A Bayesian account of learning and generalising representations in the brain

James C.R. Whittington

Magdalen College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Michaelmas 2019

Abstract

Without learning we would be limited to a set of preprogrammed behaviours. While that may be acceptable for flies[1], it does not provide the basis for adaptive or intelligent behaviours familiar to humans. Learning, then, is one of the crucial components of brain operation. Learning, however, takes time. Thus, the key to adaptive behaviour is learning to systematically generalise; that is, have learned knowledge that can be flexibly recombined to understand any world in front of you. This thesis attempts to make inroads on two questions - how can brain networks learn, and what are the principles behind representations of knowledge that allow generalisation. Though bound by a common framework of Bayesian thinking, this thesis considers the questions in two separate parts.

In the first part of the thesis, we investigate algorithms the brain may use to update connection strengths. While learning attempts to optimise a global function of the brain state, each connection only has access to local information. This is in contrast to artificial networks, where global information is easily conveyed to each synapse via the back-propagation algorithm. We show that, contrary to decades old beliefs, an analogous algorithm to back-propagation could be implemented in the local dynamics and learning rules of brain networks. We show an exact equivalence between the two algorithms and demonstrate that they perform identically on a standard machine learning benchmark. These results are the first to show that an algorithm as efficient as those used in machine learning could be implemented in the brain.

In the second part of the thesis, we investigate frameworks for learning and generalising neural representations. It is proposed that a cognitive map encoding the relationships between entities in the world supports flexible behaviour. This map is traditionally associated with the hippocampal formation, due to its beautiful representations mapping space. This cognitive map, though, seems at odds with the other well-characterised aspect of the hippocampus: relational memory. Here we unify spatial cognition and relational memory within the framework of generalising relational knowledge. Using this framework, we build a machine that learns and generalises knowledge in both spatial and non-spatial tasks, while also displaying representations that mirror those in the brain. Finally, we confirm model predictions in neural data. Together, these results provide a computational framework for a systematic organisation of knowledge spanning all domains of behaviour.

[1] Mayflies to be more specific.

A Bayesian account of learning and generalising representations in the brain

James C.R. Whittington

Magdalen College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2019

But I have not much time. I have friends to discover,
and a great many things to understand.

— Antoine de Saint-Exupéry's *The Little Prince*

Acknowledgements

There are many people I would like to thank for their contributions both academically and otherwise during this DPhil.

Firstly, I would like to thank my DPhil supervisors, Rafal Bogacz and Tim Behrens, for their enthusiastic support and advice during the course of my DPhil. Without their help none of this would have been possible. I was very lucky to have had Rafal as my first introduction to science; his ebullience is always greatly appreciated. My good fortune continued with Tim who has pushed me much further than my laziness would have otherwise dictated. A shame he couldn't help much with skiing though. I hope to continue to work, and stay friends with them both.

Many thanks to both the Behrens and Bogacz groups, with particular mention to Anna, Alon, Emma, and Jacob who have made work feel like play throughout the DPhil, and helped make the final days during the write-up more than just bearable.

Many thanks to my two thesis examiners, Andrew Saxe and Karl Friston - the viva was unexpectedly enjoyable!

Many thanks to Oxford University and Magdalen College for introducing me to many friends over the years. Long suffering ones, Alice, George, Mark, Patricia, Roland, Scarlett and Val - it is difficult to express gratitude for friendships lasting over ten years - may there be many more to come! Shorter suffering ones Harry, Kalina, Jakob, Jimi, Rose, Sam, Tim - I am very thankful for their friendship and support during the DPhil, particularly the first half when most they were still in Oxford! Many thanks to Federico and the Fosci family for welcoming me to Italy. Special mention must go to Lisanne, Ruairidh, and the Stock family - the breaks we took together have been truly wonderful, and you are two friends I am so happy to have.

Finally, I would like to thank my parents, Chris and Gilly, my sisters Bella, Midge, Rosy, and Sarah, their husbands Richard, Mark, Ruairidh, and Mike and their children Arthur, Gus, Leo, Poppy, Rupert, Ani, Rafi, Bea, Lucy, Molly, and Mungo. Asides from Christmas presents, a large family is a real blessing.

Statement of intellectual contribution

The research in this thesis is largely the work of myself along with my supervisors Tim Behrens and Rafal Bogacz. Where possible to be specific I have indicated below.

Introductory chapters

Chapter 3 is based on the review Whittington and Bogacz (2019) which is joint work with Rafal Bogacz.

Much of the material from introduction chapter 6 is based on the review Behrens et al. (2018) of which I am a co-author.

Research chapters

Chapters 4 and 5 are joint work with Rafal Bogacz and are based on Whittington and Bogacz (2017, 2019).

Chapters 7, 8 and 9 are joint work with Tim Behrens and are based on Whittington et al. (2018, 2019).

Additionally chapter 9 is joint work with fellow DPhil student Tim Muller.

Abstract

Without learning we would be limited to a set of preprogrammed behaviours. While that may be acceptable for flies[2], it does not provide the basis for adaptive or intelligent behaviours familiar to humans. Learning, then, is one of the crucial components of brain operation. Learning, however, takes time. Thus, the key to adaptive behaviour is learning to systematically generalise; that is, have learned knowledge that can be flexibly recombined to understand any world in front of you. This thesis attempts to make inroads on two questions - how can brain networks learn, and what are the principles behind representations of knowledge that allow generalisation. Though bound by a common framework of Bayesian thinking, this thesis considers the questions in two separate parts.

In the first part of the thesis, we investigate algorithms the brain may use to update connection strengths. While learning attempts to optimise a global function of the brain state, each connection only has access to local information. This is in contrast to artificial networks, where global information is easily conveyed to each synapse via the back-propagation algorithm. We show that, contrary to decades old beliefs, an analogous algorithm to back-propagation could be implemented in the local dynamics and learning rules of brain networks. We show an exact equivalence between the two algorithms and demonstrate that they perform identically on a standard machine learning benchmark. These results are the first to show that an algorithm as efficient as those used in machine learning could be implemented in the brain.

In the second part of the thesis, we investigate frameworks for learning and generalising neural representations. It is proposed that a cognitive map encoding the relationships between entities in the world supports flexible behaviour. This map is traditionally associated with the hippocampal formation, due to its beautiful representations mapping space. This cognitive map, though, seems at odds with the other well-characterised aspect of the hippocampus: relational memory. Here we unify spatial cognition and relational memory within the framework of generalising relational knowledge. Using this framework, we build a machine that learns and generalises knowledge in both spatial and non-spatial tasks, while also displaying representations that mirror those in the brain. Finally, we confirm model predictions

[2]Mayflies to be more specific.

in neural data. Together, these results provide a computational framework for a systematic organisation of knowledge spanning all domains of behaviour.

Contents

List of Figures

List of Abbreviations

1-D, 2-D . . . One- or two-dimensional.

ANN . . . . . Artificial neural network.

BPTT . . . . . Back-propagation through time.

ERC . . . . . Entorhinal cortex.

ELBO . . . . . Evidence lower bound.

HPC . . . . . Hippocampus.

LEC . . . . . Lateral entorhinal cortex.

MEC . . . . . Medial entorhinal cortex.

MLP . . . . . Multi-layer perceptron.

PCN . . . . . Predictive coding network.

pdf . . . . . Probability distribution function.

pmf . . . . . Probability mass function.

RL . . . . . Reinforcement learning.

SGD . . . . . Stochastic gradient descent.

TEM . . . . . Tolman-Eichenbaum Machine.

VAE . . . . . Variational autoencoder.

Notation Used

x	Random variable.
$\mathbf{x}$	Vector of variables x_i.
$\mathbf{z}$	Latent variables.
$\mathbf{x}^{1:N}$	N samples of variables $\mathbf{x}$.
$\mathcal{D}$	Dataset.
$\mathbf{x}_t$	Variables $\mathbf{x}$ collected at time t.
$\mathbf{s}$	Input.
$\mathbf{t}$	Target.
$\mathcal{M}$	Model.
$P(\cdots)$	Generative distribution.
$Q(\cdots)$	Variational distribution.
θ	Generative parameters.
ϕ	Variational parameters.
W	Matrix of weights for ANNs.
Θ	Matrix of weights for PCNs.
$\mathbb{E}$	Expectation.
$ELBO$	Evidence lower bound.
F	Free energy.
δ	Back-propagation error.
ε	Predictive coding error.
G	Group.
g	Group element.

1 Introduction

Contents

Aristotle believed brains cooled blood in order to keep our intelligent hearts free from hot-bloodedness. Before him, though, Alcmaeon and Hippocrates thought the brain contained the mind. Believing that dissection violated the human body, the ancient Greeks' understanding of the brain was limited. As time went on, views of such sacredness waned and the Romans began to pry inside; Galen deduced from densities and squishiness that the cerebellum controlled muscles and the softer cerebrum processed the senses. Fast-forward a millennia or two, bypassing Descartes' thoughts on dualism, consciousness, and the pineal gland, techniques to examine the brain started to improve; Galvani probed the role of electricity in nerves and von Helmholtz did the same for neurons; various lesions indicated functional segregation, such as executive control in prefrontal cortex through Phineas Gage or expressive language in Broca's area; Golgi and Cajal used microscopes and staining to examine single neurons.

With the industrialisation of science, the twentieth century bore fruit in the form of an increasingly detailed understanding of neurons, synapses, neurotransmitters, resting potentials, action potentials, networks and so on (Sherrington, 1907; Loewi, 1921; McCulloch and Pitts, 1943; Hodgkin and Huxley, 1952). Though we have gained a great level of detail about many of these micro-processes - as well as high-level understandings of intelligence thanks to philosophy, experimental psychology, and behavioural and cognitive neuroscience (Thorndike, 1898; Pavlov, 1927; Skinner, 1938; Ferster and Skinner, 1957; Gibson, 1977) - a large gulf of understanding remains between these levels of granularity.

This thesis focuses on spanning this gap by providing high-level computational frameworks that translate to low-level processes. Any high-level brain framework must have successful behaviour at its heart as that is the role of the brain. Analogously, neurons are central to low-level understanding as the basis of brain function is believed to be the transfer of information between neurons, mediated via weighted connections. Different weights lead to different functions. Thus, learning appropriate configurations of weights is the fundamental problem facing brains. There are two facets to this learning - the first is *how*, and the second is *what*. The how are the learning *algorithms* that determine updates to these synaptic connections, and the what are the neural *representations* that reflect how the world works.

1.1 Neural hardware

Neurons consist of cell bodies and axons (Figure 1.1). Cell bodies process electrical information (after aggregation via dendrites), and axons transmit information. Axons project to other neurons via terminals that synapse onto dendrites. Altering the strength of these connections is learning, with the exact amount of change prescribed by the brain's learning algorithm. This algorithm has access to little information per connection; only the neuron it projects from (Hebb, 1949), the neuron it projects to, local neuromodulators, and local field potentials. From this *local* information alone, weights (synapses) must learn their role as part of a *global* network.

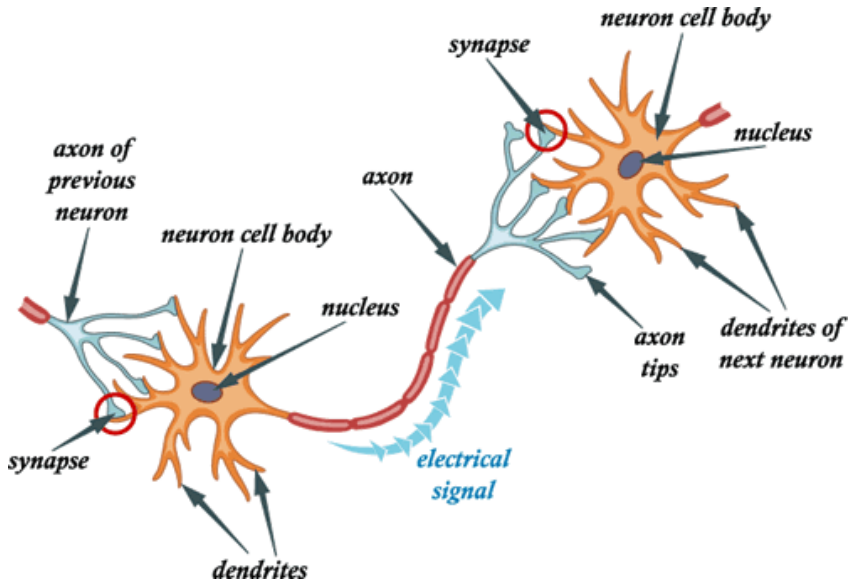

Figure 1.1: Neurons synapsing onto each other. Unknown source.

Though weights encode learned information, it is the cell bodies that represent the world via their activity. This is also the activity that experimenters record. One can record the whole brain with low granularity (e.g. fMRI), or a tiny fraction of the brain with high granularity (e.g. single neuron recording). In either extreme, characteristic patterns of activity to stimuli, known as representations, allow the experimenter to define cell types, or area specific functions.

Approximately one millionth of the brain's neurons can be recorded simultaneously. In contrast, machine learners skipped two millennia of experimental methods development and can record everything in their artificial neural networks (ANNs). Since these networks have been trained to human level performance on complex tasks (Lecun et al., 2015), surely their analysis can aid neuroscientists' understanding of biological intelligence. Alas, so far, this has not been as fruitful as had been hoped with ANN's representations deeply uninterpretable.

1.2 Thesis Research

It is at this point that the research in this thesis commences. It focusses on 1) the algorithmic implementation of learning in biological neural networks, and 2) a computational framework for the neural representations of task generalisation.

Both these research directions are bound together by Bayesian thinking, and both of these pieces of work bridge the gap between high- and low- level understanding, as well as between brains and machines.

1.2.1 ANNs and brain networks

Learning is the process of updating one's belief or understanding on the basis of new information. This is a problem for machines and, more intimately, it the problem faced by the neurons and connections that comprise our brains. Machine learners get to choose their learning algorithm. One algorithm in particular, the back-propagation algorithm (backprop; Rumelhart et al. (1986); Werbos (1994)), is proven to be highly effective. Backprop and ANNs were originally developed for supervised learning, though now also used for unsupervised and reinforcement learning. In supervised learning, the goal is to find a good mapping from inputs to targets (chapters 3).

Though ANNs are loosely based on brain networks, backprop is traditionally seen as incompatible with learning in the brain (reasons discussed in chapters 3 and 4). The main contribution of the first part of this thesis is to show that the backprop is, in fact, implementable in locally connected networks of neurons such as those found in the brain.

We treat the problem of supervised learning like a statistician would by defining a joint probability distribution over variables. We structure the dependencies of variables as a directed acyclic graph with analogous architecture to the layers of neurons in ANNs. We then show approximate inference of these variables can be implemented in locally connected networks of neurons, with global learning obtained using only local learning rules.

We show this learning has an equivalence to backprop, with exact asymptotic convergence. We verify this equivalence on a standard machine learning benchmark, demonstrating equal performance for both algorithms. Finally, we propose alternative architectures implementing efficient learning in biological neural networks.

After introducing this new network, we compare a variety of other models of biological backprop and demonstrate that many models can be viewed under a common framework.

1.2.2 Neural representations for generalisation

Learning is not done all at once nor is it all for one moment; we learn continually, acquiring and deploying new skills when needed. Learning, however, is slow, and so we learn so that we don't have to learn; we learn so that we can operate successfully in the wild when facing new and/or unknown data.

Applying old knowledge to new situations is known as generalisation. This ranges from classifying novel dogs to complex tasks where entire environments may change. The hippocampal formation has a pivotal role in generalisation, though little is known about the mechanism, or the role of its famous neural representations (O'Keefe and Nadel, 1978; Hafting et al., 2005).

The main contribution of the second part of this thesis is threefold. 1) We formalise tasks for structure generalisation - how to transfer information from one task to a structurally similar task. This task setup encompasses seemingly disparate spatial, non-spatial and classic relational memory tasks that all rely on the hippocampal formation. This formalism, then, offers a unifying account of the hippocampal formation. 2) We provide both a computational framework to solve these tasks, and a particular implementation of such ideas - the Tolman-Eichenbaum Machine (TEM). After learning, TEM exhibits neural representations analogous to those found in the hippocampal formation. Thus, TEM provides a mechanistic understanding of the hippocampal role in generalisation, while also offering explanations to the myriad of known cell representations. 3) We verify a neural prediction of this model, showing a previously unknown relationship between cells of hippocampus and entorhinal cortex.

1.3 Thesis Overview

Introduction

Chapter 2 introduces the reader to concepts fundamental to the overall research in this thesis. In particular, it introduces the reader to Bayes, sets the scene for algorithmic implementations of Bayes in the brain, and discusses philosophical considerations for structuring representations.

Part I: Algorithms for learning

Chapter 3 introduces the reader to concepts fundamental to the new research in this part of the thesis. In particular, it focusses on ANNs, and the decades old debate as to whether their learning algorithm, backprop, is implementable in the brain.

Chapter 4 shows backprop is implementable in the brain. More precisely it uses a probabilistic approach to supervised learning, and demonstrates that simple approximate Bayesian inference leads to dynamics that can be implemented in biologically plausible neural networks known as predictive coding networks. The equilibrium point of these dynamics is shown to have a direct equivalence to how errors are propagated in backprop. Furthermore, the learning rule is equivalent to backprop. This work is published in Neural Computation (Whittington and Bogacz, 2017).

Chapter 5 reviews other recent models of back-propagation in the brain, summarising them in two categories - those that represent errors explicitly and those that represent errors temporally. It further describes how many models can be viewed as performing inference and learning in an energy-based framework. This work is published in Trends in Cognitive Sciences (Whittington and Bogacz, 2019).

Part II: Representations for generalisation

Chapter 6 introduces the reader to concepts fundamental to the new research in this part of the thesis. In particular, it focusses on generalisation along with the hippocampal formation and its main functions and representations.

Chapter 7 introduces the Tolman-Eichenbaum machine (TEM), a model that learns and generalises abstract relational knowledge in both spatial and non-spatial tasks. The framework of TEM unifies two fields of neuroscience - spatial cognition and relational memory. Furthermore, TEM manages to predict a wide variety of cellular representations. In this chapter TEM is described in an intuitive, non-mathematical manner. An early version of the work of chapters 7, 8 and 9 is published at NeurIPS (Whittington et al., 2018) and a complete version is currently under review at Cell (Whittington et al., 2019).

Chapter 8 describes TEM in more detail. Specifically a complete mathematical description of the model is provided.

Chapter 9 looks at a neural prediction of TEM. TEM proposes that structural knowledge between grid and place cells is preserved, even after the apparently random remapping of the place code. This prediction is tested and verified in simultaneous recordings of place and grid cells in a remapping experiment.

Afterword

Chapter 10 draws conclusions from the previous chapters and discusses some open questions in the field.

2

Bayes and the Brain

Contents

Learning is the process of updating beliefs or understanding on the basis of new information. A formal account of this process is given using probabilities and Bayes theorem. In this chapter we present a brief overview of the Bayesian framework, consider how the brain may implement Bayes as its learning algorithm, and finally have a philosophical discussion on the nature of representing information. For a more detailed account of the Bayes used in this thesis, please see appendix A, and for more extensive examinations we refer the reader to MacKay (2003); Bishop (2006); Murphy (2014), and in particular Behrens (2004) for an excellent introduction to Bayesian thinking, from where this and some subsequent paragraphs are essentially pinched.

2.1 Modelling data with probability distributions

To understand the physical and natural world a scientist performs experiments and makes observations. Their true interest is not the observed data itself but is in discovering something deeper about how the world works. This understanding often comes in the form of a model - a simplified depiction of the system in question. This model, hopefully, not only offers a parsimonious explanation of the system, but, crucially, predicts future observations.

It is difficult to define the perfect model a priori. Nevertheless, it may be possible to describe assumptions about the structure of the system and leave some parameters to adjust themselves a posteriori. These parameters, θ, bind the model abstraction, $\mathcal{M}$, to the particular data in question, $\mathcal{D}$. For example, one can model vibrations on a string using waves with an amplitude and wavelength. Here the model (wave) prescribes the rules for how the parameters (amplitude, wavelength) combine together to produce the data. Each dataset (string) may have different unknown parameters, but the same model structure (wave) nevertheless accounts for each dataset.

There may be aspects of the system not included in our model structure e.g. thermal fluctuations of the string or differentials in elastic properties. Even if the true data generating process were deterministic, these un-modelled processes mean some behaviours of the system cannot be captured deterministically. These processes are termed as noise. Noise means we cannot make precise predictions of variables, but instead can only make predictions about distributions of variables.

Thus, it is natural to describe data in terms of probability distributions; probability mass functions (pmf) for discrete data and probability distribution functions (pdf) for continuous data. Putting this together, the probability of observing data, $\mathcal{D}$, given model, $\mathcal{M}$, and parameters, θ, is

$$P\left(\mathcal{D} \mid \mathcal{M}, \theta\right) \tag{2.1}$$

Thinking with probabilities means we are not surprised when model predictions do not exactly agree with the data. Rather, we formally account for surprise by considering where on the distribution the data lies[1].

2.1.1 Maximum likelihood learning

Learning constitutes becoming less surprised. It seems clear then that we should find the set of parameters that most reduces our surprise. More formally,

$$\theta^* = \arg\min_{\theta} P\left(\mathcal{D} \mid \mathcal{M}, \theta\right) \tag{2.2}$$

This configuration of parameters is the most likely to have generated the data, hence the name maximum likelihood estimation. There is a curiosity to this approach though. Should a new data point come along, it is very likely that the maximum likelihood estimation would change. To an entirely new set of parameters! Thus, with no single configuration offering a perfect explanation of the system, it is better to think in probabilities again, though this time with distribution over parameters configurations given the observed data

$$P\left(\theta \mid \mathcal{M}, \mathcal{D}\right) \tag{2.3}$$

This is known as the posterior distribution. It is, however, not a distribution we have immediate access to. Fortunately Bayes and Laplace thought about this long ago (Bayes, 1763; Laplace, 1774) and can offer us some useful guidance. Following their advice, we must first understand a little more about joint and conditional densities.

2.1.2 Joint and conditional densities

When considering many variables simultaneously (only two here for the sake of argument), we write their joint distribution as

$$P(x_1, x_2) \tag{2.4}$$

[1]Surprise in statistics is defined as $-\ln P\left(\mathcal{D} \mid \mathcal{M}, \theta\right)$.

If these two variables share no dependence, i.e. whatever value x_1 takes, in no way affects the distribution of values x_2 takes, then the joint equation simplifies to

$$P(x_1, x_2) = P(x_1)P(x_2) \tag{2.5}$$

This says the joint probability is the product of the marginals[2]. Two variables are said to be *independent* if and only if their joint distribution factorises this way. For example, if x_1 is a variable for whether a social media savvy academic tweets, T, on a day, and x_2 is a variable for whether the academic drinks coffee, C, then, since coffee happens regardless of other activities, these events are independent i.e. the joint probability of a tweet and coffee[3] is the product of the marginals; $P(x_1 = T, x_2 = C) = \frac{3}{14}$ and $P(x_1 = T)P(x_2 = C) = \frac{1}{4} \cdot \frac{6}{7} = \frac{3}{14}$.

Variables don't have to be independent though. If x_2 is instead whether the academic published a paper that day, R, one might imagine tweets and papers occur together. Indeed, the probability of observing a tweet and a paper $P(x_1 = T, x_2 = R) = \frac{1}{40}$, but $P(x_1 = T)P(x_2 = R) = \frac{1}{4} \cdot \frac{1}{25} = \frac{1}{100}$, i.e. $P(x_1 = T, x_2 = R) \neq P(x_1 = T)P(x_2 = R)$. How can we reconcile this and relate joint distributions with marginal distributions?

The insight of Bayes and Laplace Bayes (1763); Laplace (1774) was that a joint probability distribution can still be factored into the product of two distributions, though one of them is a conditional distribution not a marginal distribution

$$P(x_1, x_2) = P(x_1 \mid x_2)P(x_2) = P(x_2 \mid x_1)P(x_1) \tag{2.6}$$

We see the first factor is still just the marginal distribution of x_1, but now the second is a distribution of x_2 *conditioned* on x_1 (or the other way round). In our case, $P(x_1 = T \mid x_2 = R) = \frac{5}{8}$, which then makes sense as $\frac{5}{8} \cdot \frac{1}{25} = \frac{1}{40}$. We note that in the case of independence $P(x_1 \mid x_2) = P(x_1)$ and so we recover the earlier result of equation 2.5.

[2]Termed marginal distributions as it can be obtained by marginalising over the joint e.g. $P(x_1) = \int P(x_1, x_2)dx_2$

[3]Approximate data for an unnamed academic.

2.1.3 Bayes theorem

Rearranging equation 2.6 gives Bayes theorem

$$P(x_2 \mid x_1) = \frac{P(x_1 \mid x_2)P(x_2)}{P(x_1)} \tag{2.7}$$

This equation tells us how to update our beliefs on x_2 given we have observed x_1. Let's use Bayes on our prolific tweeter. If we want to work out the probability of a paper given we have observed a tweet, $P(x_2 = R \mid x_1 = T)$, then we need to know what the three terms are on the right hand side of equation 2.7. Fortunately we have them all above; the prior $P(x_1 = T) = \frac{1}{4}$; the marginal $P(x_2 = R) = \frac{1}{25}$; the likelihood $P(x_1 = T \mid x_2 = R) = \frac{5}{8}$. We are all ready to get the answer

$$P(Paper \mid Tweet) = \frac{\frac{5}{8} \cdot \frac{1}{25}}{\frac{1}{4}} = \frac{1}{10} \tag{2.8}$$

This simple equation is all we needed to update our beliefs on whether a paper was published - our beliefs of a paper went from the prior $\frac{1}{25}$ to the posterior $\frac{1}{10}$. A modest change, but a change nonetheless[4]. Arguably more important that academics and twitter, exactly the same procedure can be applied to our modelling example - we can update our beliefs over parameters without ever having to observe them

$$P\left(\theta \mid \mathcal{M}, \mathcal{D}\right) = \frac{P\left(\mathcal{D} \mid \mathcal{M}, \theta\right) P(\theta \mid \mathcal{M})}{P(\mathcal{D} \mid \mathcal{M})} \tag{2.9}$$

2.1.4 Datasets

We often collate scalar variables $\{x_1, x_2 \cdots\}$ and write them as a vector, $\mathbf{x} = [x_1, x_2 \cdots]$. This is a useful representation as often observations are high dimensional. For example, images consist of pixels and so can be described as $\mathbf{x}$ where each x_i is a single pixel. The above relations all still hold when considering collections of variables (vector) as opposed to single variables at a time.

[4]Presumably the content of the tweets would be more informative.

In general we have access to more than just one observation vector $\mathbf{x}$. For example, we may want to learn from a set of images $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(N)}\}$, where each $\mathbf{x}^{(1)}$ is a photo of a galaxy. We describe this set of observations as a dataset $\mathcal{D}$, where

$$\mathcal{D} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(N)}\} \equiv \mathbf{x}^{1:\mathrm{N}} \tag{2.10}$$

We typically assume that each observation, $\mathbf{x}^{(i)}$, is a separate and independent measurement. In this case, the $\mathbf{x}^{(i)}$ are known as independent and identically distributed (i.i.d). This assumption makes our lives as data modellers much easier. Rather than considering the joint over all data points, we can exploit independence and factorise $\mathrm{P}_\theta\left(\mathbf{x}^{1:\mathrm{N}}\right)$ as $\prod_{\mathbf{x}\in\mathcal{D}} \mathrm{P}_\theta(\mathbf{x})$.

2.1.5 Latent variable models

So far the only unknowns (unobserved variables) we have discussed are the model parameters. We can, however, think of some unobserved variables as latent causes, rather than just parameters of a model. For example, when talking to someone, I hear words (observed variables) and it is my job to understand the other person's thinking (unobserved variables) that caused those words. Thus, we make a distinction about the unknowns in the world - there may be unknown parameters or unknown (hidden/ latent) causal factors. Formalising this, we have a generative model[5] relating unknown cause, $\mathbf{z}$, and unknown parameters, θ, to observation $\mathbf{x}$

$$\mathrm{P}_\theta(\mathbf{x}, \mathbf{z}) = \mathrm{P}_\theta(\mathbf{x} \mid \mathbf{z})\,\mathrm{P}_\theta(\mathbf{z}) \tag{2.11}$$

For further, and hierarchical, specifications of latent variables see appendix A.

Variables versus parameters

Even though we made a distinction between latent variables and parameters, in reality, they are both still just parts of the mechanics of the generative model and the separation is just a further specification of the model structure, $\mathcal{M}$. And so, like good Bayesians, we consider latent variables and parameters in the same light.

[5] We drop $\mathcal{M}$ for notational clarity and also adopt the cleaner $\mathrm{P}_\theta(\mathbf{z} \mid \mathbf{x})$ as opposed to $P(\mathbf{z} \mid \mathbf{x}, \theta)$.

We can, however, think differently. Rather than inferring both $\mathbf{z}$ and θ, we can instead *infer* the causes $\mathbf{z}$ and *learn* the parameters θ. This distinction has some relevance for the brain as we shall see later. The posterior we want is then

$$\mathrm{P}_\theta\left(\mathbf{z} \mid \mathbf{x}\right) = \frac{\mathrm{P}_\theta\left(\mathbf{x} \mid \mathbf{z}\right)\mathrm{P}_\theta\left(\mathbf{z}\right)}{\mathrm{P}_\theta\left(\mathbf{x}\right)} \tag{2.12}$$

As latent variable modellers, we have to consider whether our unobserved variables are global and influence all data observations e.g. our parameters θ, or local and are associated with each data observation e.g. each $\mathbf{z}$ in $\mathbf{z}^{1:\mathrm{N}}$. In the case of the local latent variables, we can assume that the conditional $\mathrm{P}_\theta\left(\mathbf{x}^{1:\mathrm{N}} \mid \mathbf{z}^{1:\mathrm{N}}\right)$ and prior $\mathrm{P}_\theta\left(\mathbf{z}^{1:\mathrm{N}}\right)$ factorise across data points - this then induces a factorisation in the posterior.

