kriegeskorte N, Simmons WK, Bellgowan PS, and Baker CI (2009). Circular Analysis in Systems Neuroscience – the Dangers of Double Dipping. *Nature neuroscience* 12(5):535–540 (page 67).
Kriegeskorte N et al. (2008). Matching Categorical Object Representations in Inferior Temporal Cortex of Man and Monkey. *Neuron* 60(6):1126–1141 (page 37).
Kuechenhoff S (2021). From Shape to Understanding: A Hierarchy of Abstraction for Perceptual and Semantic Representations. MSc. Oxford: University of Oxford (page 59).
Laibson D (1997). Golden Eggs and Hyperbolic Discounting*. *The Quarterly Journal of Economics* 112(2):443–478 (page 13).
Lake BM, Salakhutdinov R, and Tenenbaum JB (2015). Human-Level Concept Learning through Probabilistic Program Induction. *Science* 350(6266):1332–1338 (pages 28, 29, 156).

Lake BM, Ullman TD, Tenenbaum JB, and Gershman SJ (2017). Building Machines That Learn and Think like People. *Behavioral and Brain Sciences* 40:(page 28).
Lever C et al. (2009). Boundary Vector Cells in the Subiculum of the Hippocampal Formation. *Journal of Neuroscience* 29(31):9771–9777 (pages 20, 99).
Li L, Walsh TJ, and Littman ML (2016). Towards a Unified Theory of State Abstraction for MDPs. *ISAIM*:10 (page 96).
Liu Y, Dolan RJ, Kurth-Nelson Z, and Behrens TE (2019). Human Replay Spontaneously Reorganizes Experience. *Cell*:1–13 (page 122).
Logan GD and Crump MJC (2011). Chapter One - Hierarchical Control of Cognitive Processes: The Case for Skilled Typewriting. *Psychology of Learning and Motivation*. Ed. by Ross BH. Vol. 54. Advances in Research and Theory. Academic Press, pp. 1–27 (page 123).
Love BC (2017). Concepts, Meaning, and Conceptual Relationships. *The Oxford Handbook of Cognitive Science*:137–150 (page 37).
Love BC, Medin DL, and Gureckis TM (2004). SUSTAIN: A Network Model of Category Learning. *Psychological Review* 111(2):309–332 (page 165).
Lueschow A, Miller EK, and Desimone R (1994). Inferior Temporal Mechanisms for Invariant Object Recognition. *Cerebral Cortex* 4(5):523–531 (page 65).
Mack ML, Love BC, and Preston AR (2018). Building Concepts One Episode at a Time: The Hippocampus and Concept Formation. *Neuroscience Letters*. New Perspectives on the Hippocampus and Memory 680:31–38 (page 37).
Maguire EA, Intraub H, and Mullally SL (2016). Scenes, Spaces, and Memory Traces: What Does the Hippocampus Do? *The Neuroscientist* 22(5):432–439 (page 29).
Malach R (2012). Targeting the Functional Properties of Cortical Neurons Using fMR-adaptation. *NeuroImage*. 20 YEARS OF fMRI 62(2):1163–1169 (page 65).
Manns JR and Eichenbaum H (2006). Evolution of Declarative Memory. *Hippocampus* 16(9):795–808 (page 25).
Mark S et al. (2020). Transferring Structural Knowledge across Cognitive Maps in Humans and Models. *Nature Communications* 11(1):4783 (page 143).
Mattar MG and Daw ND (2017). Prioritized Memory Access Explains Planning and Hippocampal Replay. *bioRxiv*:225664 (pages 97, 123, 155).
McClure SM, Laibson DI, Loewenstein G, and Cohen JD (2004). Separate Neural Systems Value Immediate and Delayed Monetary Rewards. *Science*:(page 13).
McCormick C, Rosenthal CR, Miller TD, and Maguire EA (2017). Deciding What Is Possible and Impossible Following Hippocampal Damage in Humans. *Hippocampus* 27(3):303–314 (pages 30, 31).
McNamee D, Wolpert DM, and Lengyel M (2016). Efficient State-Space Modularization for Planning: Theory, Behavioral and Neural Signatures. *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc. (page 124).