2.1.6 Types of learning

Bayes can be applied to a variety of different learning problems. The traditional split of learning is unsupervised learning, supervised learning and reinforcement learning. In this thesis we only consider the first two, and thus offer a little introduction here, with the key difference being whether data comes in pairs of inputs and outputs or inputs alone.

Supervised learning

Supervised learning considers both inputs and outputs, with the aim to learn good mappings from input, $\mathbf{x}$, to output[6], $\mathbf{t}$. Examples may be classification tasks where the output is class labels, or regression tasks where the output is real numbers.

Starting with a conditional model[7] from input to output $P_\theta(\mathbf{t} \mid \mathbf{x})$, there are two things to do - learning and making predictions. For learning, as described above, we are interested in $P(\theta \mid \mathcal{D})$ which we can get from Bayes rule, though many algorithms (including neural networks) just use simple maximum likelihood estimations on $\prod_{\mathbf{x},\mathbf{t}\in\mathcal{D}} P_\theta(\mathbf{t} \mid \mathbf{x})$. For prediction we are interested in $P(\mathbf{t} \mid \mathbf{x}) =$

[6] In supervised learning, $\mathcal{D} = \{(\mathbf{x}^{(1)}, \mathbf{t}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{t}^{(2)}), \cdots, (\mathbf{x}^{(N)}, \mathbf{t}^{(N)})\}$.

[7] One can also use joint models $P_\theta(\mathbf{x}, \mathbf{t})$, e.g. naive Bayes.

$\int P_\theta(\mathbf{t} \mid \mathbf{x})P(\theta \mid \mathcal{D})d\theta$, i.e. we marginalise out the weights. When θ is learned via maximum likelihood (θ^*), then $P(\mathbf{t} \mid \mathbf{x}) = P_{\theta^*}(\mathbf{t} \mid \mathbf{x})$.

We further introduce supervised learning with neural networks in chapter 3.

Unsupervised learning

Unsupervised learning considers input data alone, with the aim to learn its statistical regularities. This is exactly what we have been considering in the earlier parts of this chapter - defining a generative model with latent variables and parameters, then learning (inferring) the parameters (and latents) via a posterior distribution. Examples include clustering algorithms, dimensionality reduction and structure learning. For the rest of this chapter, our considerations lie primarily with unsupervised learning, though much is also applicable to supervised learning.

In summary, Bayes theorem offers a principled way to update beliefs, and thus a principled way to approach both learning and inference. Perhaps the brain is doing something similar.

2.2 Bayes and the brain

The Bayesian brain hypothesis (Knill and Pouget, 2004; Lee and Mumford, 2003) suggests exactly this - that the brain holds an underlying generative model of the world. How this may happen is something we will discuss later. Regardless, holding a generative model that maps latent causes to sensory observations is not enough; the hypothesis also states the brain acts as an inference machine inferring causes given sensory inputs. Holistically, perception equates inference (von Helmholtz, 1896) providing a posterior over causes given sensory data. To infer sensible causes, one needs an appropriate generative model with generative predictions that match sensory inputs. This means the brain must have learning mechanisms that try to match the generative model to sensory data[8].

[8]We apologise to aficionados for ignoring aspects of the Bayesian brain hypothesis beyond perception, for example we do not consider actions in this thesis and so skip information gain maximisation, Bayesian utility theory, planning as inference etc. A more thorough, yet still brief, review can be found in Gershman (2019).

Though 'perception as inference' is somewhat high-level, there is a reasonable amount of evidence for Bayesian inference in the brain. Early work showed participants Bayes optimally integrate information over two uncertain cues. For simple Gaussian distributions, participants precision weight cues (Jacobs and Fine, 1999; Knill and Saunders, 2003; Hillis et al., 2004). Even for highly non-Gaussian posteriors (Knill, 2003) participants also appear Bayesian. Bayes also predicts inappropriate priors will lead to perceptual biases, e.g. a bias for slow moving objects will lead to predictably incorrect perceptual inferences regarding motion; this was observed in Weiss et al. (2002). Other examples of Bayesian inferences in perceptual tasks are found in sensorimotor learning (Körding and Wolpert, 2004; Ernst and Banks, 2002).

Bayes also accounts for inferences in everyday cognition. In Griffiths and Tenenbaum (2006), participants' predictions on everyday events (e.g. lengths of marriages, movie grosses) were indistinguishable from optimal Bayesian models with access to the empirical distributions. Many other Bayesian interpretations of cognition have also been found (Tenenbaum et al., 2011; Griffiths et al., 2015; Gershman et al., 2015; Lake et al., 2015).

Away from psychology, Bayes has something to say for brain data - from Bayesian models fitting fMRI data (Behrens et al., 2007; Friston et al., 2003) to electrophysiological recordings of neurons (Beck et al., 2008; Roitman and Shadlen, 2002).

This sunny description of optimal Bayes in the brain is somewhat overdone - there are also numerous incidences of suboptimal perceptual decision making (Rahnev and Denison, 2018). Some can be reconciled with approximate Bayesian inference (Gershman et al., 2015) or efficient coding frameworks (Summerfield and Tsetsos, 2015). Nevertheless, evidence indicates the brain, in some capacity, acts as a Bayesian inference machine.

Holding an explicit generative model may not be necessary, the brain only needs to perform computations as if it has an internal generative model. Either way, brains consist of neurons connected by weighted synapses, along with neuromodulators and a variety of other cells. Following our distinction between variables and parameters,

the conventional wisdom is neurons encode variables and connection weights are the parameters of some model, be it generative or otherwise. Learning then becomes updating connection weights, and inference reconfiguring network activity.

Learning is only necessary if data is not accounted for. Inference on the other hand is always necessary as it is the process of understanding. Since learning is time consuming and sometimes swift decisions are necessary, ideally learning is done ahead of time and inference is all that's necessary. Indeed, learning should be designed to maximise the number of inferences one can make, even in novel situations. For brains, this means learning should optimise the neural representations for inference.

This brings us to a simple dichotomy[9] for the role of Bayes in the brain; Bayes may tell us about the *algorithms* behind learning and inference, but also the governing principles for *representations* that allow interesting inferences. Providing some understanding within this dichotomy is the central aim of this thesis.

2.2.1 Algorithms

We first consider how the brain may algorithmically implement Bayes. This seems simple enough - just compute the posterior distribution via Bayes theorem (equation 2.12); the product of the likelihood and prior all divided by the marginal. Unfortunately it is not so simple, for a reason both neuroscientists and machine learners care about, and another only neuroscientists care about.

We very briefly consider the latter first - how do you encoder a probability distribution with neurons? Many researchers have offered their opinions here (Pouget et al., 2013), with ideas ranging from representing the sufficient statistics alone (Friston, 2005), via sampling (Hoyer and Hyvärinen, 2003; Lee and Mumford, 2003), through population codes (Zemel et al., 1998; Ma et al., 2006).

The majority of these ideas are beyond the scope of this thesis, and so we move on to the former question; the algorithms behind the Bayesian update. Here neuroscientists face the same problem as statisticians and machine learners

[9] An imperfect dichotomy. Useful for the purposes of this thesis though.

- computing the prior and likelihood are easy but marginal distribution generally requires an intractable integral (see appendix A).

$$\mathrm{P}_\theta(\mathbf{x}) = \int \mathrm{P}_\theta(\mathbf{x}, \mathbf{z})\, d\,\mathbf{z} = \int \mathrm{P}_\theta(\mathbf{x} \mid \mathbf{z})\, \mathrm{P}_\theta(\mathbf{z})\, d\,\mathbf{z} \tag{2.13}$$

Should the search space of $\mathbf{z}$ be particularly large, or unpleasant non-linearities appear in the distributions, this integral gets difficult. Though still a major problem among those more formal disciplines, several techniques tackling this problem have been developed. One is of particular interest to neuroscience.

Variational free energy

The approach is to give up on exact Bayes and instead learn an approximation, $\mathrm{Q}_\phi\left(\mathbf{z}^{1:\mathrm{N}} \mid \mathbf{x}^{1:\mathrm{N}}\right)$, to the posterior[10]. We're not happy with any old distribution - we want this approximation to as closely match the true posterior as possible. In this vein, we optimise a distance measure between the approximate and true distribution. The standard measure used is the Kullback-Leibler divergence, D_{KL}, (Kullback and Leibler, 1951), which after substituting Bayes theorem (appendix A) we obtain

$$\begin{aligned}\mathrm{D}_{\mathrm{KL}}(\mathrm{Q}_\phi\left(\mathbf{z}^{1:\mathrm{N}} \mid \mathbf{x}^{1:\mathrm{N}}\right) \| \mathrm{P}_\theta\left(\mathbf{z}^{1:\mathrm{N}} \mid \mathbf{x}^{1:\mathrm{N}}\right)) \\ = \sum_{\mathbf{x}\in\mathcal{D}} \ln \mathrm{P}_\theta(\mathbf{x}) + \mathbb{E} \ln \mathrm{Q}_\phi(\mathbf{z} \mid \mathbf{x}) - \mathbb{E} \ln \mathrm{P}_\theta(\mathbf{x}, \mathbf{z})\end{aligned} \tag{2.14}$$

Where $\mathbb{E}$ is an expectation under $\mathrm{Q}_\phi(\mathbf{z} \mid \mathbf{x})$, and the factorisation across data samples turned logs of products into sums of logs. Since $\ln \mathrm{P}_\theta(\mathbf{x})$ is independent of $\mathbf{z}$, optimising the last two terms (collectively known as the evidence lower bound, $ELBO$, or negative variational free energy F) is enough to minimise the divergence between the true and approximate posteriors

$$ELBO = -F = \sum_{\mathbf{x}\in\mathcal{D}} \mathbb{E} \ln \mathrm{Q}_\phi(\mathbf{z} \mid \mathbf{x}) - \mathbb{E} \ln \mathrm{P}_\theta(\mathbf{x}, \mathbf{z}) \tag{2.15}$$

By optimising these quantities, in which $\mathrm{P}_\theta(\mathbf{z} \mid \mathbf{x})$ disappears and we don't have to calculate $\mathrm{P}_\theta(\mathbf{x})$, the problem has then been translated from an integration problem to an optimisation problem. These often turn out to be easier.

[10]This approximation can be parameterized by ϕ. We also assume a factorisation across data-points i.e. $\mathrm{Q}_\phi\left(\mathbf{z}^{1:\mathrm{N}} \mid \mathbf{x}^{1:\mathrm{N}}\right) = \prod_{\mathbf{x}\in\mathcal{D}} \mathrm{Q}_\phi(\mathbf{z} \mid \mathbf{x})$

Feynman (Feynman, 1972) introduced variational free energy in his path integral formulation of quantum physics. In the late 80s (Anderson et al., 1987) and early 90s (Hinton and van Camp, 1993; Neal and Hinton, 1998), statisticians and machine learners realised it as a fruitful approach and it remains so today. If the brain uses variational methods, then the brain not only holds a generative model $\mathrm{P}_\theta(\mathbf{x} \mid \mathbf{z})$ going from causes to observations, but also a recognition model $\mathrm{Q}_\phi(\mathbf{z} \mid \mathbf{x})$ from observations to causes. Furthermore, to make it all work, the brain should implement a scheme of minimising free energy for both inference and learning. This is known as the free energy principle for the brain (Friston, 2005, 2010).

The free energy principle can be extended to include actions (Friston et al., 2009); just as perception minimises free energy during inference, actions can be viewed as minimising surprise and free energy by changing the state of the world to better fit one's internal model. This is known as active inference. Beyond brains, free energy minimisation can be related to any system possessing a Markov blanket, a separation between internal and external states. Here, any system maintaining such a separation will appear to minimise free energy and therefore will be performing Bayesian inference (Friston, 2013).

This principle of free-energy optimisation, though providing an overarching framework for viewing brain (and life!) processes, is still somewhat removed from brains and neurons. For this, we need to consider the specifics of the generative and approximate distributions.

Predictive coding

It's a good idea to consider the simple things first. Let's see the very simplest variational approximation along with a simple generative model rather magically translates into a neural network (Rao and Ballard, 1999; Friston, 2005; Bogacz, 2017). Part of the magic is that seemingly nowhere in the formulation does anything like neural networks appear (see appendix A for further details).

We use a Gaussian conditional and a Gaussian prior for the generative model; $\mathrm{P}_\theta(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\mathbf{x}; \Theta_x \mathbf{z}, \Sigma_x)$, $\mathrm{P}_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \Theta_z, \Sigma_z)$. The conditional distribution is

parameterized so that its mean is a linear transformation of $\mathbf{z}$, via parameters Θ_x. The mean of the prior is a vector of θ_z. We assume both covariances, Σ, are the identity matrix. The inference distribution is a delta function[11], $Q_\phi(\mathbf{z}) = \delta(\mathbf{z} - \boldsymbol{\phi})$. Substituting in equation 2.15 leads to the following function to optimise

$$F = \sum_{\mathbf{x}, \boldsymbol{\phi} \in \mathcal{D}} (\mathbf{x} - \Theta_x \boldsymbol{\phi})^T (\mathbf{x} - \Theta_x \boldsymbol{\phi}) + (\boldsymbol{\phi} - \theta_z)^T (\boldsymbol{\phi} - \theta_z) + const \tag{2.16}$$

Where *const* is a collation of constants. We perform a two stage process (like expectation maximisation (Dempster et al., 1977) as described in appendix A), where we alternate between 'inferring' variables and 'learning' parameters. Since the variables to infer, $\boldsymbol{\phi}$, are local to each data-point, $\mathbf{x}$, we infer each variable, $\boldsymbol{\phi}$, at a time via gradient descent

$$\dot{\boldsymbol{\phi}} = -\boldsymbol{\varepsilon}_\phi + \Theta_x^T \boldsymbol{\varepsilon}_x \tag{2.17}$$

where $\boldsymbol{\varepsilon}_x = \mathbf{x} - \Theta_x \boldsymbol{\phi}$ and $\boldsymbol{\varepsilon}_\phi = \boldsymbol{\phi} - \theta_z$. The magic has happened. The seemingly innocuous equation for inference turns out to be implementable in a recurrent neural network as shown in Figure 2.1. The unique property of the network is that there are two populations of neurons, those that encode variables, $\boldsymbol{\phi}$, and those that encode prediction errors, $\boldsymbol{\varepsilon}$; hence the name predictive coding networks. Prediction errors change representations in higher layers to produce better explanations of the lower levels - perception as inference.

Once the network reaches equilibrium (guaranteed as it has an associated Lyapunov function F), the weights can be updated again via gradient descent.

[11] One can also make the Laplace approximation.

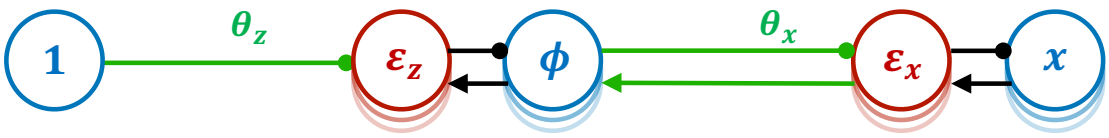

Figure 2.1: Architecture of the network. Arrows and lines ending with circles denote excitatory and inhibitory connections respectively. Connections without labels have weights fixed to 1. The node furthest to the left has activity 1, with the weights, θ_z, acting as prior beliefs in the mean of $\mathbf{z}$.

Stochastic gradient descent using a sub-sample of the data in $\mathcal{D}$ works as it provides unbiased gradient estimators of F. It also offers biological plausibility as not all the data need be considered simultaneously. Like inference, even a single data sample at a time will do

$$\Delta\Theta_x = -\,\boldsymbol{\varepsilon}_x\,\boldsymbol{\phi}^T \qquad\qquad \Delta\theta_z = -\,\boldsymbol{\varepsilon}_{\boldsymbol{\phi}} \tag{2.18}$$

These learning rules are Hebbian (Hebb, 1949), a product of variable and error neuron activities. The beauty of this approach is that optimisation of a single function gives both inference and learning, and all implementable, one data point at a time, in a biological neural network to boot.

This was vanilla predictive coding; we assumed no neuronal activation function, only considered a hierarchy of one latent variable, and assumed identity covariances. These extensions follow the same steps as detailed above and additionally offer insights into other brain mechanisms such as attention. We shall not do them here, however, but instead stop our discussion as predictive coding and its neural evidence will be discussed further in chapters 4 and 5.

Helmholtz machine

A disadvantage of the above procedure, and indeed many variational methods, is that inference requires solving a differential equation. This may be easy in simple circumstances, but for complex models the iterative procedure can be prohibitively expensive. To circumvent this, other models attempting to optimise F, parametrise their recognition distribution so that inference is amortised and performed in a single pass. The trade-off is that parameters of the generative model *and* parameters of the inference model are to be learned. Such inference models have to learn how to infer.

The earliest examples, that also offer neural insights, are Helmholtz machines and the Wake-Sleep algorithm (Dayan et al., 1995; Hinton et al., 1995). Here parameters of the generative model are adjusted during a 'wake' phase to match the observed data, and parameters of the recognition model are adjusted during

a 'sleep' phase to better match generated samples. For certain neuron activation functions, all learning is local. The wake-sleep algorithm, though, uses different objective functions for sleep and wake, which together do not correspond to F and therefore are not a bound on the marginal likelihood.

More modern variants of the Helmholtz machine have solved this issue and optimise F for both the recognition and generative parameters (Kingma and Welling (2013); see appendix A). These deep learning models, though not necessarily offering insights into the algorithms of biological learning, do offer a means of training complicated generative models, of which we can examine their learned representations.

2.2.2 Representations

Away from the particular algorithmic implementation of Bayes in the brain, thinking of generative models and Bayes pays dividends when considering representations and the principles behind their learning. Let us first consider what representations and representation learning is all about (Bengio et al., 2012).

A representation is an encoding of the world (or the task you face), and a good one fulfils certain desiderata. 1) It should contain the relevant information for downstream processes; your representation should distinguish between pigs and cows if you want to drink milk, but not if you're deciding between meat or vegetables. 2) It should generalise; when learning Siamese is a type of cat, the learned representations should assist on classifying Burmese as cats for the first time. Equally, representations for how London Underground stations connect to each should be useful for the New York City Subway. 3) It should reflect an underlying structure of the world/data; representations for doors should know they connect rooms. A neat example in machine learning showed that vectors in word embedding spaces (Mikolov et al., 2013) have semantic meaning e.g. $king - queen + girl = boy$.

These desiderata, though not mutually exclusive, can be summarised as **abstracting** away irrelevant features, learning **generalisable** features and repre-

senting **structure**. Bayesian thinking offers some insights into each of these three considerations.

The first can be illuminated when considering the optimisation of equation 2.15, which can be rewritten as

$$-\ln \mathrm{P}_\theta\left(\mathbf{x} \mid \mathbf{z}\right) + \mathrm{D}_{\mathrm{KL}}(\mathrm{Q}_\phi\left(\mathbf{z} \mid \mathbf{x}\right) \| \, \mathrm{P}_\theta\left(\mathbf{z}\right)) \tag{2.19}$$

Since it comes in two terms, there is a trade-off in the optimisation process. The learned parameters, ϕ and θ, must provide representations, $\mathbf{z}$, which not only accurately accounts for the data, via $\ln \mathrm{P}_\theta\left(\mathbf{x} \mid \mathbf{z}\right)$, but are also simple, via $\mathrm{D}_{\mathrm{KL}}(\mathrm{Q}_\phi\left(\mathbf{z} \mid \mathbf{x}\right) \| \, \mathrm{P}_\theta\left(\mathbf{z}\right))$. This second term adds a cost to encoding information far from the prior, and thus, with a learned prior or otherwise, regularises the representation to only encode relevant information. It is not just in representation learning that Bayes prefers simpler solutions, a similar Occam's Razor like effect occurs in model comparison (MacKay, 1992).

We will consider generalisation properly in chapter 6, so let us instead consider the final desiderata which, in fact, supersedes much of the other two, so we get a sneak peek at representations for generalisation nevertheless. This superseding is because we take an overarching and philosophical view linking generative modelling with the way physicists think in symmetries.

Symmetrical thinking

The notion of symmetries date long before the discipline of physics. The term συμμετρία, symmetria, comes from σύν (with) and μέτρον (measure) and originally meant commensurability, though it later took the meaning of a union of elements into a unitary whole (Brading and Castellani, 2003). This last notion is more familiar to us - a whole human consists of two halves, a hexagonal honeycomb of six triangles. A more precise and general definition of a symmetry is a transformation that leaves certain properties invariant, e.g. we look similar in a mirror.

Symmetries first found their way into physics through crystallography, with differing symmetries of crystals leading to differing scattering properties. Later

Einstein realised the importance of global space-time symmetries in the special theory of relativity and of local symmetries in his general theory of relativity. It was through the work of Noether (Noether, 1920), that symmetries in physics took their more modern form; symmetries of governing laws as opposed to symmetries of physical objects. Noether showed that any global, continuous symmetry of the action (integral of the Lagrangian) is associated with a conservation law e.g. spatial translation symmetry leads to conservation of momentum, time translation leads to conservation of energy. This way of thinking led to numerous breakthroughs in physics - from predictions of new particles to unification of physical laws (Gell-Mann, 1962; Weinberg, 1967).

Moving on from physics, it is remarkable the number of things in nature that are invariant post transformation. If I move a chair the table stays in the same place, if I paint a cat it still retains it size and shape. It is here that we make the distinction between *invariant* and *equivariant*. An invariant is something that stays the same after a transformation (like the size of the cat). An equivariant is something that may change but nevertheless retains information about the transformation. This distinction is best made with a little formalism, notably the formalism of groups[12].

A transformation, g, is associated with a group, G. A function (of feature), f, is invariant under G if for all g the following is true; $f(g(\cdots)) = f(\cdots)$ i.e. we make a transformation and our feature, f, is still the same (Figure 2.2 left). An equivariance, however, satisfies the following property; $f(g(\cdots)) = g'f(\cdots)$, where g' is a transformation in group G' (Figure 2.2 center). This says a feature post a transformation is a transformation post a feature i.e. f is no longer invariant, but it preserves the structure of the transformation g - it is equivariant. In equivariant maps, some subsets of features may be invariant to some transformations, and other not. Nonetheless the overall representation must contain all information post transformation. We note that G and G' may be the same group, should $f(\cdots)$ and $g(\cdots)$ be in the same domain.

[12]We do not give an introduction to group theory there, nor do we use full group theory formalism in the following discussions. We instead only try to offer intuition.

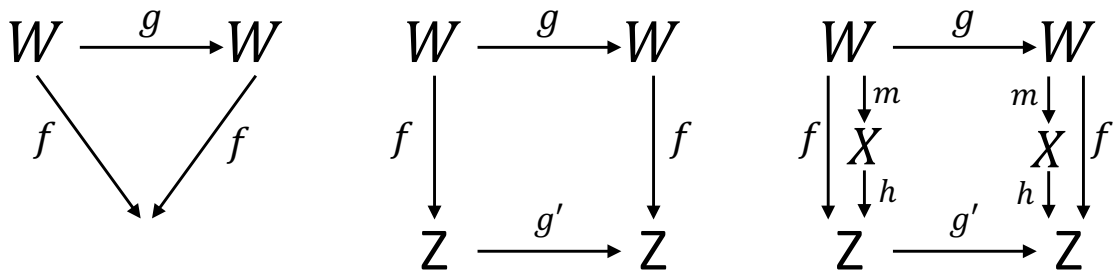

Figure 2.2: Left/center/right: Invariance/ equivariance/ link to generative models. Center: equivariance condition can be understood via a commutative diagram.

How does this relate to neural representations?[13]. If an equivariant map, f, is from the space of world states, W, to the space of an individual's brain states, Z, then when transitioning through the world, some features will stay invariant e.g. neurons encoding what's in the room, and other features may be equivariant e.g. neurons encoding where in the room (see grid cells in chapter 6).

How does this all relate to generative modelling and Bayes? Let us believe there is an abstract world state, W, and a true generative model, m, that produces sensory observations X (Figure 2.2 right). As Bayesian inference machines, h, it is our job to infer latent variables, Z, on the basis of observations X. We thus define the map from W to Z as $f = h \oplus m$. If the brain representations understand the world, then Z will reflect the same structure as W, i.e. we want to learn h so that f is an equivariant map.

It is natural to think of G and the world state representation W to be decomposable into subunits G_i and W_i where G_i acts on W_i alone and leaves other W_j unchanged e.g. G_c acts on the colour of a cat alone, or G_x and G_y may act on its x and y position. Again, if Z reflects the same structure as W, G' and Z representations follow a similar decomposition.

Disentangled representations

This raises an interesting point about representations. Representations that transform in different ways should be represented separately or disentangled from one another. This way of thinking imposes constraints on the types of representations

[13]The discussion of symmetries and their role in representations follows Higgins et al. (2018)

one should find. In probability parlance, it implies that representations should be factorised i.e.

$$P(\mathbf{z}) = P(\mathbf{z}_1, \mathbf{z}_2) = P(\mathbf{z}_1)P(\mathbf{z}_2) \tag{2.20}$$

Where $\mathbf{z}_1$ and $\mathbf{z}_2$ are each vector representations. Separate representations for things that transform differently seems sensible enough, though it's in new situations where disentanglement is particularly useful. Since new world states are just previously unseen configurations of transformations, disentangled representations allow easy understanding as combinations of seen-before representations; this is generalisation. Imagination is facilitated in exactly the same way.

Disentangled representations are also suitable to learn off. If you feel well after eating cabbages and carrots from brown soil and unwell after cabbages from green soil, then do you eat a carrot from green soil? Correctly assigning credit (blame) to the green soil is easily learned with disentangled representations (Higgins et al., 2017b), whereas entangled representations often only assign credit to the specific conjunction of green soil and cabbage.

Numerous techniques have been developed to learn such disentangled representations, from factor analysis techniques (Jolliffe, 1986; Schmidhuber, 1992) to generative adversarial networks (Chen et al., 2016) and VAEs (Higgins et al., 2017a) (Figure 2.3). Recent findings in the brain also suggest such disentangled representations; when monkeys see parameterized faces, individual neurons (from equivalent area to IT in humans; Chang and Tsao (2017)) are tuned to face axes, and blind to deviations orthogonal to this axis.

Basis functions

Though the above presented an abstract formalisation of representational structure - separate representations for the things in the world that transform differently - we can get more specific about the component parts to building such representations. To help on this quest, let us consider images. We start with the basic assumption

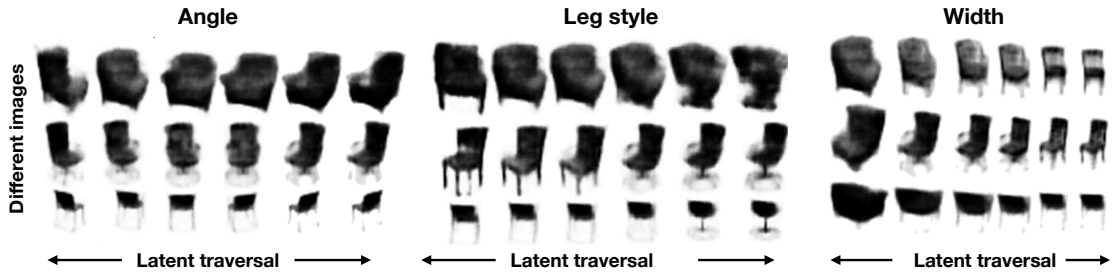

Figure 2.3: Latent traversals show disentangled representations. Three input images (top/middle/bottom) are encoded, then a latent variable is changed while keeping others fixed. Left/center/right; three different latents. We see that each latent corresponds to a single meaningful transformation - angle, leg style, and width. Figure adapted from Higgins et al. (2017a).

that any image, $I(x, y)$, can be represented as a linear superposition of basis functions $\phi_i(x, y)$

$$I(x, y) = \sum_i a_i \phi_i(x, y) \tag{2.21}$$

The basis functions are the same for each image, but their weightings, a_i, vary. Bases can come in different forms. Mutually orthogonal basis that capture the directions of maximal variance (principal component analysis) often lead to periodic representations (Field, 1987). Enforcing sparsity on a_i leads to Gabor filters (Figure 2.4A; Olshausen and Field (1996)). Independent component analysis also leads to Gabor filters (Bell and Sejnowski, 1997). These latter two are of particular interest to neuroscience as similar filters have been observed as receptive fields of simple cells in primary visual cortex (Hubel and Wiesel, 1959). Similarly, predictive coding networks additionally exhibit extra-classical receptive fields (Figure 2.4B; Rao and Ballard (1999)). These methods can often be interpreted with generative models and relate to analysis by synthesis.

Like disentangled representations, basis functions allow easy interpretation of new information in the context of previously learned knowledge (bases). They do not have to be for images, nor just for sensory representations. Bases can represent arbitrary information, they just need to span the space. Appropriately inferring their weightings, a_i, then becomes key to understanding.

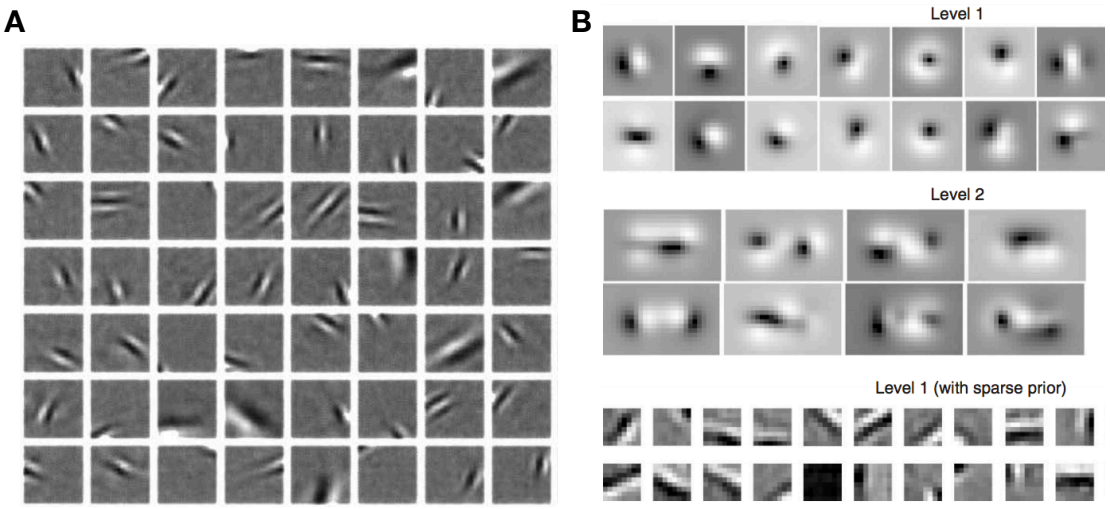

Figure 2.4: Receptive fields of networks trained on natural images. A) Field (1987). B) Rao and Ballard (1999).

Part I

Algorithms for learning

3

Learning in Artificial and Biological Neural Networks

Contents

In the past few years, computer programs using deep learning have achieved impressive results in complex cognitive tasks that were previously only in the reach of humans. These tasks include processing of natural images and language (Lecun et al., 2015), or playing arcade and board games (Mnih et al., 2015; Silver et al., 2017). Since these recent deep learning applications use extended versions of classic artificial neural networks (ANNs; Rumelhart et al. (1986)), their success has inspired studies comparing information processing in ANNs and the brain. It has been demonstrated that when ANNs learn to perform tasks such as image classification or navigation, the neurons in their layers develop representations similar to those seen in brain areas involved in these tasks, such as receptive fields across the visual hierarchy or grid cells in the entorhinal cortex (Banino et al., 2018; Whittington et al., 2018; Yamins and DiCarlo, 2016). This suggests that the brain may use

analogous algorithms. Furthermore, thanks to current computational advances, ANNs can now provide useful insights on how complex cognitive functions are achieved in the brain (Bowers, 2017).

A key question that remains open is how the brain could implement the error back-propagation algorithm used in ANNs. This algorithm describes how the weights of synaptic connections should be modified during learning, and its attractiveness, in part, comes from prescribing weight changes that reduce errors made by the network, according to a theoretical analysis. Although ANNs were originally inspired by the brain, the modification of their synaptic connections, or weights, during learning appears biologically unrealistic (Crick, 1989; Grossberg, 1987).

In this chapter we first provide a brief overview of how the back-propagation algorithm is used to train ANNs and discuss why it was considered biologically implausible.

3.1 Artificial Neural Networks and Error Back-Propagation

To effectively learn from feedback, the synaptic connections often need to be appropriately adjusted in multiple hierarchical areas simultaneously. For example, when a child learns to name letters, the incorrect pronunciation may be a combined result of incorrect synaptic connections in speech, associative, and visual areas. When a multi-layer ANN makes an error, the error back-propagation algorithm appropriately assigns credit to individual synapses throughout all levels of hierarchy and prescribes which synapses need to be modified and by how much.

How is the back-propagation algorithm used to train ANNs? The algorithm is trained on a set of examples, each consisting of an input pattern, $\mathbf{s}$, and a target pattern, $\mathbf{t}$, i.e. we have a a dataset

$$\mathcal{D} = \{(\mathbf{s}^{(1)}, \mathbf{t}^{(1)}), (\mathbf{s}^{(2)}, \mathbf{t}^{(2)}), \cdots, (\mathbf{s}^{(N)}, \mathbf{t}^{(N)})\} \tag{3.1}$$

For each such pair, the network first generates its prediction based on the input pattern and then the synaptic weights are modified to minimise the difference

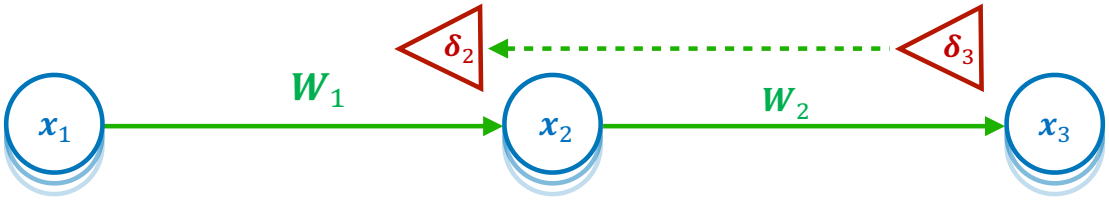

Figure 3.1: Artificial neural networks and backprop. A) Layers of neuron-like nodes are represented by sets of stacked blue circles. Feedforward connections are indicated by green arrows. Backpropagating error signals are shown as red triangles.

between the target and the predicted pattern. To determine the appropriate modification, an error term is computed for each neuron throughout the network. This describes how the activity of the neuron should change to reduce the discrepancy between the predicted and target pattern. Each weight is modified by an amount determined by the product between the activity of the neuron it projects from and the error term of the neuron it projects to.

This can be described mathematically. We consider a conventional ANN where each node receives a weighted sum of all the nodes from the previous layer (Figure 3.1). The input layer, $\mathbf{x}_1$, is first set to be the input pattern, $\mathbf{s}$, and then a prediction is made by propagating the activity through the layers according to the following

$$\mathbf{x}_l = W_{l-1} f\left(\mathbf{x}_{l-1}\right) + \mathbf{b}_l \tag{3.2}$$

Where $\mathbf{x}_l$ is a vector denoting neurons in layer l and W_{l-1} is a matrix of synaptic weights from layer $l-1$ to layer l. An activation function f is applied to each neuron to allow for nonlinear computations. The activation function can be any function, but typically sigmoid or rectified non-linearities are used. The activation function is applied to each element of the vector argument to produce a vector of the same dimension. We define the activity of a nodes slightly differently from the standard ANN, which has the activation function after the weights are applied. We do this as it makes the learning rule simpler, but there is no heuristic difference between the two formulations.

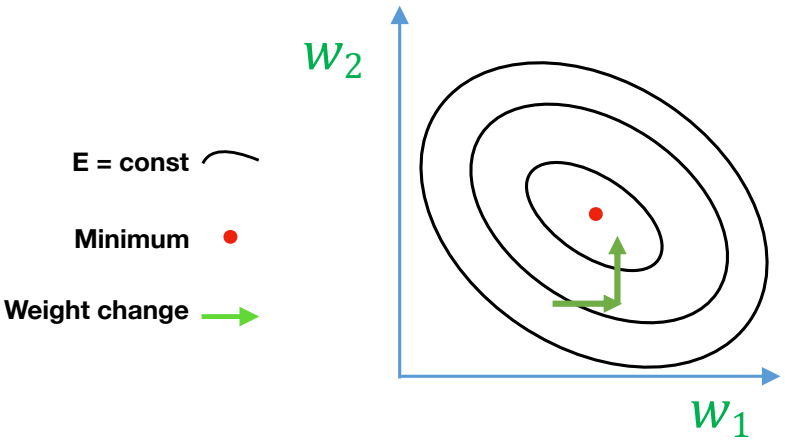

Figure 3.2: Schematic of the directions of two consecutive weight modifications (thick arrows) in the space of weights (for simplicity, only two dimensions are shown). Contours show points in weight space with equal cost function values

During learning, the synaptic connections are modified to minimise a cost function quantifying the discrepancy between the predicted and target patterns. An example function for regression is

$$E = \frac{1}{2}(\mathbf{t} - \mathbf{x}_L)^T(\mathbf{t} - \mathbf{x}_L) \tag{3.3}$$

The weights are modified in the direction of steepest decrease (or gradient) of the cost function (Figure 3.2)

$$\Delta W_a = \boldsymbol{\delta}_{a+1} f(\mathbf{x}_a)^T \tag{3.4}$$

$$\boldsymbol{\delta}_a = \begin{cases} \mathbf{t} - \mathbf{x}_a & \text{if } a = L \\ \left(W_a^T \boldsymbol{\delta}_{a+1}\right) \odot f'(\mathbf{x}_a) & \text{if } a < L \end{cases} \tag{3.5}$$

Where $\boldsymbol{\delta}_{l+1}$ is a vector of error terms associated with neurons $\mathbf{x}_{l+1}$. The error terms for the last layer L are defined in Equation 3.5 (top) as the difference between the target activity t and the predicted activity. Thus, the error of an output neuron is positive if its target activity is higher than the predicted activity. For the earlier layers, the errors are computed according to Equation 3.5 (bottom) as a sum of the errors of neurons in the layer above weighted by the strengths of their connections (and further scaled by the derivative of the activation function; in Equation 3.5

(bottom) the $\odot$ denotes element-wise multiplication). For example, an error of a hidden unit is positive if it sends excitatory projections to output units with high error terms, so increasing the activity of such a hidden neuron would reduce the error on the output. Once the errors are computed, each weight is changed according to Equation 3.4 in proportion to the product of the error term associated with a postsynaptic neuron and the activity of a presynaptic neuron.

Although the described procedure is used to train ANNs, analogous steps may take place during learning in the brain. For example, in the case of the child naming letters mentioned above, the input pattern corresponds to an image of a letter. After seeing an image, the child makes a guess at the name (predicted pattern) via a neural network between visual and speech areas. On supervision by his or her parent of the correct pronunciation (target pattern), synaptic weights along the processing stream are modified so that it is more likely that the correct sound will be produced when seeing that image again.

3.2 Biologically Questionable Aspects of the Back-Propagation Algorithm

Although the algorithmic process described above appears simple enough, there are a few problems with implementing it in biology. Below, we briefly discuss three key issues.

Lack of Local Error Representation

Conventional ANNs are only defined to compute information in a forward direction, with the back-propagating errors computed separately by an external algorithm. Without local error representation, each synaptic weight update depends on the activity and computations of all downstream neurons. Since biological synapses change their connection strength based solely on local signals (e.g., the activity of the neurons they connect), it appears unclear how the synaptic plasticity afforded by the back-propagation algorithm could be achieved in the brain. Historically, this is a major criticism; thus it is a main focus of our review article.

Symmetry of Forwards and Backwards Weights

In ANNs, the errors are back-propagated using the same weights as those when propagating information forward during prediction. This weight symmetry suggests that identical connections should exist in both directions between connected neurons. Although bidirectional connections are significantly more common in cortical networks than expected by chance, they are not always present Song et al. (2005). Furthermore, even if bidirectional connections were always present, the backwards and forwards weights would still have to correctly align themselves.

Unrealistic Models of Neurons

ANNs use artificial neurons that send a continuous output (corresponding to a firing rate of biological neurons), whereas real neurons use spikes. Generalising the back-propagation algorithm to neurons using discrete spikes is not trivial, because it is unclear how to compute the derivative term found in the back-propagation algorithm. Away from the back-propagation algorithm, the description of computations inside neurons in ANNs is also simplified as a linear summation of inputs.

3.3 Models of Biological Back-Propagation

Each of the above-mentioned issues has been investigated by multiple studies. The lack of local error representation has been addressed by early theories by proposing that the errors associated with individual neurons are not computed, but instead the synaptic plasticity is driven by a global error signal carried by neuromodulators (Mazzoni et al., 1991; Williams, 1992; Unnikrishnan and Venugopal, 1994; Seung, 2003). However, it has been demonstrated that learning in such models is slow and does not scale with network size (Werfel et al., 2005).

The criticism of weight symmetry has been addressed by demonstrating that even if the errors in ANNs are back-propagated by random connections, good performance in classification tasks can still be achieved (Lillicrap et al., 2016; Zenke and Ganguli, 2018; Mostafa et al., 2018; Liao et al., 2016; Baldi and Sadowski, 2016). This being said, there is still some concern regarding this issue (Bartunov et al.,

2018). With regard to the biological realism of neurons, it has been shown that the back-propagation algorithm can be generalised to neurons producing spikes (Sporea and Grüning, 2013) and that problems with calculating derivatives using spikes can be overcome (Zenke and Ganguli, 2018). Furthermore, it has been proposed that when more biologically realistic neurons are considered, they themselves may approximate a small ANN in their dendritic structures (Schiess et al., 2016).

We leave it to this thesis to reveal the first model demonstrating backpropagation was biologically plausible (chapter 4; (Whittington and Bogacz, 2017)), and then later integrate that model along with others in an overall scheme of energy minimisation (chapter 5).

4

Predictive coding networks implement back-propagation

Contents

4.1 Introduction

As discussed in chapter 3, it has not been known if natural neural networks could employ an algorithm analogous to the back-propagation used in ANNs. In ANNs, the change in each synaptic weight during learning is calculated by a computer as a complex, global function of activities and weights of many neurons (often not connected with the synapse being modified). In the brain however, the network must perform its learning algorithm locally, on its own without external influence, and the change in each synaptic weight must depend only on the activity of pre-synaptic and post-synaptic neurons. This led to a common view of the biological implausibility

of this algorithm (Crick, 1989), e.g. 'despite the apparent simplicity and elegance of the back-propagation learning rule, it seems quite implausible that something like equations [...] are computed in the cortex' (p. 162) (O'Reilly and Munakata, 2000).

In this chapter we show that backpropagation could be implemented in biological hardware[1]. The proposed model is inspired by predictive coding (Rao and Ballard, 1999; Friston, 2003, 2005). The predictive coding framework describes a network architecture in which learning has a particularly simple neural implementation. The distinguishing feature of the predictive coding models is that they include additional nodes encoding the difference between the activity on a given level and that predicted by the higher level, and that these prediction errors are propagated through the network (Rao and Ballard, 1999; Friston, 2005). Patterns of neural activity similar to such prediction errors have been observed during perceptual decision tasks (Summerfield et al., 2006, 2008). In this paper we show that when the predictive coding model is used for supervised learning, the prediction error nodes have activity very similar to the error terms in the back-propagation algorithm. Therefore, the weight changes required by the back-propagation algorithm can be closely approximated with simple Hebbian plasticity of connections in the predictive coding networks.

4.1.1 Predictive coding for supervised learning

We first propose a probabilistic model for supervised learning, then we describe inference in the model, its neural implementation, and finally learning of model parameters.

Probabilistic model

Figure 4.1 shows a structure of a probabilistic model that parallels the architecture of the ANN shown in Figure 3.1. It consists of L layers of variables, such that the variables on level l depend on the variables on level $l-1$. It is important to emphasize that Figure 4.1 does not show the architecture of the predictive

[1] We follow Whittington and Bogacz (2015, 2017) - the first demonstration of its plausibility.

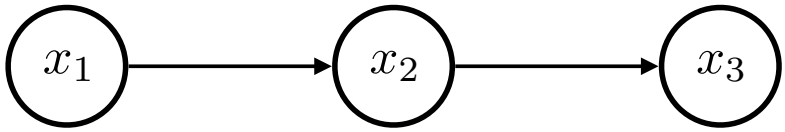

Figure 4.1: Structure of the probabilistic model. Circles denote random variables, while arrow denote dependencies between them.

coding network, but only the structure of underlying probabilistic model - as we will see below, the inference in this model can be implemented by a network with architecture shown in Figure 4.2.

By analogy to ANNs, we assume a dataset $\mathcal{D} = \{(\mathbf{s}^{(1)}, \mathbf{t}^{(1)}), \cdots, (\mathbf{s}^{(N)}, \mathbf{t}^{(N)})\}$ and that variables on the first level $\mathbf{x}_1$ are fixed to the input sample $\mathbf{s}$ and, if applicable (i.e. during learning), variables on the last level $\mathbf{x}_L$ are fixed to the output sample $\mathbf{t}$. Let us assume the following relationship between the random variables on adjacent levels:

$$P\left(\mathbf{x}_l \mid \mathbf{x}_{l-1}\right) = \mathcal{N}\left(\mathbf{x}_l; \boldsymbol{\mu}_l, \Sigma_l\right) \tag{4.1}$$

In the above equation $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \Sigma)$ is the probability density of a multivariate normal distribution with mean $\boldsymbol{\mu}$ and variance Σ. The mean on level l is a function of the values on the earlier level, analogous to the input to a node in ANN (cf. Equation 3.2):

$$\boldsymbol{\mu}_l = \Theta_{l-1} f\left(\mathbf{x}_{l-1}\right) + \mathbf{b}_l \tag{4.2}$$

In the above equation Θ_{l+1} are the parameters describing the dependence on the random variables and $\mathbf{b}_l$ is an additive bias. For simplicity in this paper we do not consider how Σ_l are learned (Friston, 2005; Bogacz, 2017), but treat them as fixed parameters.

Inference

Let us now move to describing the inference of the hidden random variables. Since our data is i.i.d., our joint distribution factorises over data samples

$$P(\mathbf{x}_L^{1:N}, ..., \mathbf{x}_2^{1:N} \mid \mathbf{x}_1^{1:N}) = \prod_{\mathbf{s},\mathbf{t} \in \mathcal{D}} P(\mathbf{x}_L, ..., \mathbf{x}_2 \mid \mathbf{x}_1) \tag{4.3}$$

Due to this factorisation, we can consider a single data sample at a time (as per chapter 2, i.e. the latter F is per data sample free energy). As a simplification to full inference, we instead choose to find the most likely values of all unconstrained random variables in the model, which maximize the probability $P(\mathbf{x}_L, ..., \mathbf{x}_2 \mid \mathbf{x}_1)$. This simplification can be formalised using delta functions as variational posteriors as in Friston (2005) (shown in appendix B).

Since the nodes on the lowest levels are fixed to the input $\mathbf{x}_1 = \mathbf{s}$, their values are not being changed but rather provide a condition on other variables. To simplify calculations, we define the per sample objective function equal to the logarithm of the joint distribution (since the logarithm is a monotonic operator, a logarithm of a function has the same maximum as the function itself):

$$F = \ln P(\mathbf{x}_L, ..., \mathbf{x}_2 \mid \mathbf{x}_1) \tag{4.4}$$

Since we assumed that the variables on one level just depend on variables of the level above, we can write the per sample objective function as:

$$F = \sum_{l=2}^{L} \ln P(\mathbf{x}_l \mid \mathbf{x}_{l-1}) \tag{4.5}$$

Substituting Equation 4.1 and the expression for a normal distribution into the above equation, we obtain:

$$F = -\frac{1}{2} \sum_{l=2}^{L} (\mathbf{x}_l - \boldsymbol{\mu}_l)^T \Sigma_l^{-1} (\mathbf{x}_l - \boldsymbol{\mu}_l) - \frac{1}{2} \ln(|\Sigma_l|) + Const \tag{4.6}$$

Recall that we wish to find the values $\mathbf{x}_l$ that maximize the above objective function. This can be achieved by modifying $\mathbf{x}_l$ proportionally to the gradient of the objective function. To calculate the derivative of F over $\mathbf{x}_l$ we note that each $\mathbf{x}_l$ influences F in two ways: it occurs in Equation 4.6 explicitly, but it also determines the values of $\boldsymbol{\mu}_{l+1}$. Thus the derivative contains two terms:

$$\frac{\partial F}{\partial \mathbf{x}_a} = -\boldsymbol{\varepsilon}_a + \left(\Theta_a^T \boldsymbol{\varepsilon}_{a+1}\right) \odot f'\left(\mathbf{x}_{a+1}\right) \tag{4.7}$$

Where

$$\boldsymbol{\varepsilon}_a = {\Sigma_a}^{-1}\left(\mathbf{x}_a - \boldsymbol{\mu}_a\right) \tag{4.8}$$

These terms describe by how much the value of a random variable on a given level differs from the mean predicted by the lower level, so let us refer to them as prediction errors. Substituting the definition of prediction errors into Equation 4.7 we obtain the following rule describing changes in $\mathbf{x}_a$ over time:

$$\dot{\mathbf{x}_a} = -\boldsymbol{\varepsilon}_a + \left(\Theta_a^T \boldsymbol{\varepsilon}_{a+1}\right) \odot f'\left(\mathbf{x}_{a+1}\right) \tag{4.9}$$

Neural implementation

The computations described by Equations 4.8-4.9 could be performed by a simple network illustrated in Figure 4.2 with nodes corresponding to prediction errors $\boldsymbol{\varepsilon}_l$ and values of random variables $\mathbf{x}_l$. The prediction errors $\boldsymbol{\varepsilon}_l$ are computed on the basis of excitation from corresponding variable nodes $\mathbf{x}_l$, and inhibition from the nodes on the lower level $\mathbf{x}_{l-1}$ weighted by strength of synaptic connections Θ_{l-1}. Conversely, the nodes $\mathbf{x}_l$ make computations on the basis of the prediction error from the corresponding level, and the prediction errors from the higher level weighted by synaptic weights.

To implement Equation 4.8, a prediction error node would get excitation from the corresponding variable node and inhibition equal to synaptic input from higher level nodes, thus it could compute the difference between them. Scaling the

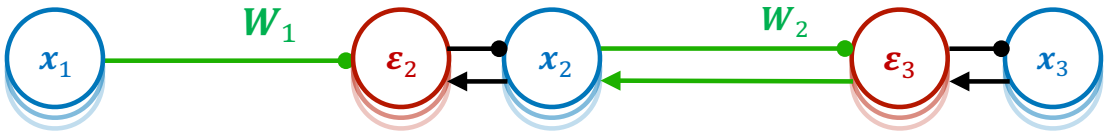

Figure 4.2: Architecture of the network. Arrows and lines ending with circles denote excitatory and inhibitory connections respectively. Connections without labels have weights fixed to 1.

activity of nodes encoding prediction error by a constant Σ_l could be implemented by self-inhibitory connections with weight Σ_l (we do not consider them here for simplicity - but for details see Friston (2005); Bogacz (2017)). Analogously to implement Equation 4.9, a variable node would need to change its activity proportionally to its inputs.

There are several other ways in which Equations 4.8-4.9 could be implemented in neural circuitry. Particular attention must be paid to how the non-linearity is represented too (see Whittington and Bogacz (2017) for an example implementation for predictive coding networks). In chapter 5 we discuss other possible implementations of this equation and show that several subsequent models also fall within this framework.

In the predictive coding model, after the input is provided, all nodes are updated according to Equations 4.8-4.9, until the network converges to a steady state. We label variables in the steady state with an asterisk e.g. $\mathbf{x}_l^*$ or F^*.

Learning parameters

During learning, the values of the nodes on the highest level are set to the output sample, i.e. $\mathbf{x}_L = \mathbf{t}$. Then the values of all nodes on levels $l \in \{2, ..., L-1\}$ are modified in the same way as described before (Equation 4.9).

Once the network has reached its steady state, the parameters of the model Θ_l are updated so the model better predicts the desired output. This is achieved by modifying Θ_l proportionally to the gradient of the objective function over the parameters. To compute the derivative of the objective function over Θ_l, we note that Θ_l affects the value of function F of Equation 4.6 by influencing $\boldsymbol{\mu}_{l+1}$, hence

$$\frac{\partial F^*}{\partial \Theta_a} = \boldsymbol{\varepsilon}_{a+1}^* f\left(\mathbf{x}_a^*\right)^T \tag{4.10}$$

and equivalently for the bias

$$\frac{\partial F^*}{\partial \mathbf{b}_a} = \boldsymbol{\varepsilon}_a \tag{4.11}$$

According to the above equation, the change in a synaptic weight Θ_a of connection between levels a and $a+1$ is proportional to the product of quantities encoded on these levels. For a linear function $f(\mathbf{x}) = \mathbf{x}$, the non-linearity in the above equation would disappear, and the weight change would simply be equal to the product of the activities of pre-synaptic and post-synaptic nodes (Figure 4.2). Even if the non-linearity is considered the weight change is fully determined by the activity of pre-synaptic and post-synaptic nodes. The learning rules of the top and bottom weights must be slightly different to maintain the symmetry of the connections, i.e. the bottom and the top connections are modified by the same amount. We refer to these changes as Hebbian in a sense that in both cases the weight change is a product of monotonically increasing functions of activity of pre-synaptic and post-synaptic neurons.

In Algorithm 1, we include pseudocode to clarify how the network operates in training mode.

Algorithm 1 Pseudocode for predictive coding during learning. Please note that in the simulations presented, to make for faster learning, first a prediction was made by inputting $\mathbf{s}$ alone and propagating through the network layer by layer, as we know that all error nodes eventually would converge to zero in the prediction phase (see next section). Then the output $\mathbf{t}$ is applied, after which inference took place.

for all Data **do**
  $\mathbf{x}_1 \leftarrow \mathbf{s}$
  $\mathbf{x}_L \leftarrow \mathbf{t}$
  repeat
    Inference: $\dot{\mathbf{x}_a} = -\boldsymbol{\varepsilon}_a + \left(\Theta_a^T \boldsymbol{\varepsilon}_{a+1}\right) \odot f'(\mathbf{x}_{a+1})$
  until convergence
  Update weights: $\Delta\Theta_a = \boldsymbol{\varepsilon}_{a+1}^* f(\mathbf{x}_a^*)^T$
  Update biases: $\Delta \mathbf{b}_a = \boldsymbol{\varepsilon}_a^*$

4.2 Results

4.2.1 Relationship of predictive coding networks to ANNs

An ANN has two modes of operation: during prediction it computes its output on the basis of $\mathbf{s}$, while during learning it updates its weights on the basis of

s and **t**. The predictive coding network can also operate in these modes. We now discuss the relationship between computations of an ANN and a predictive coding network in these two modes.

Prediction

We now show that the predictive coding network has a stable fixed point at the state where all nodes have the same values as the corresponding nodes in the ANN receiving the same input **s**. Since all nodes change proportionally to the gradient of F, the value of function F always increases. Since the network is constrained only by the input, the maximum value F can reach is 0, and because F is a negative sum of squares, and this maximum is achieved if all terms in the summation of Equation 4.6 are equal to 0, i.e. when:

$$\mathbf{x}_l^* = \boldsymbol{\mu}_l \tag{4.12}$$

Thus the nodes in the prediction mode have the same values at the fixed point as the corresponding nodes in the ANN, i.e. $\mathbf{x}_l^* = \Theta_{l-1} f\left(\mathbf{x}_{l-1}\right) + \mathbf{b}_l$.

Learning

Let us now analyse under what conditions weight changes in the predictive coding model converge to that in the back-propagation algorithm.

The weight update rules in the two models (Equations 3.4 and 4.11) have the same form, however, the prediction error terms $\boldsymbol{\delta}_l$ and $\boldsymbol{\varepsilon}_l$ were defined differently. To see the relationship between these terms, we will now derive the recursive formula for prediction errors $\boldsymbol{\varepsilon}_l$ analogous to that for $\boldsymbol{\delta}_l$ in Equation 3.5. We note that once the network reaches the steady state in the learning mode, the change in activity of each node must be equal to zero. Setting the left hand side of Equation 4.9 to 0 we obtain:

$$\boldsymbol{\varepsilon}_a = \left(\Theta_a^T \boldsymbol{\varepsilon}_{a+1}\right) \odot f'\left(\mathbf{x}_{a+1}\right) \tag{4.13}$$

We can now write a recursive formula for the prediction errors:

$$\boldsymbol{\varepsilon}_a = \begin{cases} \Sigma_L^{-1}\left(\mathbf{t} - \boldsymbol{\mu}_a^*\right) & \text{if } a = L \\ \left(\Theta_a^T\, \boldsymbol{\varepsilon}_{a+1}\right) \odot f'\left(\mathbf{x}_{(a+1)}\right) & \text{if } a < L \end{cases} \tag{4.14}$$

Let us first consider the case when all variance parameters are set to $\Sigma_l = \boldsymbol{I}$ (as in Rao and Ballard (1999)). Then the above formula has exactly the same form as for the back-propagation algorithm (Equation 3.5). Therefore, it may seem that weight change in the two models is identical. However, for the weight change to be identical, the values of the corresponding nodes must be equal, i.e. $\mathbf{x}_l^* = \mathbf{y}_l^*$. Although we have shown in the previous subsection that $\mathbf{x}_l^* = \mathbf{y}_l^*$ in the prediction mode, it may not be the case in the learning mode, because the nodes $\mathbf{x}_L$ are fixed (to $\mathbf{t}$), and thus function F may not reach the maximum of 0, so Equation 4.12 may not be satisfied.

Let us now consider under what conditions $\mathbf{x}_l^*$ is equal or close to $\mathbf{y}_l$. First, when the networks are trained such that they correctly predict the output training samples, then the objective function F can reach 0 during the relaxation and hence $\mathbf{x}_l^* = \mathbf{y}_l$, and the two models have exactly the same weight changes. In particular, the change in weights is then equal to 0, thus the weights resulting in perfect prediction are a fixed point for both models.

Second, when the networks are trained such that their predictions are close to the output training samples, then fixing $\mathbf{x}_1$ will only slightly change the activity of other nodes in the predictive coding model, so the weight change will be similar.

To illustrate this property we compare the weight changes in predictive coding models and ANN with very simple architecture shown in Figure 4.3A. This network consists of just three layers ($L = 3$) and one node in each layer. Such network has only 2 weight parameters (w_1 and w_2), so the objective function of the ANN can be easily visualized. The network was trained on a set in which input training samples were generated randomly from uniform distribution $\mathbf{s} \in [-5, 5]$, and output training samples were generated as $\mathbf{t} = W_1 \tanh(W_2 \tanh(\mathbf{s}))$, where $W_1 = W_2 = 1$ (Figure 4.3B). Figure 4.3C shows the objective function of the ANN for this training set. Thus an ANN with weights equal to $w_l = W_l$ perfectly predicts all samples in the training set, so the objective function is equal to 0. There are

also other combinations of weights resulting in good prediction, which create a 'ridge' of the objective function.