Meijer G (2021). Neurons in the Mouse Brain Correlate with Cryptocurrency Price: A Cautionary Tale (page 91).
Miller KJ, Botvinick MM, and Brody CD (2017). Dorsal Hippocampus Contributes to Model-Based Planning. *Nature Neuroscience* 20(9):1269–1276 (page 22).
Minsky M (1961). Steps toward Artificial Intelligence. *Proceedings of the IRE* 49(1):8–30 (page 95).

Mirenowicz J and Schultz W (1996). Preferential Activation of Midbrain Dopamine Neurons by Appetitive Rather than Aversive Stimuli. *Nature* 379(6564):449–451 (page 11).
Mnih V et al. (2015). Human-Level Control through Deep Reinforcement Learning. *Nature* 518(7540):529–533 (page 95).
Montague PR, Dayan P, and Sejnowski TJ (1996). A Framework for Mesencephalic Dopamine Systems Based on Predictive Hebbian Learning. *Journal of Neuroscience* 16(5):1936–1947 (page 95).
Moore KS, Yi DJ, and Chun M (2013). The Effect of Attention on Repetition Suppression and Multivoxel Pattern Similarity. *Journal of Cognitive Neuroscience* 25(8):1305–1314 (page 88).
Nichols TE and Holmes AP (2001). Nonparametric Permutation Tests for Functional Neuroimaging: A Primer with Examples. *Human Brain Mapping* 15(1):1–25 (page 70).
Nili H et al. (2014). A Toolbox for Representational Similarity Analysis. *PLoS Computational Biology* 10(4):(page 67).
Nonyane BaS and Theobald CM (2007). Design Sequences for Sensory Studies: Achieving Balance for Carry-over and Position Effects. *British Journal of Mathematical and Statistical Psychology* 60(2):339–349 (page 41).
O'Keefe J (1976). Place Units in the Hippocampus of the Freely Moving Rat. *Experimental Neurology* 51(1):78–109 (page 19).
O'Keefe J and Nadel L (1978). *The Hippocampus as a Cognitive Map*. Oxford: Oxford University Press (page 19).
Park SA et al. (2020). Map Making: Constructing, Combining, and Inferring on Abstract Cognitive Maps. *Neuron*:1–13 (page 45).
Pavlov IP (1927). Conditioned Reflexes: An Investigation of the Physiological Activity of the Cerebral Cortex. *Annals of Neurosciences* 17(3):136–141 (page 3).
Peuskens H, Vanrie J, Verfaillie K, and Orban GA (2005). Specificity of Regions Processing Biological Motion. *European Journal of Neuroscience* 21(10):2864–2875 (page 38).
Ponsen M, Taylor ME, and Tuyls K (2010). Abstraction and Generalization in Reinforcement Learning: A Summary and Framework. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 5924 LNAI:1–32 (page 96).
Power JD et al. (2012). Spurious but Systematic Correlations in Functional Connectivity MRI Networks Arise from Subject Motion. *NeuroImage* 59(3):2142–2154 (page 60).
Ramirez S et al. (2013). Creating a False Memory in the Hippocampus. *Science* 341(6144):387–391 (page 163).
Rescorla RA (1966). Predictability and Number of Pairings in Pavlovian Fear Conditioning. *Psychonomic Science* 4(11):383–384 (page 4).
Rescorla RA (1967). Pavlovian Conditioning and Its Proper Control Procedures. *Psychological Review* 74(1):71–80 (page 4).
Reynolds GS (1961). Attention in the Pigeon. *Journal of the Experimental Analysis of Behavior* 4(3):203–208 (page 4).
Robinson EC et al. (2014). MSM: A New Flexible Framework for Multimodal Surface Matching. *NeuroImage* 100:414–426 (page 89).

Robinson EC et al. (2018). Multimodal Surface Matching with Higher-Order Smoothness Constraints. *NeuroImage* 167(September 2017):453–465 (page 89).
Roux L et al. (2017). Sharp Wave Ripples during Learning Stabilize the Hippocampal Spatial Map. *Nature Neuroscience* 20(6):845–853 (page 122).
Salimi-Khorshidi G et al. (2014). Automatic Denoising of Functional MRI Data: Combining Independent Component Analysis and Hierarchical Fusion of Classifiers. *NeuroImage* 90:449–468 (page 62).