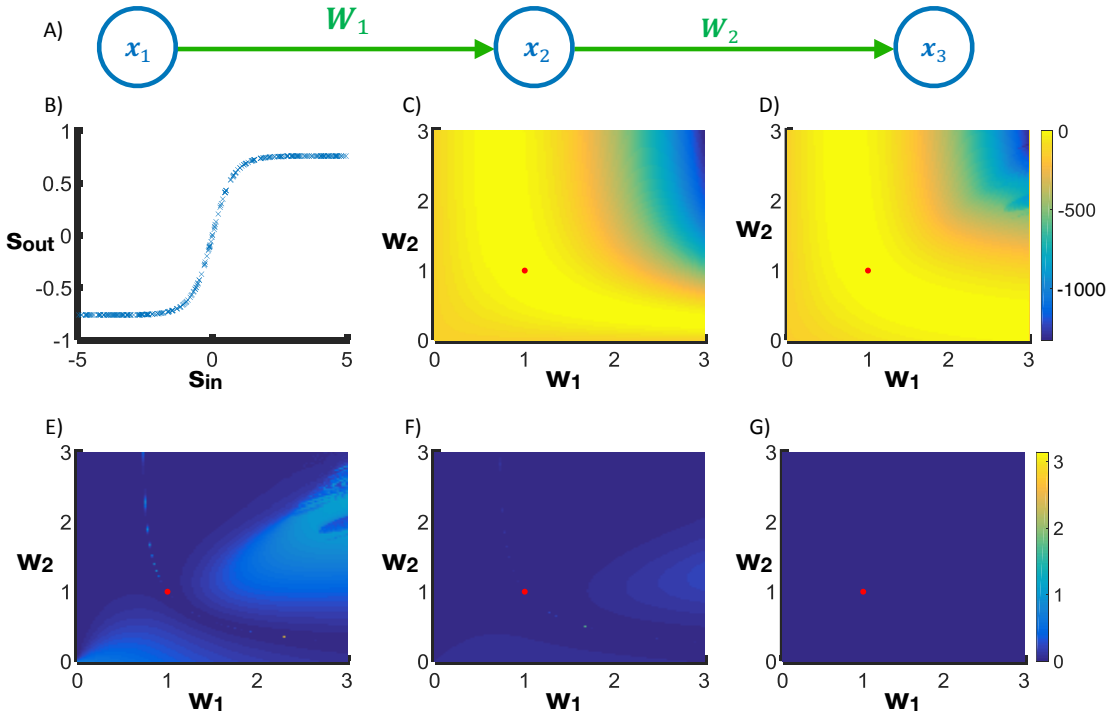

Figure 4.3: A) The structure of the network used. B) The data that the models were trained on, here $\mathbf{t} = tanh(tanh(\mathbf{s}))$ C) The objective function of an ANN for a training set with 300 samples generated as described in main text. The objective function is equal to sum of 300 terms given by Equation 3.3 corresponding to individual training samples. The red dot indicates weights that maximize the objective function. D) The objective function of the predictive coding model at the fixed point. For each set of weights and training sample, to find the state of predictive coding network at the fixed point, the nodes in layers 0 and 2 were set to training examples, and the node in layer 1 was updated according to Equation 4.9. This equation was solved using Euler method. A dynamic form of the Euler integration step was used where its size was allowed to reduce by a factor should the system not be converging (i.e. the maximum change in node activity increases from the previous step). Initial step size was 0.2. The relaxation was performed until the maximum value of $\partial F/\partial \mathbf{x}_l$ was lower than $10^{-6}/\Sigma_3$ or 1,000,000 iterations had been performed. E-G) Angle difference between the gradient for the ANN and the gradient for the predictive coding model found from Equation 4.11. Different panels correspond to different values of parameter describing sensory noise: E) $\Sigma_3 = 1$. F) $\Sigma_3 = 8$. G) $\Sigma_3 = 256$.

Figure 4.3E shows the angle between the direction of weight change in back-propagation and the predictive coding model. The directions of the gradient for the two models are very similar except for the regions where the objective functions E and F^* are misaligned (cf. Figures 4.3 C and D). Nevertheless, close to the

maximum of the objective function (indicated by a red dot), the directions of weight change become similar and the angle decreases towards 0.

There is also a third condition under which the predictive coding network approximates the back-propagation algorithm. Namely, when the value of parameters Σ_L is increased relative to other Σ_l, then the impact of fixing $\mathbf{x}_L$ on the activity of other nodes is reduced, because $\boldsymbol{\varepsilon}_L$ becomes smaller (Equation 4.8) and its influence on activity of other nodes is reduced. Thus $\mathbf{x}_l^*$ is closer to $\mathbf{y}_l$ (for $l < L$), and the weight change in the predictive coding model becomes closer to that in the back-propagation algorithm (recall that the weight changes are the same when $\mathbf{x}_l^* = \mathbf{y}_l$ for $l < L$).

Multiplying Σ_L by a constant will also reduce all $\boldsymbol{\varepsilon}_l$ by the same constant (see Equation 4.14), and consequently all weight changes will be reduced by this constant. This can be compensated by multiplying the learning rate α by the same constant, so the magnitude of the weight change remains constant. In this case, the weight updates of the predictive coding network will become asymptotically similar to the ANN, regardless of prediction accuracy.

Figures 4.3F and G show that as Σ_L increases the angle between weight changes in the two models decreases towards 0. Thus as the parameters Σ_L are increased, the weight changes in the predictive coding model converge to those in the back-propagation algorithm.

We however note that learning driven by very small values of the error nodes is less biologically plausible. However in Figure 4.4 we will show that a high value of Σ_L is not required for good learning with these networks.

4.2.2 Performance on more complex learning tasks

To asses the performance of the predictive coding model on more complex learning tasks, we tested it on the MNIST dataset. This is a dataset of 28 by 28 images of handwritten digits, each associated with one of the 10 corresponding classes of digits. We performed the analysis for an ANN of size 784-600-600-10 ($L = 4$), with predictive coding networks of the corresponding size too. We use the logistic sigmoid as the activation function. We ran the simulations for both the $\Sigma_L = 1$

case and the $\Sigma_L = 100$ case. Figure 4.4 shows the learning curves for these different models. Each curve is the mean from ten simulations, with standard error shown as the shaded regions.

We see that the predictive coding models perform similarly to the ANN. For a large value of parameter Σ_L the performance of the predictive coding model was very similar to the back-propagation algorithm, in agreement with earlier analysis showing that then the weight changes in the predictive coding model converge to those in the back-propagation algorithm. Should we have had more than 20 steps in each inference stage, i.e. allowed the network to converge in inference, then the ANN and the predictive coding network with $\Sigma_L = 100$ would have had an even more similar trajectory.

We see that all the networks eventually obtain a training error of 0.00%, and a validation error of $\sim 1.7 - 1.8\%$. We did not optimise the learning rate for validation error as we are solely highlighting the similarity between ANNs and predictive coding.

4.2.3 Effects of architecture of the predictive coding model

The probabilistic model we considered so far corresponds to brain regions in the opposite way to the original version of predictive coding (Rao and Ballard, 1999; Friston, 2005). That model, developed for unsupervised learning, consists of a hierarchical Bayesian network predicting sensory input $\mathbf{s}$. The level of hierarchy mimic those in the brain i.e. the 'final' variables in the generative model correspond to early sensory areas and higher up generative variables correspond to brain regions further up the cortical hierarchy. This is opposite to what we have just presented, where our Bayesian network predicts a class output $\mathbf{s}$, conditioned on a sensory input $\mathbf{s}$. Following the same logic would lead up to the class output being in early sensory areas and the sensory input further up the cortical hierarchy. This doesn't make too much sense. We can provide alternative explanations, such as the sensory inputs $\mathbf{s}$ are already preprocessed by sensory areas etc, but we doth explain too much, methinks.

To perhaps provide a better alternative, the sensory areas can be added to the model by considering an architecture in which there are two separate lower

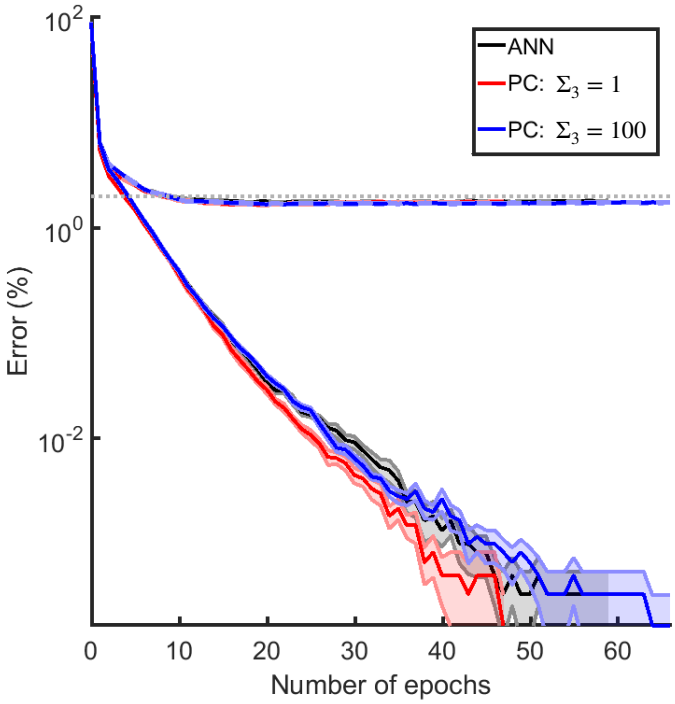

Figure 4.4: Comparison of prediction accuracy (%) for different models (indicated by colours - see key) on the MNIST dataset. Training errors are shown with solid lines, and validation errors with dashed lines. The dotted grey line denotes 2% error. The models were run 10 times each, initialised with different weights. When the training error lines stop, this is when the mean error of the 10 runs was equal to zero. The weights were drawn from a uniform distribution with maximum and minimum values of $\pm 4\sqrt{\frac{6}{N}}$ where N is the total number of neurons in the two layers either side of the weight. The input data was first transformed through an inverse logistic function as pre-processing, before being given to the network. When the network was trained with an image of class c, the c^{th} element of x_L was set to 0.97 and the others to 0.03. After inference and before the weight update, the error node values were scaled by Σ_3 so as to be able to compare between the models. We used a batch size of 20, with a learning rate of 0.001 and the stochastic optimiser Adam (Kingma and Ba, 2014) to accelerate learning - this is essentially a per-parameter learning rate, where weights that are infrequently updated are updated more and vice-versa. We chose the number of steps in the inference phase to be 20, at this point the network will not necessarily have converged, but we did so to aid speed of training. This is not the minimum number of inference iterations that allows for good learning, this notion will be explored in a future paper. Otherwise simulations were as per Figure 4.3. The shaded regions in the fainter colour describe the standard error of the mean. The figure is shown on a logarithmic plot.

level areas receiving **s** and **t**, which are both connected with higher areas (de Sa and Ballard, 1998; Hyvarinen, 1999; O'Reilly and Munakata, 2000; Bengio, 2014; Srivastava and Salakhutdinov, 2014; Hinton et al., 2006). For example, in case of learning associations between visual stimuli (e.g. shapes of letters) and auditory

stimuli (e.g. their sounds), $\mathbf{s}$ and $\mathbf{t}$ could be provided to primary visual and primary auditory cortices, respectively. Both of these primary areas project through a hierarchy of sensory areas to a common higher associative cortex.

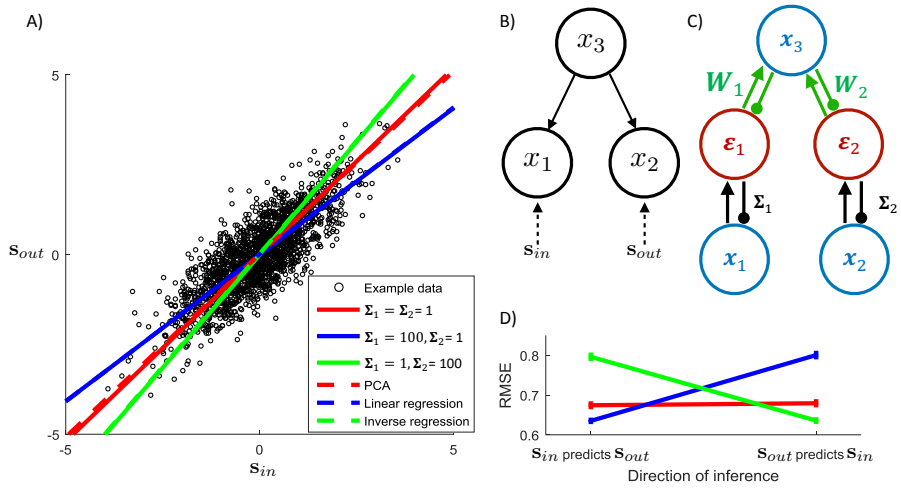

Figure 4.5: The effect of variance associated with different inputs on network predictions. A) Sample training set composed or 2000 randomly generated samples, such that $\mathbf{s} = a + b$ and $\mathbf{t} = a - b$ where $a \sim \mathcal{N}(0, 1)$ and $b \sim \mathcal{N}(0, 1/9)$. Lines compare the predictions made by the model with different parameters with predictions of standard algorithms (see key). B) Structure of probabilistic model and C) Architecture of the simulated predictive coding network. Notation as in Figure 4.2. Additionally, connections shown in grey are used if the network predicts the value of the corresponding sample. D) Root Mean Squared Error (RMSE) of the models with different parameters (see key of panel A) trained on data as in panel A and tested on further 100 samples generated from the same distribution. During the training, for each sample the network was allowed to converge to the fixed point as described in the caption of Figure 4.3 and the weights were modified with learning rate $\alpha = 1$. The entire training and testing procedure was repeated 50 times, and the error bars show the standard error.

To understand the potential benefit of such an architecture over the standard back-propagation, we analyse a simple example of learning the association between one dimensional samples shown in Figure 4.5A. Since there is a simple linear relationship (with noise) between samples in Figure 4.5A, we will consider predictions generated by a very simple network derived from a probabilistic model shown in Figure 4.5B. During training of this network the samples are provided to the nodes on the lowest level ($\mathbf{x}_2 = \mathbf{t}$ and $\mathbf{x}_1 = \mathbf{s}$).

We consider the following probabilistic model

$$P\left(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\right) = P\left(\mathbf{x}_1 \mid \mathbf{x}_3\right) P\left(\mathbf{x}_2 \mid \mathbf{x}_3\right) P\left(\mathbf{x}_3\right) \tag{4.15}$$

and for simplicity, we will assume a linear dependence of variables on the higher level:

$$P\left(\mathbf{x}_i \mid \mathbf{x}_3\right) = \mathcal{N}\left(\mathbf{x}_i; \Theta_i \, \mathbf{x}_3, \Sigma_i\right) \tag{4.16}$$

For simplicity let us assume an uninformative flat prior $P(\mathbf{x}_3) = c$, where c is a constant. Since the node on the highest level is no longer constrained, the objective function we wish to maximize is the logarithm of the joint probability of all nodes:

$$F = \ln P(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \tag{4.17}$$

Ignoring constant terms this function has analogous form as in Equation 4.6:

$$F = \sum_{i=1}^{2} -\frac{1}{2} \left(\mathbf{x}_i - \Theta_i \, \mathbf{x}_3\right)^T \Sigma_i^{-1} \left(\mathbf{x}_i - \Theta_i \, \mathbf{x}_3\right) \tag{4.18}$$

During training, the nodes on the lowest level are fixed, and node on the top level is updated proportionally to the derivative of F, analogously as in the models discussed previously:

$$\dot{\mathbf{x}_3} = \sum_{i=1}^{2} \Theta_i^T \, \boldsymbol{\varepsilon}_i \tag{4.19}$$

As before such computation can be implemented in a simple network shown in Figure 4.5C. After the nodes converge, the weights are modified to maximize F, which here is simply $\Delta\Theta_i \sim \boldsymbol{\varepsilon}_i \, \mathbf{x}_3^T$.

During testing, we only constrain $\mathbf{x}_1$ to be $\mathbf{s}$, and let all other nodes be updated to maximize F, i.e. the node on the top level evolves according to Equation 4.19, while the $\mathbf{x}_2$ evolves according to $\boldsymbol{\varepsilon}_2$.

Please note that this simple linear dependence could be captured by using a predictive coding network without a hidden layer and just by learning the means and covariance matrix i.e. $P\left(\mathbf{x}\right) = \mathcal{N}\left(\mathbf{x}; \boldsymbol{\mu}, \Sigma\right)$, where $\boldsymbol{\mu}$ is the mean and Σ the covariance

matrix. However, we use a hidden layer to show the more general network, that could learn more complicated relationships if non-linear activation functions are used.

Solid lines in Figure 4.5A show the values predicted by the model (i.e. $\mathbf{x}_2^*$) after providing different inputs. Different colours correspond to different noise parameters. When equal noise is assumed in input and output (red line), the network simply learns the probabilistic model that explains the most variance in the data, so the model learns the direction in which the data is most spread out. This direction is the same as the first principal component shown in dashed red line (any difference between the two lines is due to the iterative nature of learning in the predictive coding model).

When the noise parameter at the node receiving output samples is large (blue line in Figure 4.5A), the dynamics of the network will lead to the node at the top level converging to a linear transformation of the input sample. Given the analysis presented earlier, the model closely resembles the back-propagation algorithm, which in the case of linear $f(\mathbf{x})$ simply corresponds to linear regression, shown by dashed blue line.

Conversely, when the noise at the node receiving input samples is large (green line in Figure 4.5A), the dynamics of the network will lead to the node at the top level converging to a linear transformation of the output sample. The network in this case will learn to predict the input sample on the basis of the output sample. Hence its predictions correspond to that obtained by finding linear regression in inverse direction (i.e. the linear regression predicting $\mathbf{s}$ on the basis of $\mathbf{t}$), shown by the dashed green line.

Different predictions of the models with different noise parameters will lead to different amounts of error when tested, which are shown in the left part of Figure 4.5D (labelled $\mathbf{s}$ predicts $\mathbf{t}$). The network approximating the back-propagation algorithm is most accurate, as the back-propagation algorithm explicitly minimizes the error in predicting output samples. Next in accuracy is the network with equal noise on both input and output, followed by the model approximating inverse regression.

Due to the flexible structure of the predictive coding network, we can also test how well it is able to infer the likely value of input sample $\mathbf{s}$ on the basis of the output sample $\mathbf{t}$. In order to test it, we provide the trained network with the output sample ($\mathbf{x}_2 = \mathbf{t}$), and let the other nodes be updated. The value to which the node corresponding to the input ($\mathbf{x}_1^*$) converged is the network's inferred value of the input. We compared these values with actual $\mathbf{s}$ in the testing examples, and the resulting root mean squared errors are shown in the right part of Figure 4.5D (labelled $\mathbf{t}$ predicts $\mathbf{s}$). This time the model approximating the inverse regression is most accurate.

Figure 4.5D illustrates that when noise is present in the data, there is a trade-off between accuracy of inference in the two directions. Nevertheless, the predictive coding network with equal noise parameters for inputs and outputs is predicting relatively well in both directions, being just slightly less accurate than the optimal algorithm for the given direction.

It is also important to emphasize that the models we analysed in this section generate different predictions, only because the training samples are noisy. If the amount of noise were reduced, the models' predictions would become more and more similar (and their accuracy would increase). This parallels the property discussed earlier that the closer the predictive coding models predict all samples in the training set, the closer their computation to ANNs with back-propagation algorithm.

The networks in the cortex are likely to be non-linear and include multiple layers, but predictive coding models with corresponding architectures are still likely to retain the key properties outlined above. Namely, they would allow learning bidirectional associations between inputs and outputs, and if the mapping between the inputs and outputs could be perfectly represented by the model, the networks could be able to learn them and make accurate predictions.

4.3 Discussion

In this paper we have proposed how the predictive coding models can be used for supervised learning. We showed that they perform the same computation as

ANNs in the prediction mode, and weight modification in the learning mode has a similar form as for the back-propagation algorithm. Furthermore, in the limit of parameters describing the noise in the layer where output training samples are provided, the learning rule in the predictive coding model converges to that for the back-propagation algorithm.

Further analysis of the biological plausibility of the predictive coding model, along with comparison to other models, and relationship to experimental data, will be discussed in chapter 5.

5

Integrating biological models of back-propagation

Contents

This chapter summarises recently proposed theories on how neural circuits in the brain could approximate the error back-propagation algorithm used by artificial neural networks. Computational models implementing these theories achieve learning as efficient as artificial neural networks, but they use simple synaptic plasticity rules based on activity of presynaptic and postsynaptic neurons. The models have similarities, such as including both feedforward and feedback connections, allowing information about error to propagate throughout the network. Furthermore, they incorporate experimental evidence on neural connectivity, responses, and plasticity. These models provide insights on how brain networks might be organised such

that modification of synaptic weights on multiple levels of cortical hierarchy leads to improved performance on tasks.

5.1 Classification of models of biological back-propagation

There is a diversity of ideas on how the back-propagation algorithm may be approximated in the brain (Balduzzi et al., 2015; Krotov and Hopfield, 2019; Kuśmierz et al., 2017; Marblestone et al., 2016; Bengio, 2014; Lee et al., 2015); however, we review the principles behind a set of related models (Bengio, 2017; Sacramento et al., 2018; Whittington and Bogacz, 2017; O'Reilly, 1996) that have substantial connections with biological data while closely paralleling the back-propagation algorithm. These models operate with minimal external control, as they can compute the errors associated with individual neurons through the dynamics of the networks. Thus, synaptic weight modifications depend only on the activity of presynaptic and postsynaptic neurons. Furthermore, these models incorporate important features of brain biology, such as spike time-dependent plasticity, patterns of neural activity during learning, and properties of pyramidal neurons and cortical microcircuits. We emphasise that these models rely on fundamentally similar principles. In particular, the models include both feedforward and feedback connections, thereby allowing information about the errors made by the network to propagate throughout the network without requiring an external program to compute the errors. Furthermore, these dynamics, as well as synaptic plasticity, can be described within a common framework of energy minimisation. We divide the reviewed models in two classes differing in how the errors are represented, and we summarise them in the following sections.

5.1.1 Temporal-Error Models

This class of model encodes errors in the differences in neural activity across time. Temporal-error models describe learning in networks with recurrent feedback connections to the hidden nodes (Figure 5.1). The rate of change of activity of

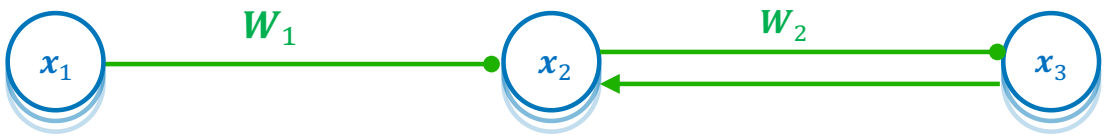

Figure 5.1: Temporal-error network architecture.

a given node is proportional to the summed inputs from adjacent layers, along with a decay term proportional to the current level of activity. As the network is now recurrent, it is no longer possible to write a simple equation describing how the activity depends on other nodes; instead, the dynamics of neurons is described by the differential equation

$$\dot{\mathbf{x}}_l = W_{l-1}\,\mathbf{x}_{l-1} + W_{l-1}^T\,\mathbf{x}_{l+1} - \mathbf{x}_l \tag{5.1}$$

where $\dot{\mathbf{x}}_l$ denotes the rate of change over time of $\mathbf{x}_l$ (all equations in this chapter ignore nonlinearities for brevity).

The first model in this class is the contrastive learning model (O'Reilly, 1996). It relies on an observation that weight changes proportional to an error (difference between predicted and target pattern - from chapter 3) can be decomposed into two separate updates: one update based on activity without the target present and the other update with the target pattern provided to the output neurons (Ackley et al., 1985). To understand learning in this model, it is easiest to consider how the weights connecting to the output layer are modified. Here the weight modification required by the back-propagation algorithm can be decomposed into two terms.

$$\Delta W_2 \sim (\mathbf{t} - \mathbf{x})\,\mathbf{x}^T = \mathbf{t}\,\mathbf{x}^T - \mathbf{x}\,\mathbf{x}^T \tag{5.2}$$

The first term corresponds to anti-Hebbian plasticity that should take place when the output activity is predicted based on the input propagated through the network. The second term corresponds to Hebbian plasticity that should take place when the output layer is set to the target pattern. O'Reilly (O'Reilly, 1996) demonstrated that in the presence of backward connections, the information about the target pattern propagates to earlier layers, and an analogous sequence

of weight modifications in the hidden layers also approximates a version of the back-propagation algorithm for recurrent networks (Pineda, 1987).

Thus, the error back-propagation algorithm can be approximated in a network in which the weights are modified twice: during prediction according to anti-Hebbian plasticity and then according to Hebbian plasticity once the target is provided and the network converges to an equilibrium (after the target activity has propagated to earlier layers via feedback connections) (O'Reilly, 1996). The role of the first modification is to 'unlearn' the existing association between input and prediction, while the role of the second modification is to learn the new association between input and target.

Although the weight modifications in the contrastive learning model involve locally available information, implementing them biologically would require a global signal informing the network which phase it is in (whether the target pattern influences the network or not) as that determines whether the plasticity should be Hebbian or anti-Hebbian. It is not clear whether such a control signal exists in the brain. This concern can be alleviated if the determination of learning phases is coordinated by information locally available in the oscillatory rhythms (Baldi and Pineda, 1991), such as hippocampal theta oscillations (Ketz et al., 2013). In these models, the neurons in the output layer are driven by feedforward inputs in one part of the cycle and forced to take the value of the target pattern in the other.

The complications of separate phases have been recently addressed in the continuous update model (Bengio, 2017), where during training the output neuron activities are gradually changed from the predicted pattern ($\mathbf{x}_3 \mid_{\rightharpoonup \mathbf{t}}$) towards the target, $\mathbf{t}$ (see Figure 5.1. Thus, the temporal derivative of output activity, $\dot{\mathbf{x}_3}$, is proportional to ($\mathbf{t} - \mathbf{x}_3 \mid_{\rightharpoonup \mathbf{t}}$), that is, to the error on the output (equivalent to ANN). Hence, the weight modification required by back-propagation is simply equal to the product of presynaptic activity and the rate of change of the postsynaptic activity

$$\Delta W_2 \sim \boldsymbol{\delta}_3 \, \mathbf{x}_2^T \sim \dot{\mathbf{x}_3} \, \mathbf{x}_2^T \tag{5.3}$$

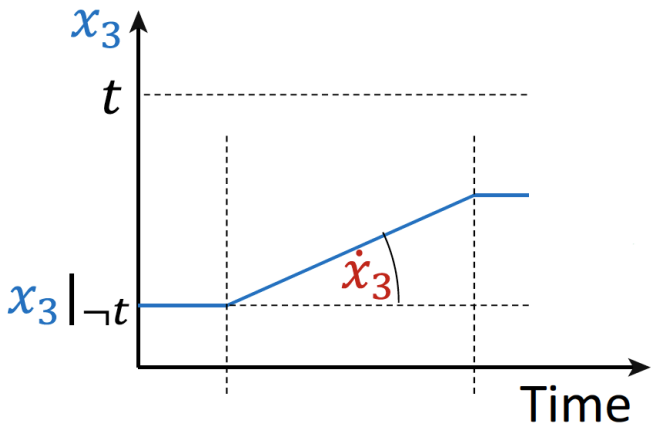

Figure 5.2: Schematic showing that if the output neurons are gradually changed form prediction, $\mathbf{x}_3\,|_{\neg t}$, to target, t, then the rate of change of output activity is the required error.

Consequently, the weight modification required by the back-propagation algorithm could arise from local plasticity based on the rate of change of activity. Although the continuous update model does not involve two different learning rules during prediction and learning, it still requires a control signal indicating whether the target pattern is present or not, because plasticity should not take place during prediction.

5.1.2 Explicit-Error Models

In this section, we describe alternative models that do not require control signals but as a trade-off have more complex architectures that explicitly compute and represent errors.

It has been recently noticed (chapter 3; Whittington and Bogacz (2017) that the error back-propagation algorithm can be approximated in a widely used model of information processing in hierarchical cortical circuits called predictive coding (Rao and Ballard, 1999). In its original formulation, the predictive coding model was developed for unsupervised learning, and it has been shown that when the model is presented with natural images, it learns representations similar to those in visual cortex (Rao and Ballard, 1999). Predictive coding models have also been proposed as a general framework for describing different types of information processing in the brain (Friston, 2010). It has been recently shown that when a

predictive coding network is used for supervised learning, it closely approximates the error back-propagation algorithm (Whittington and Bogacz, 2017). We saw this in chapter 4 but remind ourselves of the basics now.

An architecture of a predictive coding network contains error nodes that are each associated with corresponding value nodes (Figure 5.3). During prediction, when the network is presented with an input pattern, activity is propagated between the value nodes via the error nodes. The governing dynamics are given by (in the case of linear activation functions)

$$\dot{\mathbf{x}}_l = -\boldsymbol{\varepsilon}_l + W_l^T \boldsymbol{\varepsilon}_{l+1} \qquad\qquad \boldsymbol{\varepsilon}_l = \mathbf{x}_l - W_{l-1}\,\mathbf{x}_{l-1} \tag{5.4}$$

The network converges to an equilibrium, in which the error nodes decay to zero and all value nodes converge to the same values as the corresponding artificial neural network i.e. when $\boldsymbol{\varepsilon}_l = 0$, $\mathbf{x}_l = W_{l-1}\,\mathbf{x}_{l-1} + \mathbf{b}_l$.

During learning, both the input and the output layers are set to the training patterns. The error nodes can no longer decrease their activity to zero, instead they converge to values as if the errors had been back-propagated (Whittington and Bogacz, 2017). Once the state of the predictive coding network converges to equilibrium, the weights are modified, according to a Hebbian plasticity rule. These weight changes closely approximate that of the back-propagation algorithm.

$$\Delta W_l = \boldsymbol{\varepsilon}_{l+1}\,\mathbf{x}_l^T \tag{5.5}$$

$$\boldsymbol{\varepsilon}_l = W_l^T\,\boldsymbol{\varepsilon}_{l+1} \tag{5.6}$$

An important property of the predictive coding networks is that they work autonomously: irrespective of the target pattern being provided, the same rules for node dynamics and plasticity are used. If the output nodes are unconstrained, the error nodes converge to zero, so the Hebbian weight change is equal to zero. Thus, the networks operate without any need for external control except for providing different inputs and outputs. However, the one-to-one connectivity of error nodes

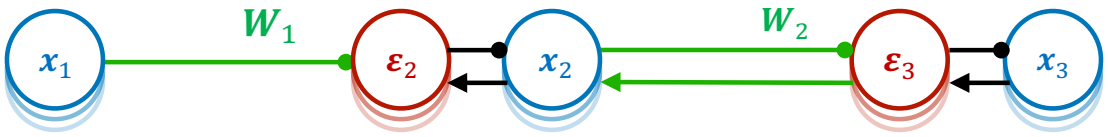

Figure 5.3: Predictive coding. (A) Network architecture. Blue and red circles denote the value and error nodes, respectively. Arrows and lines ending with circles denote excitatory and inhibitory connections, respectively; green double lines indicate connections between all neurons in one layer and all neurons in the next layer, while single black lines indicate within layer connections between a corresponding error and value node.

to their corresponding value nodes is inconsistent with diffused patterns of neuronal connectivity in the cortex.

A solution to this inconsistency has been proposed in several models in which the error is represented in dendrites of the corresponding neuron (Richards and Lillicrap, 2019; Kording and Konig, 2001; Körding and König, 2001). In this chapter, we focus on a popular model called the dendritic error model (Sacramento et al., 2018). This model describes networks of pyramidal neurons and assumes that the errors in the activity of pyramidal neurons are computed in their apical dendrites. In this model, the apical dendrites compare the feedback from the higher levels with a locally generated prediction of higher-level activity computed via interneurons. Error information can be transmitted from the apical dendrite to the rest of the neuron through internal signals. For example, a recent computational model proposed that errors encoded in apical dendrites can determine the plasticity in the whole neuron (Guergiuev et al., 2017). The model is based on observations that activating apical dendrites induces plateau potentials via calcium influx, leading to a burst of spikes by the neuron (Larkum et al., 1999). Such bursts of spikes may subsequently trigger synaptic plasticity (Pike et al., 1999; Roelfsema and Holtmaat, 2018).

An easy way to understand why such an architecture approximates the back-propagation algorithm is to notice that it is closely related to predictive coding networks, which approximate artificial neural networks. Simply rearranging the equations describing the dynamics of predictive coding model gives a description of a network with the same architecture as the dendritic error model, in which dendrites encode the error terms.

$$\dot{\mathbf{x}}_l = -\mathbf{x}_l - W_{l-1}\,\mathbf{x}_{l-1} + W_l^T\,\mathbf{x}_{l+1} - W_l^T W_l\,\mathbf{x}_l \tag{5.7}$$

This describes the dynamics of pyramidal neurons in Figure 5.4. The right side of Equation 5.7 consists of four terms corresponding to various connections in Figure 5.4. The first is simply a decay, the second is a feedforward input from the previous layer, the third is a feedback from the layer above, and the fourth term is a within layer recurrent input. This last term has a negative sign, while pyramidal neurons are excitatory, so it needs to be provided by interneurons. If we assume that the interneurons have activity $\mathbf{i}_l = W_l\,\mathbf{x}_l$, they need to be connected with the pyramidal neurons via weights W_l. The key property of this network is that when it converges to the equilibrium, the neurons with activity $\mathbf{x}_l$ encode their corresponding error terms $\boldsymbol{\varepsilon}_l$ in their apical dendrites. To see why this is the case, note that the first two terms on the right of Equation 5.7 are equal to $-\boldsymbol{\varepsilon}_l$ according to the definition of Equation 5.6. At equilibrium $\dot{\mathbf{x}}_l = 0$, the two last terms in Equation 5.7 must be equal to $\boldsymbol{\varepsilon}_l$ (so that the right-hand side of Equation 5.7 adds up to 0), and it is these two terms that define the input to the apical dendrite.

As the errors $\boldsymbol{\varepsilon}_l$ are encoded in apical dendrites, the weight modification required by the back-propagation algorithm (equation 3.4 for ANNs and equation 5.5 for PCNs) only involves quantities encoded in pre-and postsynaptic neurons and therefore corresponds to local synaptic plasticity. Appropriately updating weights between pyramidal and interneurons is more challenging. This is because the interneurons must learn to produce activity encoding the same information as the higher-level pyramidal neurons. To allow training of the interneurons, the dendritic error model includes special one-to-one connections to the interneurons from corresponding higher-level pyramidal neurons (black dashed arrows in Figure 5.4).

Although the dendritic error network makes significant steps to increase the biological realism of predictive coding models, it also introduces extra one-to-one connections (dotted arrow in Figure 5.4) that enforce the interneurons to take on similar values to the neurons in next layer and thus help them to predict the feedback from the next level. Furthermore, the exact dynamics in the dendritic error model are

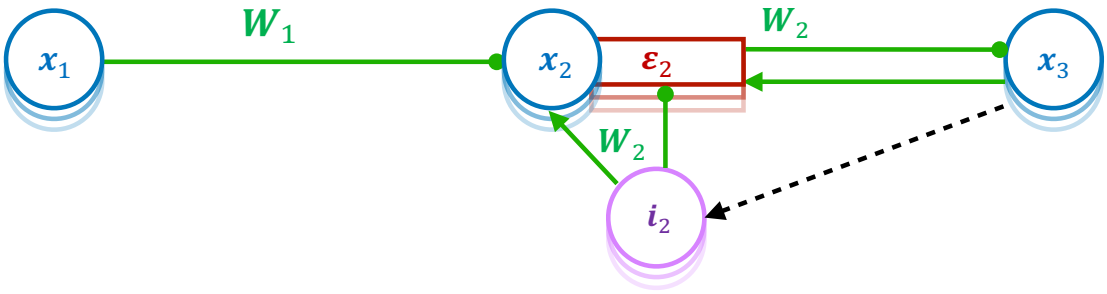

Figure 5.4: Network architecture of the dendritic error model (Sacramento et al., 2018). Blue circles indicate pyramidal neurons, red rectangles indicate their apical dendrites, and purple circles denote interneurons. In this network, the activity is propagated through the layers via connections between pyramidal neurons. The errors in the activity of pyramidal neurons are computed in their apical dendrites.

much more complex than described, as it describes details of changes in membrane potential in multiple compartments. Nevertheless, it is important to highlight that the architecture of dendritic error networks can approximate the back-propagation algorithm, and it offers an alternative hypothesis on how the computations assumed by the predictive coding model could be implemented in cortical circuits.

5.2 Comparing the Models

Given the biological plausibility of the above-mentioned models, in this and the coming sections, we compare the models in terms of their computational properties (as more efficient networks may be favoured by evolution) and their relationships to experimental data (summarised in Figure 5.5).

5.3 Models of Biological Back-Propagation

5.3.1 Computational Properties

For correct weight modification, the temporal-error models require a mechanism informing whether the target pattern constrains the output neurons, while the explicit-error models do not. However, as a trade-off, the temporal-error models have simpler architectures, while the explicit-error models need to have intricate architectures with certain constraints on connectivity, and both predictive coding and

the dendritic error model include one-to-one connections in their network structure. As mentioned, there is no evidence for such one-to-one connectivity in the neocortex.

The models differ in the time required for signals to propagate through the layers. To make a prediction in networks with L layers, predictive coding networks need to propagate information through $2L - 1$ synapses, whereas the other models only need to propagate through $L - 1$ synapses. This is because in a predictive coding network, to propagate from one layer to the next, the information must travel via an error neuron, whereas in the other models the information is propagated directly to the neurons in the layer above. There is a clear evolutionary benefit to propagating information via fewer synapses, as it would result in faster responses and a smaller number of noise sources.

In the dendritic error model, for errors to be computed in the dendrites, the inhibitory interneurons first need to learn to predict the feedback from the higher level. Thus, before the network can learn feedforward connections, ideally the inhibitory neurons need to first be pre-trained. Although it has been shown that the feedforward and inhibitory weights can be learned in parallel, learning in the dendritic error model may well be slower as the reported number of iterations required to learn a benchmark task was higher for the dendritic error model (Sacramento et al., 2018) than for contrastive learning (Scellier and Bengio, 2017a) and predictive coding (Whittington and Bogacz, 2017) models. Such statements, however, should be taken with reservations as not only were simulations not necessarily comparable but also computations in standard von-Neumann computers may not be representative of computations in biological hardware.

5.3.2 Relationship to Experimental Data

The models differ in their predictions on whether errors should be explicitly represented in neural activity. In particular, the predictive coding model includes dedicated neurons encoding errors, and the dendritic error model suggests that errors computed in dendrites may trigger bursts of firing of pyramidal neurons, while in temporal models there is no direct association between error and the overall

		Temporal-error model		Explicit-error model	
		Contrastive learning	Continuous update	Predictive coding	Dendritic error
Properties[a]	Control signal	Required	Required	Not required	Not required
	Connectivity	Unconstrained	Unconstrained	Constrained	Constrained
	Propagation time	L-1	L-1	2L-1	L-1
	Pre-training	Not required	Not required	Not required	Required
Error encoded in		Difference in activity between separate phases	Rate of change of activity	Activity of specialised neurons	Apical dendrites of pyramidal neurons
Data accounted for		Neural responses and behaviour in a variety of tasks	Typical spike-time-dependent plasticity	Increased neural activity to unpredicted stimuli	Properties of pyramidal neurons
MNIST performance[b]		~2–3	–	~1.7	~1.96

Figure 5.5: [a]Green indicates properties desired for biological plausibility, while red indicates less desired properties. [b]These are error percentages reported on a testing set in a benchmark task of handwritten digit classification (lower is better), for predictive coding (Whittington and Bogacz, 2017), dendritic error (Sacramento et al., 2018), and contrastive learning models (Scellier and Bengio, 2017a) (in this simulation, the output neurons were not set to the target pattern, but slightly moved or 'nudged' towards it). We are not aware of reported simulations of the continuous update model on this benchmark problem. MNIST, Modified National Institute of Standards and Technology database.

activity level at a given time. In line with the explicit-error models, increased neural activity has been observed when sensory input does not match the expectations encoded by higher-level areas. For example, responses of neurons in the primary visual cortex were increased at brief intervals in which visual input did not match expectation based on animal movements (Attinger et al., 2017). An increase in neural activity when expectations about stimuli were violated has also been found with fMRI (Summerfield et al., 2008). Further details are discussed in several excellent reviews (Summerfield and de Lange, 2014; Bastos et al., 2012; de Lange et al., 2018; Clark, 2013). The two explicit models differ in predictions on whether errors and values are represented by separate neuronal populations or within the same neurons. Experimental data relevant to this question have been reviewed in an excellent chapter by Kok and de Lange (Kok and Lange, 2015). Although they conclude that there is 'no direct unequivocal evidence for the existence of separate populations', they discuss several studies suggesting preferential encoding of errors and values by different neurons. For example, in a part of visual cortex (inferior temporal cortex), the inhibitory neurons tended to have higher responses to novel stimuli, while excitatory neurons typically produced highest response

for their preferred familiar stimuli (Woloszyn and Sheinberg, 2012). Kok and de Lange point that these responses may potentially reflect error and value nodes, respectively (Kok and Lange, 2015).

Each model accounts for specific aspects of experimental data. The models based on contrastive learning rules have been shown to reproduce neural activity and behaviour in a wide range of tasks (O'Reilly and Munakata, 2000). The learning rule in the continuous update model (in which the synaptic modification depends on the rate of change of the postsynaptic neuron; Figure 5.6A), can be implemented with classic spike-time-dependent plasticity (Figure 5.6B) (Bengio, 2017). In this form of plasticity, the direction of modification (increase or decrease) depends on whether the spike of a presynaptic neuron precedes or follows the postsynaptic spike (Bi and Poo, 1998). Figure 5.6C shows the effect of such plasticity in a case when the postsynaptic neuron increases its firing. If the postsynaptic spike follows the presynaptic spike, the synaptic weight is increased (pink area), while if the postsynaptic spike precedes the presynaptic spike, the weight is decreased (yellow area). If the postsynaptic neuron increases its firing rate (as in the example), there will be more postsynaptic spikes in pink than in yellow area on average, so the overall weight change will be positive. Analogously, the weight is weakened if the postsynaptic activity decreases (Figure 5.6D). In summary, with asymmetric spike-time-dependent plasticity, the direction of weight change depends on the gradient of a postsynaptic neuron activity around a presynaptic spike, as in the continuous update model.

The relationship of spike-time-dependent plasticity to other models requires further clarifying work. Nevertheless, Vogels and colleagues (Vogels et al., 2011) demonstrated that a learning rule in which the direction of modification depends on activity of neurons in equilibrium (Figure 5.6E), as in the predictive coding model, can arise from an alternate form of spike-time-dependent plasticity. They considered a form of plasticity where the weight is increased by nearly coincident pre- and postsynaptic spikes, irrespectively of their order, and additionally the weight is slightly decreased by each presynaptic spike. The overall direction of weight modification in this rule is shown in Figure 5.6F. Such a form of plasticity

may exist in a several types of synapse in the brain (Abbott and Nelson, 2000). Figure 5.6G illustrates that with such plasticity, the weights are increased if the intervals between pre- and postsynaptic spikes are short, which is likely to occur when the two neurons have high activity. When the postsynaptic neuron is less active (Figure 5.6H), the short intervals (pink area) are less common, while longer intervals are more common (yellow area), so overall the weight change is negative. In summary, with symmetric spike-time-dependent plasticity the direction of weight change depends on whether the postsynaptic neuron activity is above or below a certain level (which may correspond to a baseline level typically denoted with zero in computational models), as in the predictive coding model.

The dendritic error model describes the computations in apical dendrites of pyramidal neurons and features of cortical micro-circuitry such as connectivity of a group of interneurons called the Martinotti cells, which receive input from pyramidal neurons in the same cortical area (Silberberg and Markram, 2007) and project to their apical dendrites (Kubota, 2014). Furthermore, there is some evidence that inhibitory interneurons also receive feedback from higher areas in the cortical hierarchy (Leinweber et al., 2017).

5.4 Integrating Models

The above-mentioned comparison shows that each model has its own computational advantages, accounts for different data, and describes plasticity at different types of synapses. It is important to note that the cortical circuitry is much more complicated than any of the proposed models' architectures. Therefore, the models presented above need not be viewed as competitors but may be considered as descriptions of learning in different motifs of more complex brain networks.

Different classes of models may be more suited for different tasks faced by brain networks. One task engaging the primary sensory areas is predicting the next value of sensory input from the previous ones. A recent modelling study suggests that primary visual and auditory cortices may use an algorithm similar to back-propagation while learning to predict sensory input (Singer et al., 2018). This

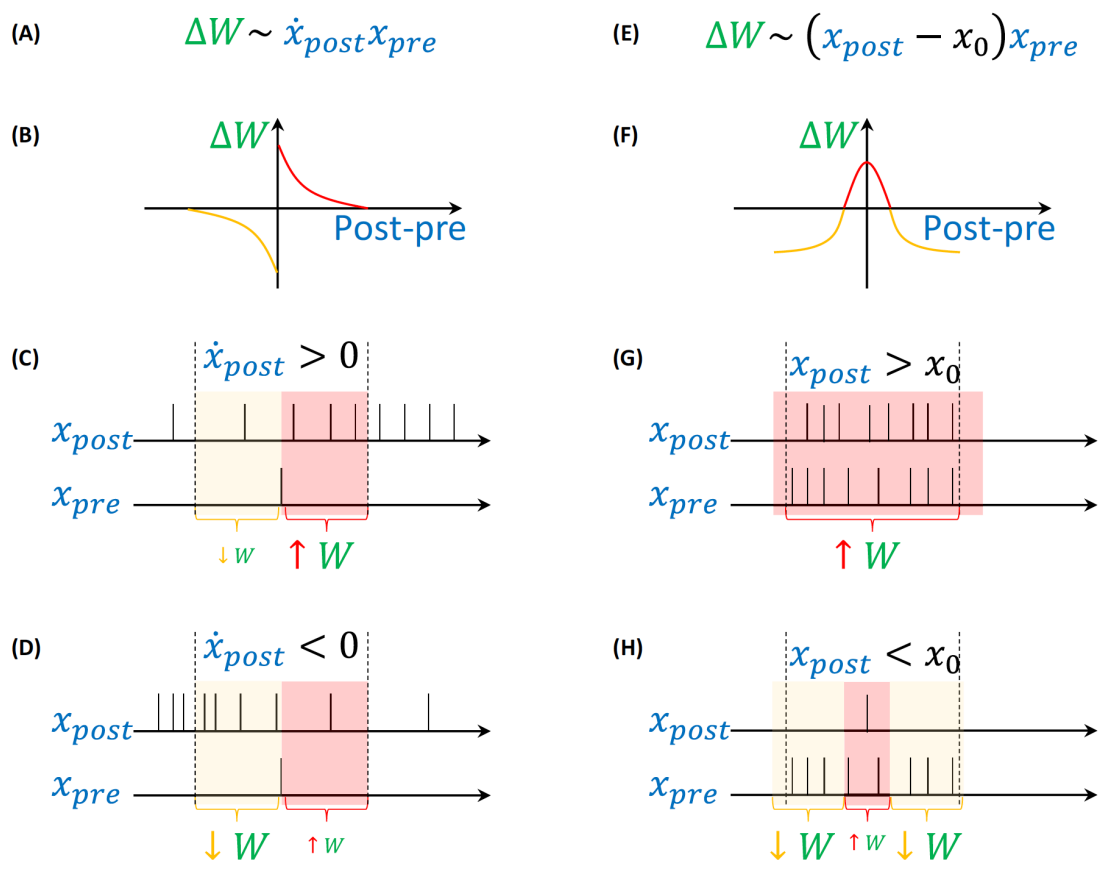

Figure 5.6: (A) Plasticity dependent on the rate of change of postsynaptic activity, illustrated by the left column of panels. (B) Asymmetric spike-time-dependent plasticity often observed in cortical neurons (Bi and Poo, 1998). The curve schematically shows the change in synaptic weights as a function of the difference between the timings of postsynaptic and presynaptic spikes. Red and orange parts of the curve correspond to increases and decreases in synaptic weights, respectively. (C) Strengthening of a synaptic weight due to increasing postsynaptic activity. Hypothetical spike trains of two neurons are shown. The top sequence corresponds to an output neuron, which increases its activity over time towards the target. The bottom sequence corresponds to a neuron in the hidden layer; for simplicity, only a single spike is shown. The pink and yellow areas correspond to spike timings in which the weights are increased and decreased, respectively. In these areas the differences in spike timing result in weight changes indicated by red and orange parts of the curve in the panel B. (D) Weakening of weight due to decrease in postsynaptic activity. (E) Plasticity dependent on postsynaptic activity, illustrated by the right column of panels. In the equation, $\mathbf{x}_0$ denotes the baseline firing rate. (F) Symmetric spike-time-dependent plasticity, where weight change depends on spike proximity. (G) Increase in synaptic weight due to high activity of the postsynaptic neuron. (H) Decrease in synaptic weight when the postsynaptic neurons is less active.

study demonstrated that the temporal properties of receptive field in these areas are similar to those in artificial neural networks trained to predict the next video or

audio frames on the basis of past history in clips of natural scenes (Singer et al., 2018). In such sensory prediction tasks, the target (i.e., the next 'frame' of sensory input) always arrives, so the temporal-error models may be particularly suited for this task, as there is no need for the control signal indicating target presence.

The explicit-error models are suitable for tasks where the timing of target pattern presentation is more uncertain. Although the predictive coding and dendritic error networks are closely related, they also exhibit a trade-off: the predictive coding networks are slow to propagate information once trained, while the dendritic error networks are slower to train. It is conceivable that cortical networks include elements of predictive coding networks in addition to dendritic error motifs, as the cortical networks include many other interneuron types in addition to the Martinotti cells and have a much richer organisation than either model. Such a combined network could initially rely on predictive coding motifs to support fast learning and, with time, the dendritic error models could take over, allowing faster information processing. Thus, by combining different motifs, brain networks may 'beat the trade-offs' and inherit advantages of each model.

Furthermore, predictive coding models may describe information processing in subcortical parts of brain networks that do not include pyramidal cells and thus may not be able to support computations of the dendritic error model. Indeed, it has been recently suggested how the predictive coding model can be mapped on the anatomy of cerebellum (Friston and Herreros, 2016), and the model may also describe aspects of information processing in basal ganglia, where the dopaminergic neurons are well known to encode reward prediction error in their activity (Schultz et al., 1997).

As the brain networks may incorporate elements of different models, it is important to understand how individual models relate to each other and how they can be combined. Such insights have been revealed by a recently proposed framework called equilibrium propagation (Scellier and Bengio, 2017a,b). Here, it was noticed that the dynamics of many models of neuronal networks can be defined in terms of the optimisation of a particular function. This function is known as the network energy. For example, recurrently connected networks of excitatory neurons, such as

the temporal-error models, under certain assumptions converge to an equilibrium in which strongly connected neurons tend to have similar levels of activity. Indeed, they minimise a function that summarises the dissimilarity in the activity of strongly connected nodes, called the Hopfield energy (Hopfield, 1984). The predictive coding networks are also known to minimise a function during their dynamics, called the free energy (Friston, 2005). The free energy has a particularly nice statistical interpretation, as its negative provides a lower bound on the log probability of predicting the target pattern by the network (Friston, 2005; Bogacz, 2017) (in case of supervised learning, this probability is conditioned on the input patterns). Since the dendritic error models have approximately similar dynamics as the predictive coding models, all models reviewed above can be considered as energy-based models described within the equilibrium propagation framework (Figure 5.7).

It has been shown that network error can be minimised if the synaptic weights are modified in two steps (Scellier and Bengio (2017a); schematically illustrated by the two displays in Figure 5.7). First, with only the input pattern provided, once the network converges, weights are modified in the direction in which the energy increases. Second, the output layer is additionally constrained to values closer to the target pattern (particular details described in (Scellier and Bengio, 2017a)). Constraining the output nodes changes the energy landscape for the units in the middle layers.

Once these units converge to a new equilibrium, weights are modified in the direction in which the energy decreases. Scellier and Bengio (Scellier and Bengio, 2017a) noted that for temporal-error networks, this procedure gives the contrastive learning rule (Equation 5.2). The predictive coding networks, however, converge to an equilibrium in the first step where the free-energy function reaches its global minimum (Whittington and Bogacz, 2017); thus, there is no weight modification required by the equilibrium propagation framework. Therefore, only a single phase (i.e., the second phase) and a single weight update are required in the explicit-error models, and it only involves Hebbian plasticity.

Importantly, the framework can describe learning in more complex networks, which could include the elements of the different models. For any network for which

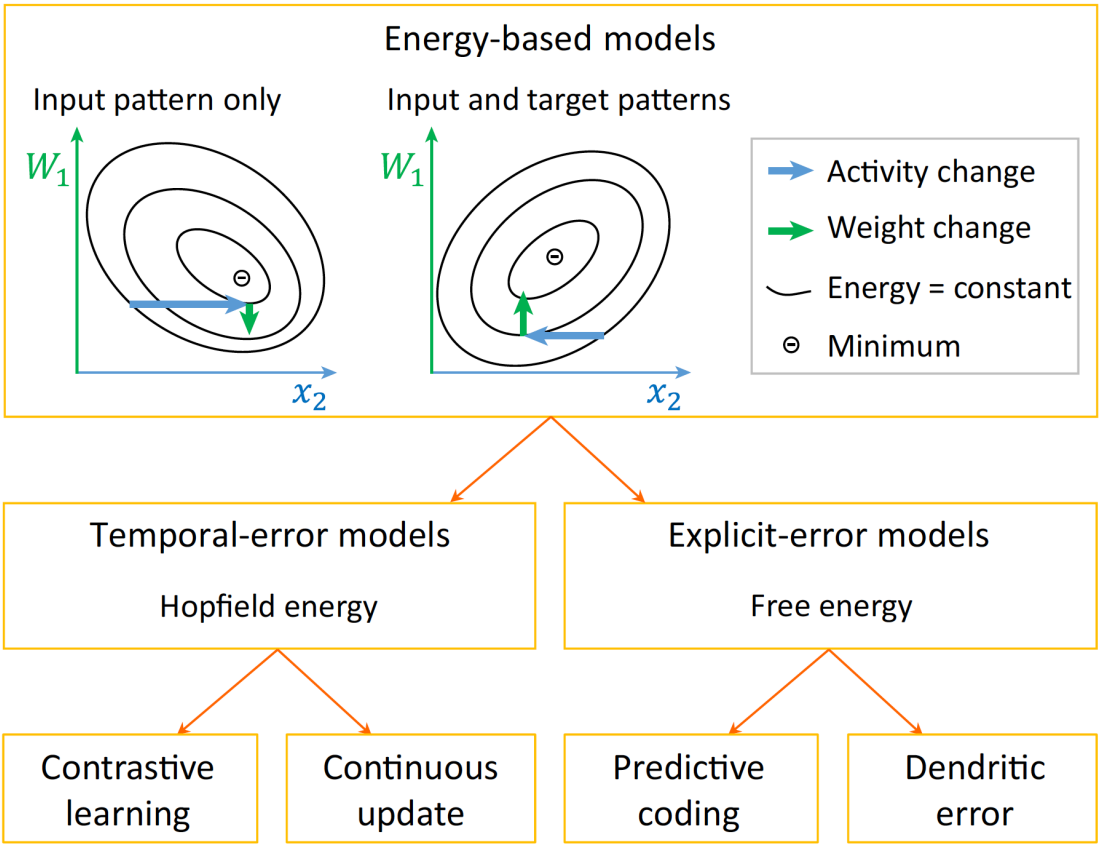

Figure 5.7: The framework considers networks with dynamics described by the minimisation of an energy function. As the activity of these networks converges to an equilibrium, the energy simultaneously decays (blue arrows) to a minimum given the current weights. Once in equilibrium, the weights are modified (green arrows).

an energy function can be defined, the framework describes the plasticity rules of individual synapses required for efficient learning.

Nevertheless, the form of energy function minimised by a network may influence its performance. So far, the biologically plausible networks that perform best in a handwritten digit classification task are those that minimise energies analogous to the free energy (Figure 5.5). The superior performance of networks minimising free energy may stem from the probabilistic interpretation of free energy, which ensures that the networks are trained to maximise the probability of predicting target patterns.

5.5 Remarks

This chapter has not been exhaustive of all current biological models but nevertheless has described main classes of recent models; those that represent errors temporally and those that represent them explicitly, as well as a framework unifying these methods. These theoretical results elucidate the constraints required for efficient learning in hierarchical networks. However, much more work needs to be done both empirically and theoretically, for example, on how the networks scale to larger architectures (Bartunov et al., 2018), as well as linking theory to neurobiological data.

It is crucial to map the models implementing efficient deep learning on biological networks in the brain. In particular, mapping the nodes in the model on distinct cell types in the cortex may be a fruitful route to identifying their computational function. The framework of equilibrium propagation (or its future extensions) may prove particularly useful in this endeavour. Based on known patterns of connectivity, models could be defined and their energy function formulated. The framework could then be used to predict properties of synaptic plasticity that could be compared with experimental data, and the results of such comparisons could be iteratively used to improve the models.

Part II

Representations for generalisation

6

Generalisation, space and relational memory in the hippocampal formation

Contents

You see a page, a page you have never seen before, a page of just white and black, a page you nevertheless piece together into a (hopefully!) coherent message. Though you have seen paper, ink, letters and words before, this particular configuration is new. Nevertheless, it is seamlessly integrated together. This seamlessness hides a veritable riot of neural activity, a riot ascending from your retina up the cortical hierarchy. Yet among that riot, there are neurons that notice an edge or letter or word or some more abstract quality that fire the same as the last time they saw it. Elements (neurons; bases; chapter 2) that have characteristic response to similarities in the world are rather useful, as similarities, by their nature,

generalise across experiences. Indeed, it is through learned similarities that one generalises knowledge. Holistically, understanding new experiences in the context of older learnings is generalisation. More formally, generalisation is quantified as an algorithm's (brain or machine) performance gap on seen-before tasks and never-seen-before tasks. Naturally a small generalisation gap is preferred as learning to run away from a tiger might also be a good strategy when faced with a lion.

In this chapter we delve deeper into the topic of generalisation in both machines and brains. While doing the brain part, we home in on a particular brain region - the hippocampal formation. Where possible, we refer back to chapter 2; generalisation and Bayesian thinking are very much linked. Finally we note that much of this chapter discusses similar, and often identical, content to Behrens et al. (2018), where, when similar, surely a more eloquent account is provided.

6.1 Generalisation

We start with an excessively formal and precise introduction to generalisation from statistical learning theory, then quickly damp it down to suit the mess of biology and this thesis.

In learning, we wish to find a function $f(\mathbf{x})$ accepting $\mathbf{x}$ as an input that successfully predicts output values of $\mathbf{y}$. Should the function be your brain, the input and output may be a question and answer, or sensory input separated in time. The expected error, E, of a particular function, f_n, is defined as

$$E[f_n] = \int_{\mathbf{x},\mathbf{y}} L\left(f_n(\mathbf{x}), \mathbf{y}\right) P(\mathbf{x}, \mathbf{y}) d\,\mathbf{x}\, d\,\mathbf{y} \tag{6.1}$$

where L is a loss function, and $P(\mathbf{x}, \mathbf{y})$ is the joint probability of $\mathbf{x}$ and $\mathbf{y}$. We don't know the joint probability, so we instead compute the empirical error on sample data $\mathcal{D}$

$$E_{\mathcal{D}}[f_n] = \frac{1}{n} \sum_{i=1}^{n} L\left(f_n(\mathbf{x}_i), \mathbf{y}_i\right) \tag{6.2}$$

An algorithm trained on sampled data, $\mathcal{D}_{train}$, will obtain a particular error, $E_{\mathcal{D}_{train}}[f_n]$. A low $E_{\mathcal{D}_{train}}[f_n]$ means something has been learned. The *generalisation* error, G, measures the difference in the algorithm's performance on training data, $\mathcal{D}_{train}$, and unseen data, $\mathcal{D}_{test}$

$$G = E_{\mathcal{D}_{test}}[f_n] - E_{\mathcal{D}_{train}} \tag{6.3}$$

An algorithm that generalises to new data has low G. Ideally we want both a low G and a low $E_{\mathcal{D}_{train}}$, as then we have learned and generalised knowledge. Often $G \gg 0$ due to *overfitting*. Overfitting is where the function $f(\mathbf{x})$ starts to fit noise in the data, as opposed to the true structure. This means that $E_{\mathcal{D}_{train}}$ becomes low, but $E_{\mathcal{D}_{test}}[f_n]$ becomes high. There are numerous techniques to avoid overfitting, including explicit regularisation (Tikhonov, 1943), under-parameterisation, and even over-parameterisation in neural networks (Advani and Saxe, 2017; Belkin et al., 2019).