Samborska V et al. (2021). Complementary Task Representations in Hippocampus and Prefrontal Cortex for Generalising the Structure of Problems (page 143).
Sapountzis P, Schluppeck D, Bowtell R, and Peirce JW (2010). A Comparison of fMRI Adaptation and Multivariate Pattern Classification Analysis in Visual Cortex. *NeuroImage* 49(2):1632–1640 (page 88).
Saxe AM, Earle AC, and Rosman B (2017). Hierarchy Through Composition with Multitask {LMDP}s. *Proceedings of the 34th International Conference on Machine Learning* 70:3017–3026 (page 124).
Schapiro AC et al. (2014). The Necessity of the Medial Temporal Lobe for Statistical Learning. *Journal of Cognitive Neuroscience* 26(8):1736–1747 (page 39).
Schuck NW, Cai MB, Wilson RC, and Niv Y (2016). Human Orbitofrontal Cortex Represents a Cognitive Map of State Space. *Neuron* 91(6):1402–1412 (pages 21, 22, 46, 156).
Schultz W, Dayan P, and Montague PR (1997). A Neural Substrate of Prediction and Reward. *Science* 275(5306):1593–1599 (pages 11, 12, 95).
Scoville WB and Milner B (1957). Loss of Recent Memory After Bilateral Hippocampal Lesions. *Journal of Neurology, Neurosurgery & Psychiatry* 20(1):11–21 (pages 23, 168).
Seymour B et al. (2004). Temporal Difference Models Describe Higher-Order Learning in Humans. *Nature* 429(6992):664–667 (page 12).
Shamash P et al. (2020). Mice Learn Multi-Step Routes by Memorizing Subgoal Locations. *bioRxiv*:2020.08.19.256867 (pages 96, 155).
Silver D et al. (2016). Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* 529(7587):484–489 (page 95).
Skinner BF (1937). Two Types of Conditioned Reflex: A Reply to Konorski and Miller. *The Journal of General Psychology* 16(1):272–279 (page 5).
Skinner BF (1958). Reinforcement Today. *American Psychologist* 13(3):94–99 (page 5).
Smith SM and Brady JM (1997). SUSAN—A New Approach to Low Level Image Processing. *International Journal of Computer Vision* 23(1):45–78 (page 62).
Solway A et al. (2014). Optimal Behavioral Hierarchy. *PLoS Computational Biology* 10(8):(page 124).
Stachenfeld KL, Botvinick MM, and Gershman SJ (2017). The Hippocampus as a Predictive Map. *Nature Neuroscience* 20(11):1643–1653 (page 96).
Steinmetz NA, Zatka-Haas P, Carandini M, and Harris KD (2019). Distributed Coding of Choice, Action and Engagement across the Mouse Brain. *Nature* 576(December):1–8 (pages 91, 164).
Summerfield C, Luyckx F, and Sheahan H (2020). Structure Learning and the Posterior Parietal Cortex. *Progress in Neurobiology* 184(September 2019):(pages 31, 38).
Sutton RS (1988). Learning to Predict by the Methods of Temporal Differences. *Machine Learning* 3(1):9–44 (page 95).

Sutton RS (1991). Dyna, an Integrated Architecture for Learning, Planning, and Reacting. *ACM SIGART Bulletin* 2(4):160–163 (page 95).
Sutton RS and Barto AG (2018). *Reinforcement Learning, Second Edition: An Introduction*. MIT Press (pages 5, 94).
Sutton RS, Precup D, and Singh S (1999). Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial Intelligence* 112(1):181–211 (page 124).
Szucs D and Ioannidis JP (2020). Sample Size Evolution in Neuroimaging Research: An Evaluation of Highly-Cited Studies (1990–2012) and of Latest Practices (2017–2018) in High-Impact Journals. *NeuroImage* 221:117164 (page 91).
Tenenbaum JB, Kemp C, Griffiths TL, and Goodman ND (2011). How to Grow a Mind: Structure, Statistics, and Abstraction. *Science* 331(6022):1279–1285 (page 30).