Bias and variance

We can gain some insight into learning algorithms by considering a squared error loss, $L\left(f_n(\mathbf{x}), \mathbf{y}\right) \equiv \| \mathbf{y} - f_n(\mathbf{x})\|^2$, and assume a 'true' function, f, i.e. $\mathbf{y} = f(\mathbf{x}) + \boldsymbol{\varepsilon}$ where $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma)$. Here the expected error (equation 6.1) can be decomposed into two parts (Geman et al., 1992)

$$\mathbb{E}\,\| \mathbf{y} - f_n(\mathbf{x})\|^2 = \| \underbrace{\mathbb{E}\, f_n(\mathbf{x}) - \mathbb{E}\, f(\mathbf{x})}_{\text{bias}} \|^2 + \underbrace{\mathbb{E}\, \| f_n(\mathbf{x}) - \mathbb{E}\, f_n(\mathbf{x})\|^2}_{\text{variance}} + \sigma^2 \tag{6.4}$$

The first part is the bias of f_n from f and the second is the variance of f_n. This says that the error of a learning algorithm is the sum of how far it is off the true function (the bias squared) with how consistent algorithm is from one training run to the next (the variance). There is often a trade-off: high variance methods may overfit the training set and so generalise poorly, low variance methods may generalise well but have poor predictive power due to a bias.

Inductive biases

We want the best of all possible worlds - low variance for consistency and low bias to learn the true function - leading to low G for generalisation. To help on this quest, the learning algorithm can know something a priori: an inductive bias. Inductive biases embed knowledge about the world (data), that encourages the algorithm to learn the true f (or whatever the practitioner thinks is the true f).

With an appropriate configuration of weights, neural networks can approximate any function (Cybenko, 1989), though the learnability of this configuration is not certain. To encourage learning, machine learners embed inductive biases within their algorithms (Goodman, 1955; Mitchell, 1980). Convolutional neural networks (Lecun et al., 1998) have translational invariance of image statistics embedded via tying weights of filters together; a filter learned because of a tiger in the top left of an image, will detect a tiger in the bottom left of the next image. Similarly if we believe our data is generated from a recurrent mechanism, tying weights through time is useful (Rumelhart et al., 1988).

Back to Bayes, since the posterior is influenced by the prior, its distribution is biased to put greater weight on those values with higher prior weightings. Prior beliefs are thus inductive biases in the learning algorithm - learning and inference is done in the context of prior beliefs. For example, prior beliefs of simpler hypotheses encourage simpler posteriors and simpler solutions often generalise better (MacKay, 1992). In this vein, a prior can also be seen as regularisation.

Inductive biases, however, affect the generality of the algorithm. Convolutional neural networks may have a harder time than fully connected network to understand genomics. Finding appropriate inductive biases that help learning but don't hinder generality is key.

6.1.1 What gets generalised

For generalisation to be a useful strategy there must be regularities in the world to profit from. As chapter 2 details, symmetries of the natural world imply these regularities exist. These regularities can be at any level of abstraction - whether it

be of simple visual features; the perceptual qualities of a dog are the same whether it be in Woking or Tokyo, the behaviour of entities; the physics of orbits are the same for Earth as it is for any exoplanet (Mayor and Queloz, 1995), or the structure of stories; each one has a beginning, middle and end. Understanding regularities allows quick inferences; categorising a dog you have never seen before, inferring the object at the center of another's orbit is the (much) heavier one, inferring the news program is about to end after hearing a happy story.

Learning set

Like the examples above, regularities don't have to be simple perceptual statistics, but can be in the task itself. A classic experiment on learning such task statistics is Harlow's task (Figure 6.1A; Harlow (1949)), where participants (human and non-human primates) choose between two stimuli to find a reward. Critically, the reward is always associated with one of the stimuli and every 6 trials (a block) the stimulus set gets changed. If you know all this, it's pretty easy to get rewards - just repeat the rewarded choice or switch if unrewarded. However, if you don't know the task set-up, you may not expect the rewarded stimuli to stay the same for the 6 trials. This dichotomy of knowing and not knowing the task leads to an interesting phenomenon. During initial stages of the task (i.e. early blocks) subjects don't perform well, however as the task progresses, subjects get continually better at learning which stimuli was rewarded in each 6 trial block (Figure 6.1B). They learn how to learn. They learn the abstract rules/ knowledge of the task - 'one stimulus is rewarded and the other is not'. This is termed 'learning set' (Harlow, 1949).

A learning set requires understanding abstract relationships of the task. To facilitate this process, relational inductive biases can be introduced (Battaglia et al., 2018). These can either be explicit biases for specific tasks (Whittington et al., 2018) or soft biases that with enough data can learn in a variety of different task distributions ('meta-learning'; Finn et al. (2017); Andrychowicz et al. (2016); Wang et al. (2016)).

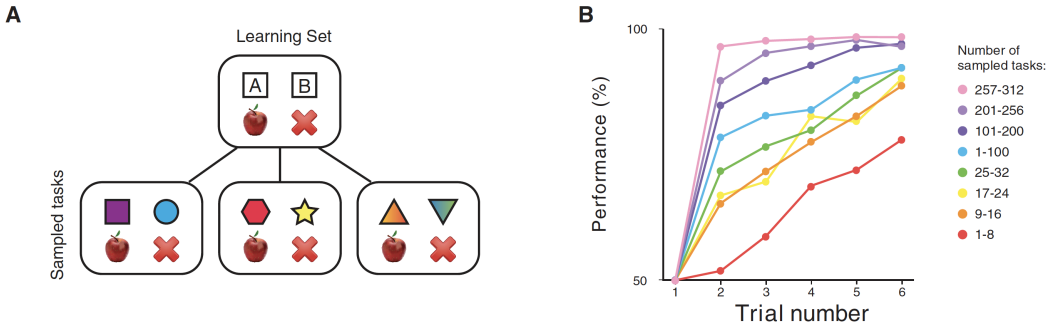

Figure 6.1: Harlow (1949) demonstrated the learning of task structure from repeated exposure to the same task. (A) Schematic of Harlow's task. Different instantiations of the task share a common underlying structure ('only one object is rewarded') that can be exploited to facilitate faster learning. (B) Accuracy data from Harlow (1949). Over multiple exposures to the task, monkeys acquire this underlying structure, termed a 'learning set', and use it to learn faster in new instantiations of the task. Figure adapted from Behrens et al. (2018)

6.2 A neuroscientist's perspective

Machine learning has come on leaps and bounds over the recent years (Lecun et al., 2015), now able to train artificial agents to super-human levels in games as diverse as Atari (Mnih et al., 2015) and Go (Silver et al., 2017). It is also clear however, that these algorithms are far from exhibiting the complex behaviours and flexible inferences of humans and other animals. Humans not only repeat previously good options, but also leverage past experience to understand new tasks or imagine new experiences. We are capable of generalisations machines can only dream of[1].

Other than Harlow's task, the canonical example of rich and flexible behaviour in neuroscience and psychology is from Edward Tolman (Tolman and Honzig, 1930). He had rats wander about mazes in the absence of rewards and later saw that they utilised these seemingly aimless behaviours when faced with tasks. The rats took short-cuts to reach rewards, and similarly, if old paths were blocked, they configured new routes on the fly (Tolman et al., 1946). Short-cuts and new routes are only possible if the rats knew the structure of the maze - if they had an internal map of space. Tolman named this internal organisation of the world a 'cognitive map'.

[1] If machines dream

6.2.1 Cognitive maps of space in the hippocampal formation

Though Tolman saw cognitive maps as a knowledge organising principle beyond space, it is with space and spatial cognition that these ideas really took hold. This is surely attributed to the beautiful and precise neural representations that map space found in the hippocampal formation (Figure 6.2A) The first of these cells were hippocampal 'place cells' (O'Keefe and Nadel, 1978) - cells that fire in single (or a couple) of locations in space (Figure 6.2B). Each cell only cares about a small region of space, but across the population they cover (map) the whole space. Completing the 2014 Nobel prize duo, are entorhinal 'grid cells' (Figure 6.2B). These, like place cells, care about locations in space, only this time they care about many locations, with each location lying quite remarkably on a triangular lattice (Hafting et al., 2005). These periodic cells tile the space, implying they know not only local information, but also something global. Indeed grid cells can be used as a metric, for vector relationships between locations (Bush et al., 2015; Stemmler et al., 2015), or for providing a basis for knowing where you are with your eyes closed - path integration (Sreenivasan and Fiete, 2011).

There is a plethora of other spatially selective cells (Figure 6.2B) that did not make the Nobel cut; band cells (Krupic et al., 2012); cells that encode the vector relationships to borders (Lever et al., 2009; Solstad et al., 2008), objects (Høydal et al., 2019), rewards (Gauthier and Tank, 2018), and goals (Sarel et al., 2017); cells that encode head direction (Taube et al., 1990); cells that encode the locations of other agents (Danjo et al., 2018; Omer et al., 2018).

Hierarchies of knowledge

Knowledge can be represented at more than one scale. When reminiscing on past holiday experiences, you likely first think of the country, then the city, then particular activities. Such hierarchies are found in the brain's cognitive maps. Grid (Brun et al., 2008; Stensola et al., 2012) and place cells Kjelstrup et al. (2008) have differing spatial scale or field size, respectively, along the dorsal ventral axis. For

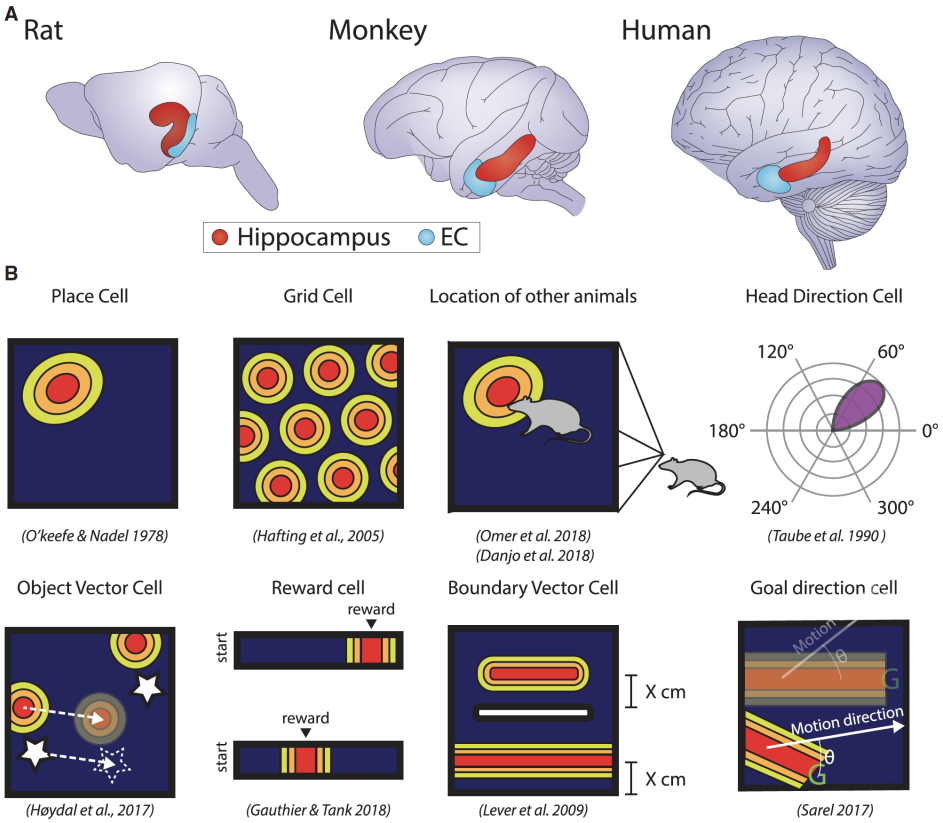

Figure 6.2: A) Anatomical location of the hippocampus and entorhinal cortex in different species. Adapted with permission from Strange et al. (2014). (B) A variety of cells in the hippocampal formation represent different spatial variables. Place cells (O'Keefe and Nadel, 1978) are active when an animal is in a single (sometimes multiple) location. Grid cells (Hafting et al., 2005) are active when an animal is in one of multiple locations on a triangular lattice. "Social place cells" (Danjo et al., 2018; Omer et al., 2018) are active in one animal when it observes that another animal is in a particular location. Head-direction cells (Taube et al., 1990) are active when an animal's head is facing a particular direction. Object-vector cells (Høydal et al., 2018) are active when an animal is in a particular direction and distance from any object. Reward cells (Gauthier and Tank, 2018) are active when an animal is in the vicinity of reward. Boundary vector cells (Lever et al., 2009) are active at a given distance away from a boundary in a particular allocentric orientation. Goal direction cells (Sarel et al., 2017) are active when the goal of an animal is in a particular direction relative to its current movement direction. The green "G" indicates the goal location. Figure adapted from Behrens et al. (2018)

grid cells these scales come in discrete modules, with each module approximately 1.5 times the scale of the previous.

Map representations generalise

Cells of the hippocampal formation have an intriguing property that obeys anatomical boundaries - entorhinal cells generalise and hippocampal cells don't. When

moving between structurally identical, but sensorally different environments, cells that were grid cells stay grid cells. Moreover, grid cells that were active next to each other, remain active next to each other (Fyhn et al. (2007); Figure 6.3A). In other words, grid cell correlation structure is preserved - grid cell activity stays on a set manifold. The cells may be phase shifted, but they all have the same phase shift - known as grid cell realignment. The above properties are true for each module independently. These suggest that the entorhinal map embeds knowledge common across environments - knowledge of 2D space.

Place cells, however, do not preserve their correlation structure, with neighbouring cells not necessarily neighbours in new environments. This is called hippocampal remapping (Bostock et al., 1991; Leutgeb et al., 2005). This entorhinal-hippocampal split is also true the other cell types e.g. object vector cells in entorhinal cortex activate for any object in the environment, whereas landmark cells in hippocampus activate for only a subset of objects (Deshmukh and Knierim (2013); Høydal et al. (2018); Figure 6.3B). Though the hippocampus plays a crucial role in generalisation, the cells themselves do not appear to generalise.

The hippocampus, however, does receive inputs from brain regions that generalise in different ways (and are factorised from each other; Manns and Eichenbaum (2006)). In medial entorhinal cortex the representations are of 2D maps as described above, whereas in lateral entorhinal cortex the representations are sensory. Hippocampal representations appear to be conjunctions of sensory and 2D structure - cells are active for a particular object in a particular location, but not for the object or location alone (Komorowski et al., 2009; Wood et al., 1999).

Asides from grid and place cells, the other cells only show their faces in specific circumstances (e.g. with borders or objects) - i.e. there is a collection of spatial representations that, depending on the environment, have certain weightings - perhaps basis functions (chapter 2).

Nevertheless, we have only spoken about cells mapping space, whereas the hippocampal formation has a wider role in generalization, inference, imagination,

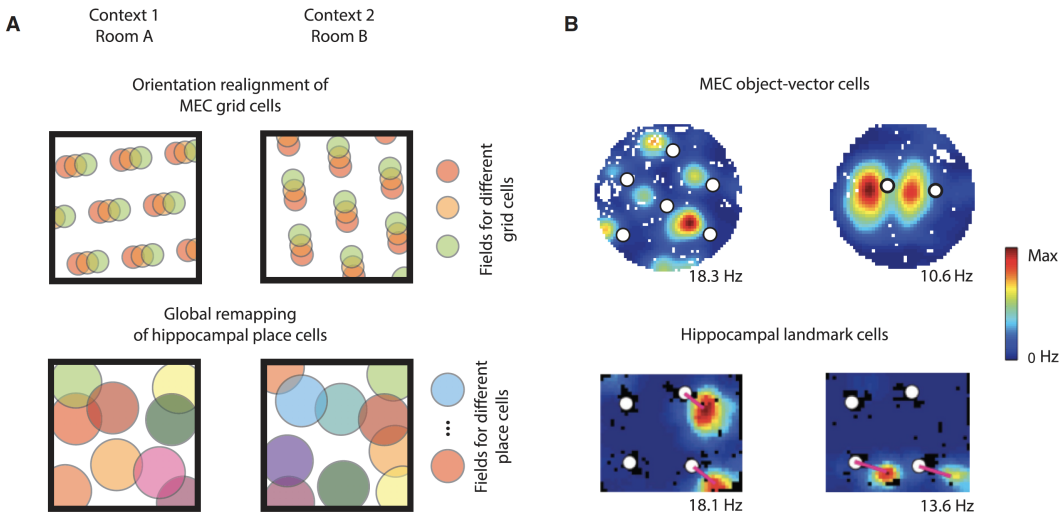

Figure 6.3: (A) In a spatial remapping experiment, animals are moved between two different environments. Entorhinal grid cells maintain a constant spatial phase structure (top), in contrast with the global remapping of hippocampal place cells (bottom) (Bostock et al., 1991; Leutgeb et al., 2005). (B) MEC object vector cells respond specifically when an animal is at a given direction and distance from any object, regardless of identity of the object (top). Cells with object-vector properties are also found in the hippocampus (bottom). These cells, however, respond to only a subset of the objects (bottom). MEC data from (Høydal et al., 2018); hippocampal data from (Deshmukh and Knierim, 2013). Figure adapted from Behrens et al. (2018)

social cognition, and memory (Hassabis et al., 2007; van der Meer et al., 2012; Ólafsdóttir et al., 2015; Tavares et al., 2015).

6.2.2 Mapping non-space

Tolman proposed the same principles organising spatial abstraction exist across all domains. Taking this seriously, we should be able to find non-spatial tasks that afford the same organisation as space, thus cells similar to the aforementioned spatial cells should map these spaces. Exactly this has been done, with positive results found in the same brain regions as spatial cells (Constantinescu et al., 2016; Aronov et al., 2017).

Asking humans to navigate 2D, non-spatial, spaces reveals a grid like code. In Constantinescu et al. (2016) birds with varying neck length (1st dimension) and leg length (2nd dimension) define a 2D space. Participants navigate the space via a joystick to reach several target birds associated with rewards. Grid-like coding was

found via a 6-fold oscillation in fMRI activity (Doeller et al., 2010) as a function of moving direction in both medial entorhinal cortex and ventral frontal cortex.

Similar games have been played with rodents; less complex tasks for less complex beings, though more complex recordings for more lenient animal rights. Training a rodent to hold a lever until a tone reaches a desired frequency is a 1D task as frequencies lie on a line. Electrophysiological recordings show hippocampal cells fire for certain frequencies (or places) and entorhinal cells (including a third of spatial grid cells) have multiple distinct fields. Other studies also present evidence of non-spatial coding using spatial representations e.g. grid cells also encode gaze location on a 2D image in both nonhuman (Killian and Buffalo, 2018) and human primates (Julian et al., 2018; Nau et al., 2018).

This generalisation of spatial representations to non-spatial problems suggests that grid and place cells are a reflection of a more general coding principle regarding the underlying relational structure / topology of tasks.

Other representations that generalise

Though our focus is on the hippocampal formation, there are other neural representations that generalise. Examples are complex cells of Hubel and Wiesel (1959) only care about edges, or neurons that respond to value of stimuli independent of the stimulus itself (Xie and Padoa-Schioppa, 2016).

6.2.3 Relational memory in the hippocampal formation

The spatial representations literature, from O'Keefe and Nadel (1978) to Hafting et al. (2005), though of great elegance, has seemed irreconcilable with the other major function of the hippocampus - relational/ declarative memory (Scoville and Milner, 1957; Cohen and Squire, 1980). The reliance of memory on the hippocampus was first discovered via H.M., a bilateral hippocampal resection patient (Scoville and Milner, 1957; Cohen and Squire, 1980). H.M. had particularly bad amnesia, with the impairment described as 'forgetting the incidents of daily life as fast as they occur' (Scoville and Milner, 1957). The distinction of declarative memory

came from (Cohen and Squire, 1980), who noted that amnesic patients could learn procedural / rule based skills as competently as controls - it was the 'knowing that' rather than the 'knowing how' that hippocampus governed.

Cohen and Eichenbaum (1993) extended hippocampus and declarative memory to more general relational processing termed 'relational memory'. Relational memories bind together parts of experience, linking memories via their common elements. Linked memories provide a platform for arbitrary recombinations and generalisation.

The classic relational memory paradigm is transitive inference (Burt, 1911; Dusek and Eichenbaum, 1997; Mcgonigle and Chalmers, 1977). Here relational memories allow inferences in never-seen-before choices. For example, if an animal is trained to prefer A over B, and separately B over C, then one can then ask whether it will prefer A or C. Though the animal has never seen this choice before, it will choose A. This is transitive inference; A is better than B, and B is better than C, therefore A is better than C. Transitive inference depends on hippocampus and its inputs (Dusek and Eichenbaum, 1997; Gilboa et al., 2014; Wikenheiser and Schoenbaum, 2016; Buckmaster et al., 2004) - lesions to these structures impairs the ability to make the novel inferences (A vs C) but preserves the seen before choices (A vs B). There can be many more than 3 stimuli, and separate sequences can be subsequently stitched together (Treichler and Van Tilburg, 1996). This line-like ordinal structure of A>B>C etc, has clear parallels to the linear ordering of sound frequencies that we saw above as well as the linear ordering of temporal events; perhaps it is no surprise that one sees hippocampal cells that code for time in a task (time cells; (MacDonald et al., 2011). Other relational memory tasks have also shown hippocampal dependence e.g. decisions based on social value (Kumaran et al., 2012).

6.3 Dreams of a unifying theory

In the sections above, a variety of different functions and representations of the hippocampal formation were discussed. Here we begin to consider how the different functions can be bound together, and how this may offer insights into representations.

6.3.1 The schism of space and relational memory

Does hippocampus represent relational memories, or does it support spatial navigation via path integration? Here, paraphrasing Eichenbaum and Cohen (2014), we discuss how hippocampal processing cannot be solely for spatial processing.

Eichenbaum and Cohen (2014) describe that 1) spatial cognition is often based on memories of routes and destination, 2) path integrations alone is extremely noisy with humans inaccurate after just a few body lengths (Kim et al., 2013), 3) path integration is still intact post hippocampal damage (Etienne and Jeffery, 2004), and 4) place fields are conjunctive with specific events (Komorowski et al., 2009). Eichenbaum and Cohen (2014) dive deeper into these arguments, leaving the reader with the prospect that hippocampus is likely not about space per se, but its involvement in spatial cognition is one of relational memory.

6.3.2 A common framework for space and non-space

Though unifying hippocampal functions of spatial cognition with relational memory is an open problem, there have been strides bringing space and non-space together in the hippocampus through a framework of transitions and path integration. To understand the framework we consider a formalism of how states relate to each other.

Consider a set of states (a graph) with transition matrix, T, that tells us the transition probabilities, T_{ij}, of state j to state i. This matrix describes the pairwise relations between states, and so describes things like going in a loop takes you back to the same place. It also tells you the likely states you'll be in; multiplying the current distribution over state occupancy, $\mathbf{s}$, on the left gives $T\,\mathbf{s}$ which is the expected occupancy distributions after one step. In general, $T^n\,\mathbf{s}$ gives the state occupancy distribution after n steps. Abstract this may be, but an appropriate form of this matrix links together spatial and non-spatial coding and offers a framework for cognitive maps in the hippocampal formation.

If this matrix can be diagonalized $T = Q\Lambda Q^T$, where $Q = [\mathbf{v}_1, \mathbf{v}_2 ...]$ is an orthogonal matrix of eigenvectors, $\mathbf{v}$, and Λ is a diagonal matrix of associated eigenvalues, then it turns out the eigenvectors, for 2D graphs, are very reminiscent

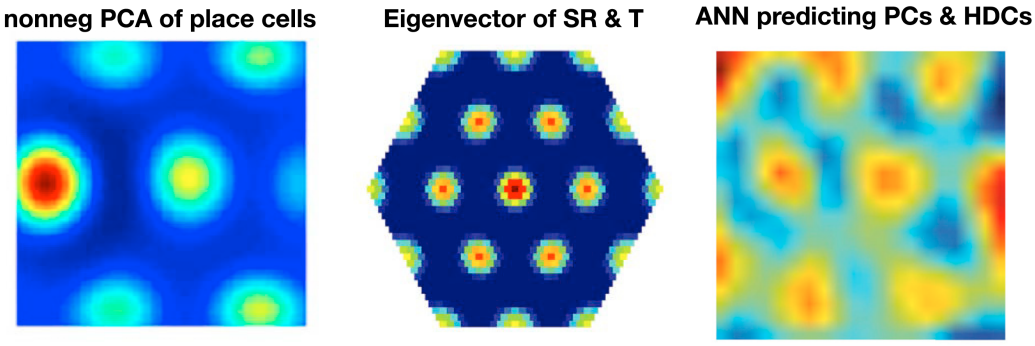

Figure 6.4: Grid-cell-like firing fields can be obtained by casting 2D space under this state-space framework. The eigenvectors of the covariance of 2D-distributed place cells (obtained by non-negative principal component analysis, PCA), of 2D transition matrices, and of successor representations of 2D state spaces are all grid-like, as are units of an artificial neural network (ANN) tasked with predicting 2D-distributed place cells (PCs) and head-direction cells (HDCs). Figures adapted from(Dordek et al., 2016) (left), (Stachenfeld et al., 2017) (middle) and (Banino et al., 2018) (right).

of grid cells (Stachenfeld et al. (2017); Figure 6.4 middle). Additionally, T can be written as $\sum_i \lambda_i \, \mathbf{v}_i \, \mathbf{v}_i^T$ and thus $T^n = Q\Lambda^n Q^T = \sum_i \lambda_i^n \, \mathbf{v} \, \mathbf{v}^T$. The distribution over states after n steps is then just a simple linear combination of eigenvectors

$$T^n \, \mathbf{s} = \sum_i \lambda_i^n \, \mathbf{v} \left[\mathbf{v}^T \, \mathbf{s} \right] \tag{6.5}$$

Where $\left[\mathbf{v}^T \, \mathbf{s} \right]$ is an inner product. This means that distances between states can be computed with ease without expensive lookahead algorithms (Baram et al., 2018) - a desirable property for the brain. So it turns out these eigenvectors not only represent space like the brain does, but are also what you need for future plans.

Place-like coding can be found from a similar matrix. This matrix, S, known as the successor representation (Dayan, 1993) describes the expected discounted future occupancy $S = I + \gamma T + \gamma^2 T^2 \cdots = (I - \gamma T)^{-1}$. This matrix has the same eigenvectors as T and thus the same grid like properties. Place-like activity comes from the rows of this matrix (Stachenfeld et al., 2017). This overall predictive framework of the successor representation accounts for properties of place and grid cells, and has been shown to be consistent with re-planning representations observed in hippocampus (Garvert et al., 2017; Momennejad et al., 2017).

It has long been proposed that cortical areas extract the statistics of hippocampal memories (complementary learning systems; McClelland et al. (1995)). In this vein, and similar to the above transition story, eigenvectors from principal component analysis of place cell activity are periodic, and hexagonal grid-like when constrained to be non-negative ((Dordek et al., 2016); Figure 6.4 left). Other models also obtain interesting representations. Recurrent neural networks trained on either $[x, y]$ coordinates (Cueva and Wei, 2018) or place cell representations (Banino et al., 2018; Sorscher et al., 2019) (Figure 6.4 right), obtain grid-like representations. Here the grid cells must provide a basis for path integration.

These above frameworks, though linking space and non-space through path integration as well as providing an understanding of hippocampal formation representations, do not offer insights into, or explanations of, generalisation and relational memories in the hippocampal formation.

7

The Tolman-Eichenbaum Machine for neuroscientists

Contents

The hippocampal-entorhinal system is important for spatial and relational memory tasks. We formally link these domains; provide a mechanistic understanding of the hippocampal role in generalisation; and offer unifying principles underlying many entorhinal and hippocampal cell-types. We propose medial entorhinal cells form a basis describing structural knowledge, and hippocampal cells link this basis

with sensory representations. Adopting these principles, we introduce the Tolman-Eichenbaum machine (TEM). After learning, TEM entorhinal cells display diverse properties resembling apparently bespoke spatial responses, such as grid, band, border and object-vector cells. TEM hippocampal cells include place and landmark cells, that remap between environments. Crucially, TEM also predicts empirically recorded representations in complex non-spatial tasks. TEM predicts hippocampal remapping is not random as previously believed. Rather structural knowledge is preserved across environments. We confirm this structural transfer over remapping in simultaneously recorded place and grid cells.

Here we give a clear and concise description of TEM, designed for neuroscientifically minded readers to enjoy. For those with a greater appetite for equations, details and machine learning, we point you to chapter 8 where further particulars on the task and model can be found.

7.1 Introduction

Humans and other animals make complex inferences from sparse observations and rapidly integrate new knowledge to control their behaviour. Tolman argued that these facilities rely on a systematic organisation of knowledge called a cognitive map (Tolman, 1948). In the hippocampal formation, during spatial tasks, individual neurons appear precisely tuned to bespoke features of this mapping problem (O'Keefe and Nadel, 1978; Taube et al., 1990; Hafting et al., 2005). However, hippocampus is also critical for non-spatial inferences that rely on understanding the relationships or associations between objects and events - termed relational memory (Cohen and Eichenbaum, 1993). Whilst it has been suggested that relational memory and spatial reasoning might be related by a common mechanism (Eichenbaum and Cohen, 2014), it remains unclear whether such a mechanism exists or how it could account for the diverse array of apparently bespoke spatial cell types.

One promising approach casts spatial and non-spatial problems as a connected graph, with neural responses as efficient representations of this graph (Gustafson and Daw, 2011; Stachenfeld et al., 2017). This has led to new potential interpretations for

place cells (Gustafson and Daw, 2011) and grid cells (Stachenfeld et al., 2017; Dordek et al., 2016). However, such approaches cannot account for the rapid inferences and generalisations characteristic of hippocampal function in both spatial and relational memory, and do not explain the myriad types of spatial representations observed, or predict how they will change across different environments (remapping).

We aim to account for this broad set of hippocampal properties by re-casting both spatial and relational memory problems as examples of structural abstraction (Kemp and Tenenbaum, 2008) and generalisation (Figure 7.1A-C). Spatial reasoning can be cast as structural generalisation, as different spatial environments share the common regularities of Euclidean space that define which inferences can be made, and which shortcuts might exist. For example, moving $south \rightarrow east \rightarrow north \rightarrow west$ will return you to where you started. Structural regularities also permit inferences in non-spatial relational problems. For example, transitive inference problems (which depend on hippocampus; Bunsey and Eichenbaum (1996); Dusek and Eichenbaum (1997)) require stimuli to be represented on an abstract ordered line, such that $A > B$ and $B > C$ implies $A > C$. Similarly, abstraction of hierarchical structure permits rapid inferences when encountering new social situations.

Structural generalisation offers dramatic benefits for new learning and flexible inference, and is a key issue in artificial intelligence. One promising approach is to maintain "factorised" representations in which different aspects of knowledge are represented separately and can then be flexibly re-combined to represent novel experiences (Higgins et al., 2017a). Factorising the relationships between experiences from the content of each experience could offer a powerful mechanism for generalising this structural knowledge to new situations. Notably, exactly such a factorisation exists between sensory and spatial representations in medial and lateral entorhinal cortices respectively. Manns and Eichenbaum propose that novel conjunctions of these two representations form the hippocampal representation for relational memory (Manns and Eichenbaum, 2006).

We demonstrate that this factorisation and conjunction approach is sufficient to build a relational memory system (the Tolman-Eichenbaum machine; TEM) that

generalises structural knowledge in space and non-space; predicts a broad range of neuronal representations observed in spatial and relational memory tasks; and accounts for observed remapping phenomena in both hippocampus and entorhinal cortex. Notably, although hippocampal remapping is thought to be random, TEM predicts that this apparent randomness hides a structural representation that is preserved across environments. We verify this prediction in simultaneously recorded place and grid cells. These results suggest a general framework for hippocampal-entorhinal representation, inference and generalisation across spatial and non-spatial tasks.

7.2 Results

7.2.1 Spatial and relational inferences can be cast as structural generalisation.

We consider the unsupervised learning problem where an agent must predict the next sensory experience in a sequence derived from probabilistic transitions on graphs (Figure 7.1D). The agent does not see the graph, only a sequence of sensory observations and the "action" or "relation" that caused each transition. Different types of relation exist, e.g. in a family hierarchy, parent, aunt, child and nephew imply different transitions on the graph, but each transition-type has the same meaning at every point on the graph. Similarly, in space, action is defined by heading direction (e.g. NESW on 4-connected graphs).

If all transitions have been experienced, the graph can be stored in memory and perfect predictions made without any structural abstraction. However, if structural properties of the graph are known a priori, perfect prediction is possible long before all transitions have been experienced; it only requires each node to have been experienced (Figure 7.2B,C): This can be easily understood: When the structure of the graph is known, a new node can be introduced with a single relation (Bob has a daughter, Emily - Figure 7.1E) and all other relations can immediately be inferred (Emily is Alice's granddaughter and Cat's niece etc.). Similarly, in space, if the structure of 2D graphs is known, then placing a new

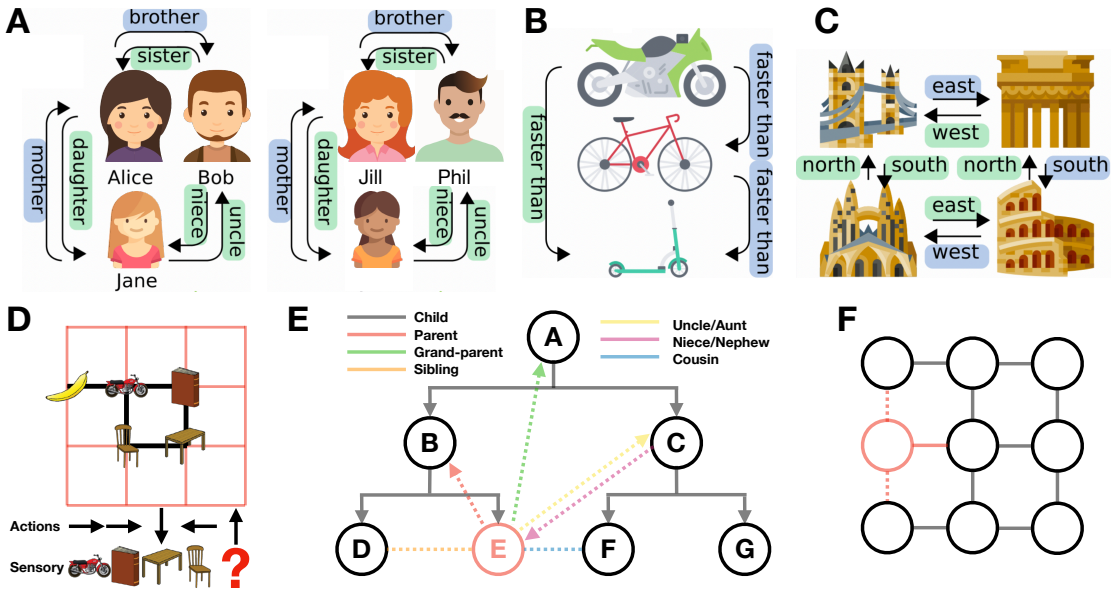

Figure 7.1: Spatial and relational inferences can be cast as structural generalisation. Structured relationships exist in many situations, and can often be formalised on a connected graph e.g. **A)** Social hierarchies, **B)** transitive inference and **C)** spatial reasoning. Often the same relationships generalise across different sets of sensory objects (e.g. left/right in **A)**). This transferable structure allows quick inference, e.g. seeing only the blue relationships allows you to infer the green ones. **D)** Our task is predicting the next sensory observation in sequences derived from probabilistic transitions on the graph. We use arbitrary sensory experiences on each node, e.g. a banana. An agent transitions on the graph observing only the immediate sensory stimuli and associated action taken. For example having seen *motorbike* → *book* → *table* → *chair*, it should predict the *motorbike* next if it understands the rules of the graph. **E)** Should you know the underlying structure of social hierarchies, when observing a new node (in red) via a single relationship - e.g. Emily is Bob's daughter - immediately allows inference about the new node's (Emily's) relationship to all other nodes (shown in black/gray). **F)** Similarly for spatial graphs observing a new (red) node 'on the left' (solid red line) also tells us whether it is 'above' or 'below' (dashed red lines) other surrounding nodes. Icons made by Nikita Gobulev (transport), FreePik (faces) and Abib Sulthon (landmarks) from flaticons.com.

node on an X-Y coordinate is sufficient to infer relational information to every other point on the graph (Figure 7.1F).

In summary, after experiencing many graphs with different sensory observations and learning their common relational structure, an agent should maximise its ability to predict the next sensory observation after each transition on a new graph (Figure 7.2A).

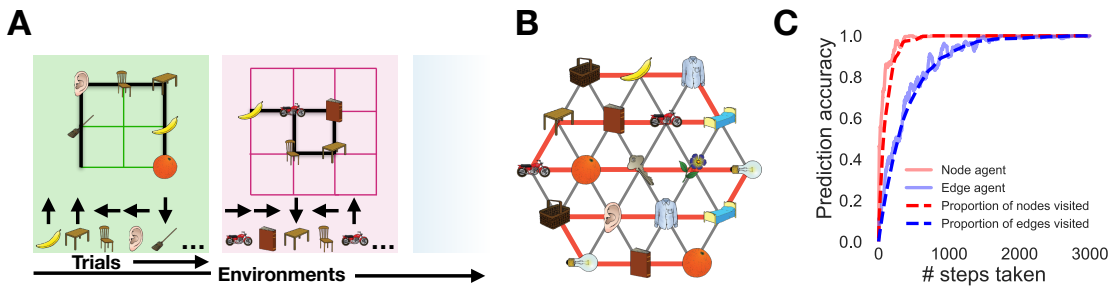

Figure 7.2: Spatial and relational inferences can be cast as structural generalisation. G) Our agent performs this next step prediction task in many different worlds which share the same underlying structure (e.g. 6- or 4-connected graphs), but they differ in their size and arrangement of sensory stimuli. The agent must learn the commonality amongst all the worlds (the shared graph structure) in order to generalise and perform quick inferences. **H)** Knowing the structure allows full graph understanding after only visiting all nodes and not all edges; here only 18 steps were taken (red line) but a perfect agent infers all 42 links. **I)** An agent that knows structure (node agent) will reach peak predictive performance after it has visited all nodes; quicker than one that has to see all transitions (edge agent). Further details on tasks can be found in chapter 8 and appendix C.

7.2.2 The Tolman Eichenbaum machine.

To build a machine that solves this problem, we first consider a normative solution. We then show that a tractable approximation to this solution maps simply onto the functional anatomy of the hippocampal system. We give complete descriptions of TEM in chapter 8, at both intuitive and detailed levels of description.

Generative model

We want to estimate the probability of the next sensory observation given all previous observations on this and all other graphs. A parsimonious solution will reflect the fact that each task is composed of two factors, a graph-structure and sensory observations (Figure 7.3A).

This problem can be expressed graphically as a generative model (Figures 7.3B and 7.4 red), where latent variables are positions that result from taking relational actions in a cognitive map. To facilitate generalisation of knowledge across domains we separate latent variables of abstract location that generalise across maps, $\mathbf{g}$, from those that are *grounded* in sensory experience and therefore specific to a particular map $\mathbf{p}$. Each variable is represented as a population (vector) of units. Conditioned

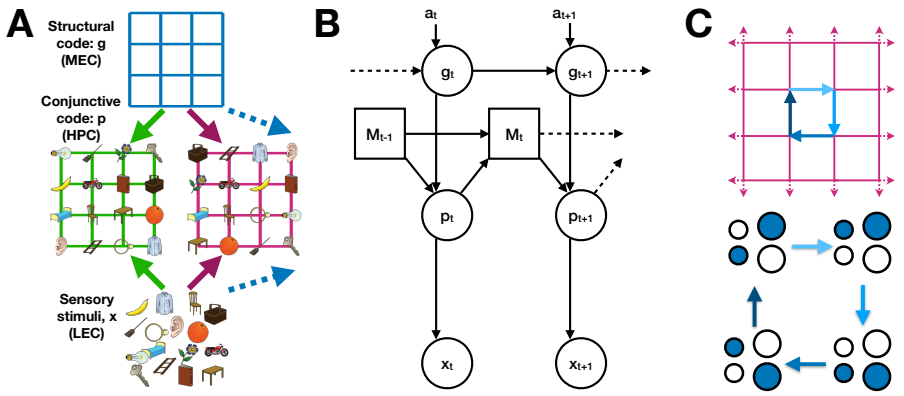

Figure 7.3: The Tolman-Eichenbaum Machine. A) Factorisation and conjunction as a principle for generalisation. Separating structural codes (the transition rules of the graph) from the sensory codes (the sensory objects on each node) allows generalisation over environments that share the same structure. The conjunctive code represents the current environment in the context of this learned structure. **B)** The generative model of TEM with observed variables $\mathbf{x}_t$ and $\mathbf{a}_t$ (sensory experience and action taken respectively), and latent variables $\mathbf{g}_t$ and $\mathbf{p}_t$ (where I am and a memory of what is where respectively). This is a graphical model which shows the dependencies of variables, not the connections. See Figure 7.4 (red) for equivalent connections. TEM transitions through latent variables $\mathbf{g}$, and retrieves memories $\mathbf{p}$ using Hebbian weights M. **C)** TEM must learn structural codes that 1) have a different representation for each state so that different memories can be stored and retrieved, but also 2) have the same code on returning to a state (from any direction) so the appropriate memory can be retrieved.

on an action, $\mathbf{g}$ transitions to a new 'abstract location'. This new $\mathbf{g}$ can then index Hebbian memories M, retrieving a memory of 'grounded location' $\mathbf{p}$, that, if the state has been visited before, can correctly predict the sensory experience $\mathbf{x}$.

This *generative model* allows sensory data to be simulated from an arbitrary map but, to be useful, we must also be able to *infer* the latent variables $[\mathbf{p}, \mathbf{g}]$ from real sensory data, so that predictions can be made about **this** map. In Bayesian reasoning, this involves ***inverting*** the generative model. The exact solution to this problem is intractable, so we turn to approximate methods.

Inference and the hippocampal formation.

We use modern Bayesian methods (Kingma and Welling, 2013; Gemici et al., 2017) to *learn* an inference network (Figure 7.4 green) that maps the stream of observations and actions to the latent variables, updating abstract location, $\mathbf{g}$, and grounded location, $\mathbf{p}$, as new sensory data appear. This inference network is independent

of the generative model, but, once trained, approximates optimal inference on the generative model's latent variables (See **Model training** and chapter 8).

Although the parameters of the inference network are learned, we are able to make critical choices in its architecture. The resulting network maps simply onto the functional anatomy of the hippocampal formation and its computations and can be intuitively represented in schematics (Figure 7.4 green).

Following Eichenbaum (Manns and Eichenbaum, 2006), hippocampal representations, ($\mathbf{p}$), are inferred as a conjunction between sensory input ($\mathbf{x}$) in the lateral (LEC), and abstract location ($\mathbf{g}$) in the medial (MEC) entorhinal cortex. Mirroring hippocampal synaptic potentiation (Bliss and Collingridge, 1993), this enables memories to be rapidly stored in weights (M) between $\mathbf{p}$ using simple Hebbian learning between co-active neurons; and retrieved by the natural attractor dynamics of the resultant auto-associative network (Figure 7.4 middle).

To infer a new $\mathbf{g}$ representation, TEM performs path integration from the previous $\mathbf{g}$, conditional on the current action/relation. This can be related to continuous attractor models (CANNS, Zhang (1996)) of grid cells (Burak and Fiete, 2009). Like CANNs, different recurrent weights mediate the effects of different actions/relations in a recurrent neural network (Figure 7.4 middle). Unlike CANNs, however, weights are not hardcoded, but **learnt from experience**, allowing map-like abstractions and path integration to extend to arbitrary non-spatial problems.

Path integration accumulates errors (Mittelstaedt and Mittelstaedt, 1980). To overcome this problem TEM can take advantage of a second source of information about $\mathbf{g}$ - the conjunctive representations, $\mathbf{p}$, stored in the hippocampal memory M. TEM indexes M with the current sensory experience, $\mathbf{x}$, to retrieve a set of candidate representations of $\mathbf{g}$ (previously visited places with a similar sensory experience), and uses these to refine the path integrated $\mathbf{g}$ via a Bayesian update.

When representing tasks that have self-repeating structure, it is efficient to organise cognitive maps hierarchically. To allow such hierarchy to emerge, we separate our model into multiple parallel streams, each as described above. These streams are only combined when retrieving $\mathbf{p}$ in the attractor network (see chapter 8 for details).

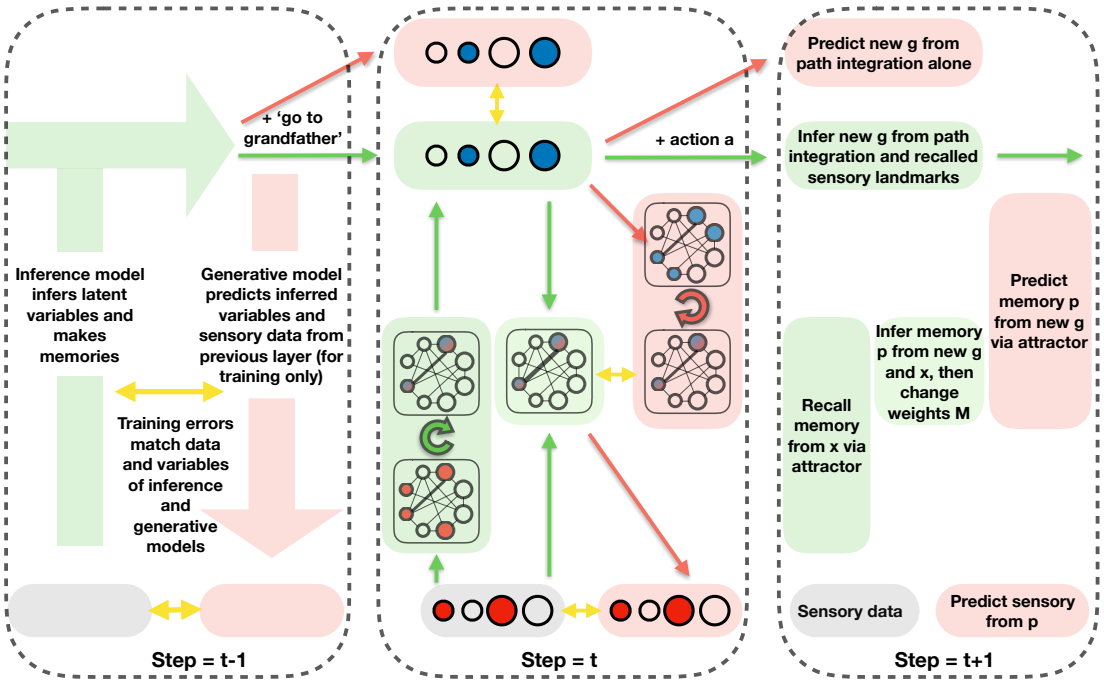

Figure 7.4: The Tolman-Eichenbaum Machine. Depiction of TEM at three time-points, with each time-point described at a different level of detail. Green/ red show inference and generative networks. Timepoint $t-1$ shows the overall Bayesian logic, t shows network implementation, $t+1$ describes each computation in words. Circles depict neurons (blue is $\mathbf{g}$, red is $\mathbf{x}$, blue/red is $\mathbf{p}$); shaded boxes depict computation steps; arrows show learnable weights (green and red are weights in inference and generative networks); looped arrows describe recurrent attractor. Black lines between neurons in attractor describe Hebbian weights M. Yellow arrows show training errors.

Model training

Both the generative and inference models have weights that must be learnt. The objective of training is for the generative model to *predict the sensory input*, $\mathbf{x}$, and for the inference model to *infer the generative model's latent variables*, $[\mathbf{p}, \mathbf{g}]$, from the sensory input. The resulting training algorithm (Figure 7.4) involves an interplay between generative (red) and inference (green) models, in which the generative model takes the current state of the inference model and (from this) predicts its next state (including the next sensory data). This leads to errors between the predicted and inferred/observed variables at each level $[\mathbf{p}, \mathbf{g}, \mathbf{x}]$. The weights in both networks are adjusted along a gradient that reduces these errors using backpropagation through time (see chapter 8).

This scheme is similar to the Wake Sleep algorithm (Hinton et al., 1995)

and Helmholtz Machine, which learn generative and inference models for sensory inference (Dayan et al., 1995). Unlike these sensory models through, TEM predicts temporal sequences (Gemici et al., 2017) and uses an attractor memory to allow rapid generalisation of structural knowledge. The reliance on generative model predictions is notable as hippocampal replay appears to sample from a generative model of the environment (Foster and Wilson, 2006; O'Neill et al., 2017).

The model is trained in multiple different environments, differing in size and sensory experience. Different environments use the same network weights, but different Hebbian weights, M. The most important weights are those that transition **g** as they encode the structure of the map. They must ensure 1) that each location in the map has a different **g** representation (so a unique memory can be built), 2) that arriving at the same location after different actions causes the same **g** representation (so the same memory can be retrieved) - a form of path integration for arbitrary graph structures. For example, the relation "uncle" must cause the same change in **g** as father followed by brother, but different from brother followed by father.

7.2.3 TEM generalises structural knowledge to novel sensory environments.

We first test TEM on classic non-spatial relational memory tasks thought to depend on hippocampus - transitive inference and social hierarchy tasks[1]. After training, TEM makes first presentation inferences in novel transitive inference tasks (Figure 7.5F) e.g. regardless of particular sensory identities (e.g. A, B, C, D, E or $cat, dog, elephant, fox, badger$) after being shown sequences such as $A > B > C > D > E$, TEM answers 'B' to the query 'what is 3 more than E', i.e. TEM has learned ordinal structural knowledge. Equally, in social hierarchy tasks, TEM infers relationships based on limited information (Figure 7.5E) e.g. after being shown that 'Bob is the brother of Cat who is Fran's mother', TEM answers 'Fran' when queried 'who is Bob's niece?'. In both cases, although TEM had never seen the particular sensory details of the task before, it had seen similar relational structures

[1]Further details on all TEM simulations and analyses can be found in appendix C.

which it could learn from and generalise. Such first presentation inferences we refer to as 'zero-shot' inference - *only* possible with learned structural knowledge.

Knowing the underlying structure allows one to know the entire relational structure after a single visit to each state (Figure 7.2B,C). TEM demonstrates this data efficiency with its performance in line with the proportion of states visited in the graph, not the edges taken (Figure 7.5D).

We also test TEM on tasks with an underlying spatial structure (e.g. Figure 7.1F,7.2B). Again, TEM performs zero-shot inference in spatial generalisation tasks (Figure 7.5C) - only possible with learned structural knowledge, whilst also storing long term relational memories (Figure 7.5B).

7.2.4 TEM represents structure with grid cells that generalise across environments

We begin by considering TEM agents that diffuse from state to state at random on 2D graphs, constrained only by the neighbourhood transitions in the environment. Here, TEM's 'abstract location' ($\mathbf{g}$) representations resemble grid cells (Figure 7.6A) and band cells (Figure 7.6B) as recorded in rodent medial entorhinal cortex (Hafting et al., 2005; Krupic et al., 2012; Banino et al., 2018; Cueva and Wei, 2018). As in the brain, we observe modules of grid cells at different spatial frequencies and, within module, we observe cells at different grid phases (Figure 7.6A).

TEM's top-level ($\mathbf{g}$) representations reflect the need to be both maximally different at different spatial locations, to enable independent memories at each location and invariant to approaching the same location from different trajectories (path integration) so that the correct memory can be retrieved. Our results suggest that these two constraints are sufficient to produce grid- and band- like representations. Further cell representations can be found in appendix C.

Importantly, top-layer TEM representations generalise, retaining their properties across different environments. This is true of both the first and second-order properties of the population. For example, a grid cell in environment 1 is a grid cell of the same frequency in environment 2, and the correlation structure across

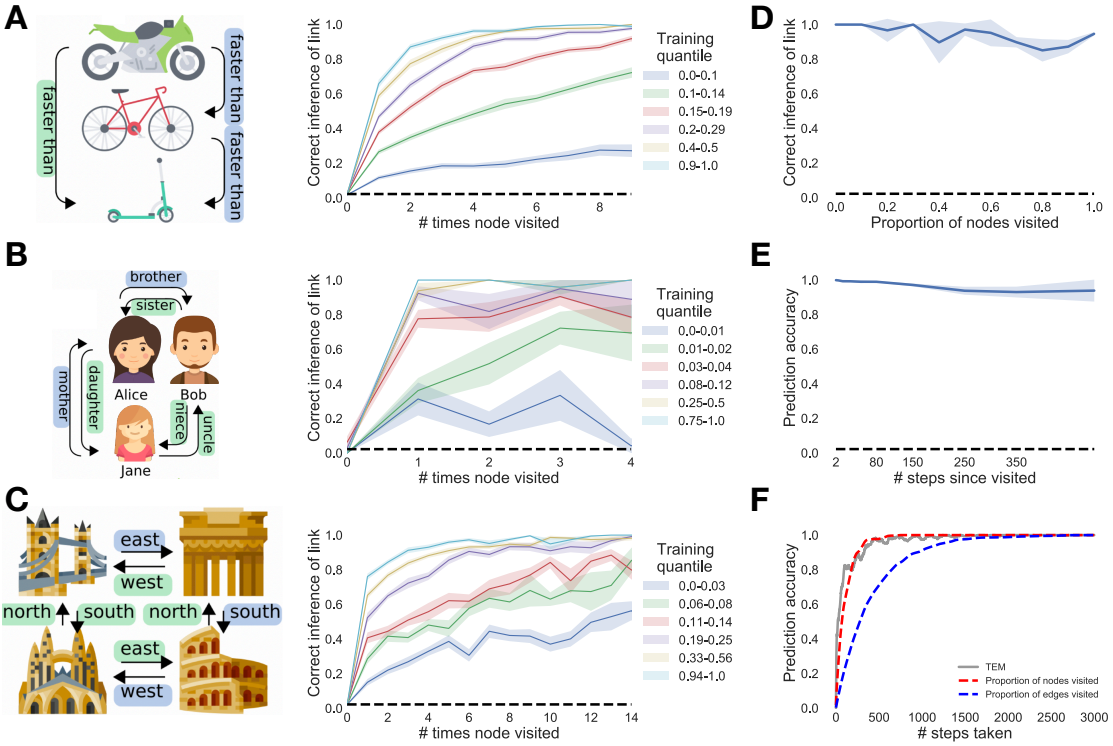

Figure 7.5: TEM learns and generalises abstract relational knowledge. A-C) Learning to learn: When TEM has only seen a few environments (blue/green) it takes many visits to each node for it to be remembered - this is because it 1) does not understand the structure the graph and 2) it has not learned to how utilise memories. After visiting more environments and discovering the common structure (cyan/yellow) TEM is able to correctly predict a node on the second visit regardless of the edge taken - TEM now understands both the rules of the graph and how to store and retrieve memories. **A)** Transitive inference graphs, **B)** social hierarchy graphs and **C)** 2D spatial graphs. **D-F)** On 2D graphs. **D)** Here we consider how TEM performs on zero-shot link inference: TEM is asked to predict sensory observations when returning to a node via a *new* direction - this is only possible with learned structural knowledge. TEM is able to do this. **E)** TEM is able to store memories for long periods of time, even though it is only trained on sequences of length 25. **F)** TEM's performance tracks nodes visited not edges. All these results demonstrate that TEM has learned and generalised abstract structural knowledge.

grid cells is preserved - that is grid cells (in the same module) that fire next to each other in one environment, do so in all environments. This generalisation is agnostic to environment size, thus TEM has not just learned to represent a single environment size but has instead learned a general representation of 2D topology. These preserved properties provide the substrate for generalisation of relational structure, and are also observed in rodent grid cell populations recorded in multiple environments (Fyhn et al., 2007; Yoon et al., 2013).

7.2.5 TEM forms memories with place cell representations that remap across environments

In TEM, each 'hippocampal' cell, $\mathbf{p}$, is a conjunction between a TEM structural 'entorhinal' cells, $\mathbf{g}$, and sensory input, $\mathbf{x}$; they will only be active when both the structural cells and sensory input are both active (Figure 8.3A). In diffusive worlds, TEM learns sparse representations that resemble hippocampal place cells (Figure 7.6C). These place-like fields span multiple sizes, mirroring the hierarchical composition of hippocampal place fields (Jung et al., 1994; Kjelstrup et al., 2008). Further cell representations can be found in appendix C.

Importantly TEM's 'hippocampal' cells, unlike their 'entorhinal' counterparts, do not generalise. Although each environment shares the same structure, the sensory objects have a different distribution. The conjunctive nature of the hippocampal representation means that TEM's hippocampal cells *do not fully* preserve their correlation structure across environments (Figure 7.6F) but instead relocate apparently at random in different environments. This phenomenon is commonly observed in rodent hippocampal cells and is termed *global remapping* (Bostock et al., 1991; Anderson and Jeffery, 2003).