Theves S, Fernandez G, and Doeller CF (2019). The Hippocampus Encodes Distances in Multidimensional Feature Space. *Current Biology* 0(0):1–6 (page 46).
Thorndike EL (1898). Animal Intelligence: An Experimental Study of the Associative Processes in Animals. *The Psychological Review: Monograph Supplements* 2(4):i–109 (pages 5, 6).
Thorndike EL (1911). *Animal Intelligence: Experimental Studies*. Lewiston, NY, US: Macmillan Press, pp. viii, 297 (page 5).
Tolman EC, Ritchie BF, and Kalish D (1946). Studies in Spatial Learning. I. Orientation and the Short-Cut. *Journal of Experimental Psychology* 36(1):13–24 (pages 15, 16).
Tolman EC (1948). Cognitive Maps in Rats and Men. *Psychological Review* 55(4):189–208 (pages 15, 16, 95).
Tomlinson MT and Love BC (2006). From Pigeons to Humans: Grounding Relational Learning in Concrete Examples. *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*. AAAI'06. Boston, Massachusetts: AAAI Press, pp. 199–204 (page 37).
Tootell RBH et al. (1995). Visual Motion Aftereffect in Human Cortical Area MT Revealed by Functional Magnetic Resonance Imaging. *Nature* 375(6527):139–141 (pages 41, 71).
Vikbladh OM et al. (2019). Hippocampal Contributions to Model-Based Planning and Spatial Memory. *Neuron* 102(3):683–693.e4 (page 22).
Walther A et al. (2016). Reliability of Dissimilarity Measures for Multi-Voxel Pattern Analysis. *NeuroImage* 137:188–200 (page 48).
Warrington EK (1975). The Selective Impairment of Semantic Memory. *Quarterly Journal of Experimental Psychology* 27(4):635–657 (page 154).
Watkins CJCH and Dayan P (1992). Q-Learning. *Machine Learning* 8(3):279–292 (page 95).
Whittington JCR et al. (2020). The Tolman-Eichenbaum Machine: Unifying Space and Relational Memory through Generalization in the Hippocampal Formation. *Cell* 183(5):1249–1263.e23 (pages 25, 27, 121, 134, 136).
Widloski J and Foster DJ (2022). Flexible Rerouting of Hippocampal Replay Sequences around Changing Barriers in the Absence of Global Place Field Remapping. *Neuron*:S089662732200109X (page 122).
Wikenheiser AM and Schoenbaum G (2016). Over the River, through the Woods: Cognitive Maps in the Hippocampus and Orbitofrontal Cortex. *Nature Reviews Neuroscience* 17(8):513–523 (page 21).

Wilson RC, Takahashi YK, Schoenbaum G, and Niv Y (2014). Orbitofrontal Cortex as a Cognitive Map of Task Space. *Neuron* 81(2):267–278 (pages 21, 156).
Winkler AM et al. (2014). Permutation Inference for the General Linear Model. *NeuroImage* 92:381–397 (page 70).
Wohlgemuth A (1931). University of California. Publications in Psychology, Vol. 4, Nos. 3, 5, 8, 9, 10, 11. Reviews. *Journal of Mental Science* 77(317):413–414 (page 15).
Woolrich MW, Ripley BD, Brady M, and Smith SM (2001). Temporal Autocorrelation in Univariate Linear Modeling of FMRI Data. *NeuroImage* 14(6):1370–1386 (page 61).
Woolrich MW et al. (2004). Multilevel Linear Modelling for FMRI Group Analysis Using Bayesian Inference. *NeuroImage* 21(4):1732–1747 (page 70).
Zhan J, Ince RAA, van Rijsbergen N, and Schyns PG (2019). Dynamic Construction of Reduced Representations in the Brain for Perceptual Decision Behavior. *Current Biology* 29(2):319–326.e4 (page 153).
Zhang Y, Brady M, and Smith S (2001). Segmentation of Brain MR Images through a Hidden Markov Random Field Model and the Expectation-Maximization Algorithm. *IEEE Transactions on Medical Imaging* 20(1):45–57 (page 62).
Zhou J et al. (2021). Evolving Schema Representations in Orbitofrontal Ensembles during Learning. *Nature* 590(7847):606–611 (pages 45, 156).