7.2.6 Diverse Entorhinal and hippocampal cell types form a basis for transition statistics

Animals do not move by diffusion (Purcell, 1977). We next examined representations learned by TEM when trained with different behavioural paradigms (Figure 7.7, see chapter 8 for full experimental details). For non-diffusive transitions, we mimic animals that prefer to spend time near boundaries, and approach objects. Here, because the transition statistics change, so do the optimal representations for predicting future location. Indeed, training TEM on these *behavioural* transition statistics leads to the emergence of new cellular representations that are also found in rodents. Now entorhinal representations, $\mathbf{g}$, in TEM include border cells (Solstad et al. (2008); Figure 7.7C) and cells that fire at the same distance and angle from any object (object vector cells; Høydal et al. (2019); Figure 7.7A) for the two cases

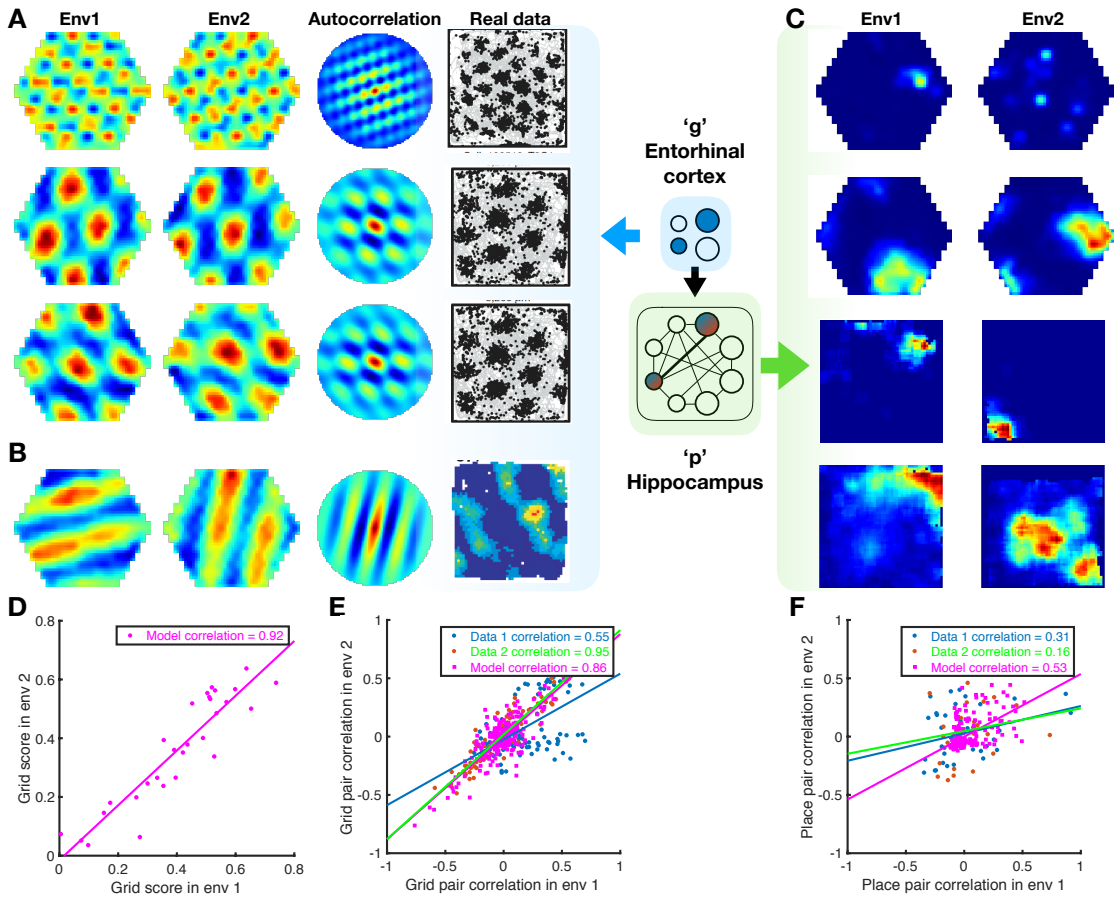

Figure 7.6: TEM learns structural grid cells that generalise and conjunctive memory place cells that remap. A-B) TEM learned structural representations for random walks on 2D graphs (left to right: environments 1 and 2, autocorrelation, real data from (Stensola et al., 2012; Krupic et al., 2012), different cells shown vertically). **A)** TEM learns grid-like cells, of different frequencies (top vs middle), and of different phases (middle vs bottom). **B)** Band like cells are also learned by TEM. Importantly these TEM structural representations A-B) generalise across the different environments 1 and 2. **C)** Learned memory representations resemble place cells (left/right: environments 1 and 2; top 2 simulated, bottom 2 real cells) and have different field sizes. These cells remap between environments (left/right), i.e. do not generalise. **D)** Grid scores of TEM grid-like cells correlate across environments. **E-F)** To examine whether relationships between cells are preserved between environments, we correlated the spatial correlation coefficients of pairs of grid or place fields from each environment, using data from Barry et al. (2012), and Chen et al. (2018b). **E)** The spatial correlation coefficients of pairs of TEM structural cells and real data grid cells correlate strongly. **F)** TEM hippocampal and real data place cells preserved their relationship to a lesser extent. This suggests that TEM structural cells, along with real grid cells, encode generalisable relationships to a greater extent that TEM hippocampal and real place cells.

respectively. This is easily understood: In order to make next-state predictions TEM learns predictive representations, with object vector cells predicting the next

transition is towards the object - as is behaviourally the case.

Critically, these TEM entorhinal cells also generalise across environments, with TEM object vector cells generalising to all objects both within and across environments. The cells do not represent the objects themselves, but rather their predictions about transitions, and they do so in a way that generalises, allowing immediate inferences in new environments. Notably, these same properties are observed in object vector cells in rodent entorhinal cortex (Høydal et al. (2019) ;Figure 7.7A).

Similar cells exist in TEM's hippocampal layer, **p**, with a crucial difference. Here, object sensitive cells represent the vector to a particular object-type but do not generalise across objects (Figure 7.7B) - they represent the conjunction between the structural representation and the sensory data. These cells are reminiscent of 'landmark' cells that have been recorded in rodent hippocampus (but not entorhinal cortex) (Deshmukh and Knierim, 2013).

Objects occur at random locations within each environment, thus when representing the transition statistics of the environment, TEM's entorhinal layer **g** is able to arbitrarily *compose* object vector cell representations (at any location) along with grid and other entorhinal representations. These results suggest that the 'zoo' of different cell types found in entorhinal cortex may be viewed under a unified framework - summarising the common statistics of tasks into basis functions that can be flexibly *composed* depending on the particular structural constraints of the environment the animal/agent faces.

7.2.7 A mechanistic understanding of complex non-spatial abstractions.

While cellular responses are well understood in rodent open field tasks, we have little knowledge of how they combine to control behaviour in more complex problems. Because TEM can learn arbitrary structural abstractions, it can also account formally for hippocampal and entorhinal responses in complex non-spatial tasks.

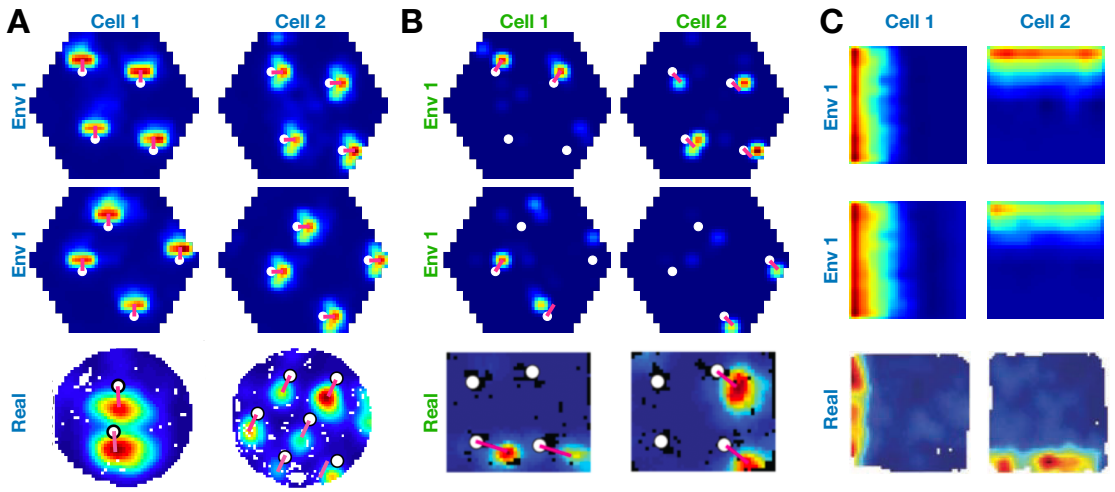

Figure 7.7: TEM learned representations reflect transition statistics. When the agent's transition statistics mimic different behaviours TEM learns different representations (left to right: different cells, top to bottom: environments 1 and 2 and real data). **A)** When biased to move towards objects (white dots) TEM learns structural cells that are active at a certain distance and orientation from the object - object vector cells (Høydal et al., 2019). These cells generalise to *all* objects. **B)** TEM hippocampal cells also reflect this behavioural transition change with similar cells. However, these TEM hippocampal cells do not generalise to all objects - landmark cells (Deshmukh and Knierim, 2013). **C)** When biased to stay near boundaries, TEM is more likely to learn border cell like representations.

To illustrate this, we consider a recent finding by Sun et al. (Sun et al., 2019). Rodents perform laps of a circular track, but only receive reward every four laps. Now hippocampal cells develop a new representation. Whilst some cells represent location on the track, (i.e. place cells; Figure 7.8A top), others are also spatially selective, but fire only on one of the 4 laps (Figure 7.8A middle). A third set fire again at a set spatial location, but vary their firing continuously as a function of lap number (Figure 7.8A bottom). Hippocampal cells maintain a complex combined representation of space and lap number.

When TEM was trained on this task (using 3 laps instead of 4 for computational reasons), it learned these same 3 representations in hippocampus (Figure 7.8B, see appendix C for further cells). Here 'reward' was just a particular sensory event that repeated every 3 trials. As in the biological data, some TEM hippocampal cells encode location on every lap. These cells allow TEM to predict the sensory events that are unchanged between laps. However, as in recorded data, some TEM

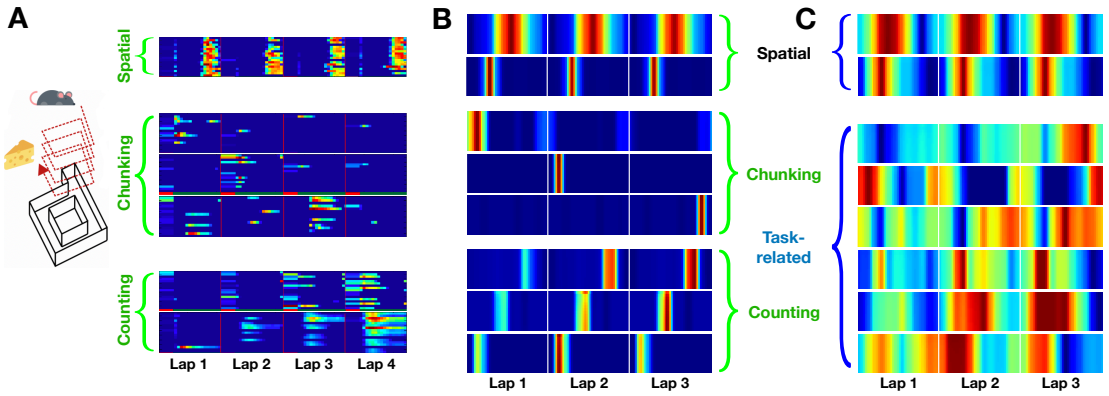

Figure 7.8: TEM representations complex tasks. A) In Sun et al. (2019), rodent performs laps of a track, only 'rewarded' every 4 laps. Different hippocampal cell types are found: spatial place like cells (top), those that preferentially fire on a given lap (middle) and those that count laps (bottom). **B)** When TEM performs a similar task, only 'rewarded' every 3 laps (see appendix C), similar representations are learned. **C)** TEM entorhinal cells not only learn spatially periodic cells (top), but also cells that represent the complex non-spatial task structure of 'every 3 laps' (bottom). Cells of this type are yet to have experimentally observed, but are predicted by TEM. Icons made by FreePik (mouse) and Smashicons (cheese) from flaticons.com.

cells encode location on of the three laps, and some with a lap gradient. These cells allow TEM to represent its position within the 3-lap cycle.

Importantly, TEM allows us to reveal a candidate mechanism. TEM's medial entorhinal cells have reconfigured to code differently for each lap (Figure 7.8C top), understanding that the abstract task space is not a single lap but three (Figure 7.8C bottom). This mechanism is consistent with empirical data as manipulation of entorhinal cortex removes lap-sensitive hippocampal cells (Sun et al., 2019). However, TEM's entorhinal representations stand as a prediction as Sun et al. did not record entorhinal cells. Further cell representations can be found in appendix C.

These results suggest entorhinal cells can learn to represent tasks at multiple levels of cognitive abstraction simultaneously, with hippocampal cells reflecting their conjunction with sensory experience. More broadly, TEM predicts hippocampal cells will reflect a combination of the necessity to predict the current sensory input and the necessity to separate states that predict different distant futures. They will therefore contain representations of the current location in space, but also the current location in the task.

7.3 Discussion

Building an understanding that spans from computation through cellular activity to behaviour is a central goal of neuroscience. One field that promises such an understanding is spatial navigation and the hippocampus. However, whilst cells are precisely described for open field foraging in small arenas, it has been unclear how these responses could generalise to real-world behaviours. Similarly, it has been unclear how to understand these spatial responses in the context of hippocampal involvement in memory broadly writ (Scoville and Milner, 1957) and relational memory in particular (Eichenbaum and Cohen, 2014). In this work, we have shown that by formalising the problem of relational abstraction, using factorisation and conjunction of representations, it is possible to account for spatial inferences as a special case of relational memory as hypothesized by Eichenbaum and colleagues

In doing so, we provided unifying principles that account for a number of seemingly disparate phenomena. For example, grid cells, band cells, border cells, object vector cells all emerge from bases describing likely transitions. We show that this structural basis is also important for understanding several seemingly unrelated processes such as hippocampal remapping (Bostock et al., 1991; Anderson and Jeffery, 2003), and transitive inference (Bunsey and Eichenbaum, 1996) which are shown to be two sides of the same coin - the structural knowledge transferred during remapping supports the generalisation of transitive structure. Whilst the idea that hippocampal memories are summarised in cortex is influential (McClelland et al., 1995), TEM therefore also suggests how cortical representations feed back onto the hippocampus to organise new experience and memory.

Spatial reasoning not only provides a particularly clean example of the factorisation of relational and sensory knowledge, but also affords generalisation as relational meaning repeats regularly across space (O'Keefe and Nadel, 1978). However, by considering the relational memory problem more broadly, we have shown that cellular responses can be formally accounted for in situations more complex than open-field foraging. Whilst we have so far considered simple behaviours (e.g. running 4 laps of a maze to attain reward), we envisage that, together with exciting insights from

reinforcement learning (Stachenfeld et al., 2017), this and similar frameworks may be useful in extending our computational understanding from open field navigation towards Tolman's original ideas of a systematic organisation of knowledge spanning all domains of behaviour (Tolman, 1948).

RIP Howard Eichenbaum (1947-2017).

8

The Tolman-Eichenbaum Machine for machine learners

Contents

Here we follow on from chapter 7, where a more intuitive and neuroscience focused approach was taken, and give a complete description of the Tolman-Eichenbaum Machine. For an introduction and grounding of the problem, please refer back to chapter 7.

8.1 A task for structure generalisation

We wish to formalise a task type that not only relates to known hippocampal function, but also tests the learning and generalising of abstract structural knowledge. This would then offer a single common framework (generalising structural knowledge) that explains many hippocampal functions, and as it turns out, explains many known cell representations.

We formalise this via relational understanding on graph structures (a graph is a set of nodes that relate to each other). We demonstrate that two seemingly distinct hippocampal functions - spatial navigation and relational memory tasks - can be viewed in this common framework.

Should one passively move on a graph (e.g. 8.1), where each node is associated with a non-unique sensory observation (e.g. an image of a banana), then predicting the subsequent sensory observation tests whether you understand the graph structure you are in. For example, if you return to a previously visited node (Figure 8.1 pink) by a new direction - it is only possible to predict correctly if you know that a $right \rightarrow down \rightarrow left \rightarrow up$ means you're back in the same place. Knowledge of such loop closures is equivalent to understanding the structure of the graph.

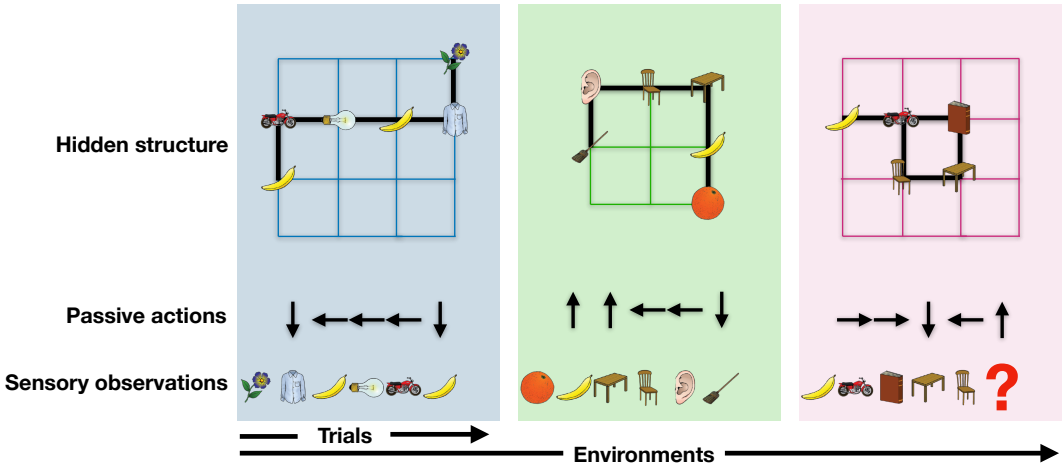

Figure 8.1: Task schematics for TEM. Learning to predict the next sensory observation in environments that share the same structure but differ in their sensory observations. TEM only sees the sensory observations and associated action taken, it is not told about the underlying structure - this must be learned.

We thus train our model (TEM) on these graphs with it trying to predict the next sensory observation. TEM is trained on many environments sharing the same structure, e.g. 2D graphs (Figure 8.1), however the stimulus distribution is different (each vertex is randomly assigned a stimulus). Should it be able to learn and generalise this structural knowledge, then it should be able to enter new environments (structurally similar but with different stimulus distributions) and perform feats of loop closure on first presentation.

We note that this feat of loop closure on first presentation has an intuitive meaning for space, but it is identical to the first presentation inferences made on tasks of transitive inference (Bunsey and Eichenbaum, 1996) and social hierarchy (Kumaran et al., 2012) - tasks that the hippocampus plays a crucial role in.

In order to show that under a single framework (TEM), many aspects of hippocampus and entorhinal cortex can be explained, we thus choose graphs structures that reflect the types of tasks (space, transitive inference, social hierarchies) under which these brain areas have been studied. We describe the details of each task, along with details of simulations in Section C.1.

8.2 The Tolman-Eichenbaum Machine

In the following description, we try to repeat the same information at successively increasing levels of detail. We hope this will allow readers to build an understanding of the model at their preferred level.

8.2.1 Problem statement - Intuitive

We are faced with the problem of predicting sensory observations that come from probabilistic transitions on a graph. The training data is a continuous stream of sensory observations and actions/relations (Figure 8.1). For example, the network will see "banana, north, tree, east, book, south, door ..." or "Joe, parent, Denise, niece, Anna, sibling, Fred ...". The model should predict the next sensory observation with high probability.

8.2.2 Problem statement - Formal

Given data of the form $\mathcal{D} = \{(\mathbf{x}^k_{\leq T}, \mathbf{a}^k_{\leq T})\}$ with $k \in \{1, \cdots, N\}$ (which environment it is in), where $\mathbf{x}_{\leq T}$ and $\mathbf{a}_{\leq T}$ are a sequence of sensory observations and associated actions/relations (Figure 8.1), N is the number of environments in the dataset, and T is the duration of time-steps in each environment, our model should maximise its probability of observing the sensory observations for each environment, $\mathrm{p}_\theta(\mathbf{x}_{\leq T})$, where θ are the model parameters.

8.2.3 High level model description

We choose to model our problem probabilistically using a generative model - this allows us to offer a normative model for how the observed data depends on unobserved latent variables e.g. seeing a banana depends on where you are, but "where you are" is a latent variable - it is never observed. One can then in principle use Bayes' theorem to invert this generative model and provide the optimal posterior distribution over the latent variables given the observations ("inference"). However, in most scenarios, including ours, this inversion is computationally impractical. Many methods have been proposed to **approximate** this inference. One particularly powerful method is to learn the parameters of an "inference" model. Once trained, this model will approximately invert the generative model and perform the inference, mapping the observed data to the latent (unobserved) variables. This idea was introduced in the Wake-sleep algorithm (Hinton et al., 1995) and the Helmholtz machine (Dayan et al., 1995), and has since been adopted by Variational Autoencoders (Kingma and Welling, 2013; Rezende et al., 2014).

In common instantiations of generative models, latent variables are the "causes" of, for example, pixels in stationary images. Here, we provide a generative model where latent variables are positions that result from taking relational actions in a cognitive map. We further enable generalisation of knowledge across domains by separating latent variables of location that generalise across maps, $\mathbf{g}$, from those that are 'grounded' in sensory experience and therefore specific to a particular

map $\mathbf{p}$. 'Grounded locations', $\mathbf{p}$ encode [abstract location, sensory experience] conjunctions for the current environment.

The model aims to predict the next sensory experience from all previous sensory experiences. This problem is not inherently *Markovian*. The next sensory experience can depend on historic experiences independently from the most recent experience (old locations might be encountered on meandering paths). However, the problem can be rendered Markovian by the inclusion of a Memory M that remembers what experience is where in the current environment. The inclusion of "grounded location" variables, $\mathbf{p}$, means that this memory simply has to remember these variables.

We give the full generative model for the general probabilistic case with noise in both the action and sensory inputs and derive the appropriate loss functions. However, in this manuscript we only consider tasks where there is no noise in these inputs. We therefore implement a version of the model that ignores noise (discussed in Section 8.3) - this leads to faster training and more accurate inference in the noise-free case.

8.2.4 TEM and the brain

We propose TEM's abstract location representations ($\mathbf{g}$) as medial entorhinal cells, TEM's grounded locations ($\mathbf{p}$) as hippocampal cells, and TEM's sensory input $\mathbf{x}$ as lateral entorhinal cells. In other words, TEM's sensory data (the experience of a state) comes from the 'what stream' via lateral entorhinal cortex, and TEM's abstract location representations are the 'where stream' coming from medial entorhinal cortex. TEM's (hippocampal) conjunctive memory links 'what' to 'where', such that when we revisit 'where' we remember 'what'.

TEM's medial entorhinal representation, ($\mathbf{g}$), invites comparison to continuous attractor neural networks (CANNs) (Zhang, 1996), commonly used to model grid cell activity in spatial tasks (Fuhs and Touretzky, 2006; Guanella and Verschure, 2006; Burak and Fiete, 2009). Like CANNs, TEM's $\mathbf{g}$ layer is a recurrent neural network. Like CANNs, different recurrent weights mediate the effects of different actions/relations. Unlike CANNs, however, our weights are not hardcoded, but

learnt from experience. Furthermore, due to the factorisation afforded by $\mathbf{p}$, they can be learnt directly from sensory experience without any "location" input. They can therefore learn abstract map-like representations not only in spatial problems, but also in arbitrary non-spatial problems - even those in which it would be difficult for humans to hand code an effective "location" representation (such as a family tree).

TEM's grounded locations ($\mathbf{p}$) resemble hippocampal cells, encoding location-sensory conjunctions (Wood et al., 1999; Komorowski et al., 2009; Chen et al., 2019) and enabling fast episodic memories (Bostock et al., 1991; Wills et al., 2005; Nakazawa et al., 2002).

TEM's sensory representations ($\mathbf{x}$) resemble lateral entorhinal representations, encoding processed sensory input (here - objects) (Neunuebel et al., 2013; Deshmukh and Knierim, 2011; Manns and Eichenbaum, 2006). Notably, TEM learns most effectively if sensory representations are passed through an approximate Laplace Transform as is reported in lateral entorhinal cells (Tsao et al., 2018) (see section 8.2.8 and Section 8.2.10).

TEM describes the hippocampal-entorhinal system as one that performs inference; TEM medial entorhinal cells infer a location in abstract space based based on their previous belief of location (and optionally sensory information linked to previous locations via memory). TEM hippocampal cells infer the current memory representation based on a conjunction between the sensory data and believed location in abstract space.

Though we already refer to these variables as entorhinal and hippocampal cells, we reiterate that no representations are hard-coded - *all TEM representations are learned.*

8.2.5 High-level algorithmic description

.

We now describe the fundamentals behind the Tolman-Eichenbaum Machine (TEM). TEM sees a stream of sensory observations and actions ($\mathbf{x}$ and $\mathbf{a}$). It's objective is to predict the next sensory input. If these observations are arranged

on a graph with any regularities, TEM can profit from these regularities to predict the sensory consequences of edges it has never previously taken. After learning these regularities, TEM can transfer them to new environments that have the same regularities, but different sensory observations.

Principles

TEM relies on two simple principles / components. Firstly a map-like component that learns about the abstract structure shared across environments (Tolman), and secondly a conjunctive memory component that grounds this learned abstract structure to the current environment (Eichenbaum). We denote the map-like variables as $\mathbf{g}$, and the grounded conjunctive variables as $\mathbf{p}$.

Each grounded location $\mathbf{p}$ is a conjunction, tying and abstract location $\mathbf{g}$ to a sensory experience $\mathbf{x}$. Each abstract location, however, has the potential to instantiate many different grounded locations - one for each for possible sensory experience. An attractor network memory learns, after a single experience, which location-experience pair is valid in the current world. The opposite is also true - a sensory experience can re-instantiate the memory of a grounded location i.e. the conjunctive memory process allows both abstract location to predict sensory experience, and sensory experience to predict abstract location.

Naturally, TEM can only predict a sensory observation should it have seen it before and formed a memory of its grounded location. TEM re-instantiates memories of grounded locations via indexing from its abstract location, $\mathbf{g}$, and so re-instantiating the correct grounded location requires TEM to index using the same abstract location code as when the memory of grounded location was formed.

This puts strong constraints on the types of representations TEM must learn. Firstly, it must learn a structural map-like code that transferably path-integrates such that $\mathbf{g}$ is the same when returning to a state (so the correct memory is indexed). Secondly it must learn representations $\mathbf{g}$ that are different for different states - so that each state can have a separate memory attached to it. These

two constraints are fundamental to TEM representations, and are shown to be satisfied by grid-cell and other entorhinal codes.

Generative model

The generative model (Figure 8.2 left) sees an action $\mathbf{a}$, combines this with its previous $\mathbf{g}$ to predict the next abstract location in its cognitive map $\mathbf{g}$ which then proposes candidate grounded locations. An attractor network pattern completes these candidates, suppressing those that have not been experienced before, to restore a memory of the appropriate grounded location $\mathbf{p}$. The restored memory/grounded location then predicts sensory observation $\mathbf{x}$. We show this schematically in Figure 8.3B

Inference Model

The inference model (Figure 8.2 right) sees a sensory observation $\mathbf{x}$, retrieves a memory of grounded location best related to this sensory observation, then infers the next $\mathbf{g}$ from both the previous $\mathbf{g}$ (and action $\mathbf{a}$) and this memory of grounded location. $\mathbf{p}$ is then re-inferred using the best estimate of $\mathbf{g}$ and the new $\mathbf{x}$ (Figure 8.3A). This new grounded location $\mathbf{p}$ is used to update memory weights, M.

Training

Both the generative and inference models have weights that must be learnt. The objective of training is for the generative model to predict the sensory input, $\mathbf{x}$, and for the inference model to infer the generative model's latent variables, [$\mathbf{p}$, $\mathbf{g}$], from the sensory input. The resulting training algorithm involves an interplay between generative and inference models (Figure 7.4), in which the generative model takes the current state of the inference model and (from this) predicts its next state (including the next sensory data). This leads to errors between the predicted and inferred/observed variables at each level [$\mathbf{p}$, $\mathbf{g}$, $\mathbf{x}$]. The weights in both networks are adjusted along a gradient that reduces these errors using backpropagation through time.

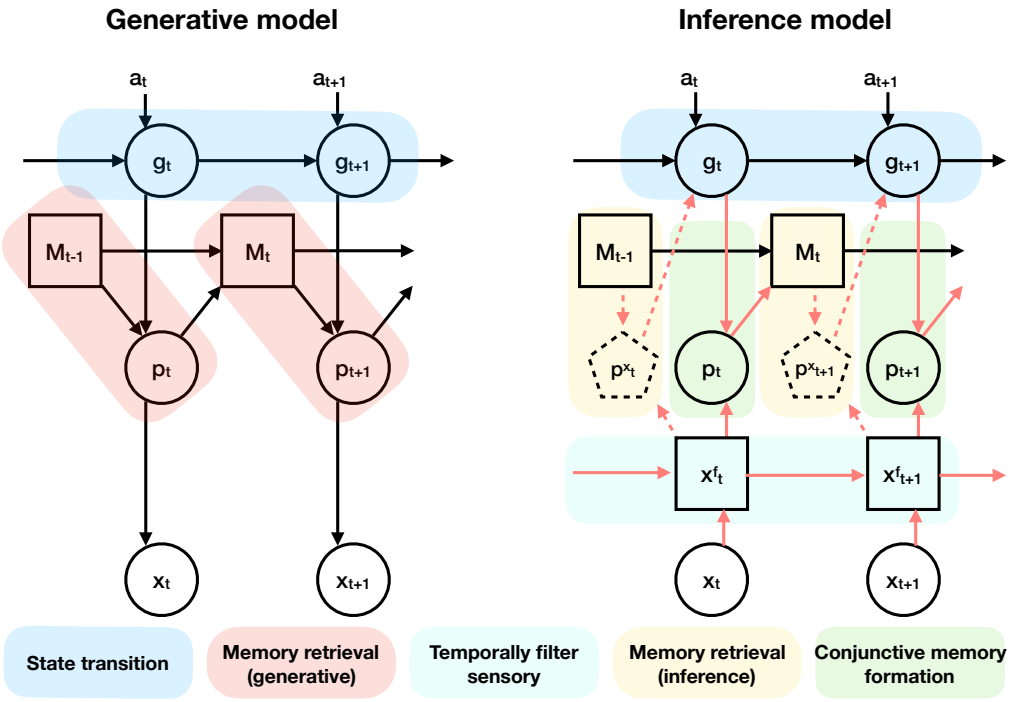

Figure 8.2: TEM generative and inference model. Circled/ boxed variables are stochastic/ deterministic. Red arrows indicate additional inference dependencies. Dashed arrows/boxes are optional as explained in the text.

The model is trained in multiple different environments, differing in size and sensory experience. When entering a new environment, network weights are retained but Hebbian weights M are reset. The most important weights are those that transition **g** as they encode the structure of the map. They must ensure (1) that each location in the map has a different **g** representation (so a unique memory can be built), (2) that arriving at the same location after different actions causes the same **g** representation (so the same memory can be retrieved) - a form of path integration for arbitrary graph structures. For example, the relation "uncle" must cause the same change in **g** as father followed by brother, but different from brother followed by father ('non-associative' relationships, not just associative ones seen in 2d graphs). These transition weights are shared between generative and inference models, though other weights are not. Shared weights are atypical for variational autoencoders, but are important for biological considerations. At each step we compared what was inferred to what was predicted from the inferred at the previous time-step - we add up the losses for a sequence and then update the weights.

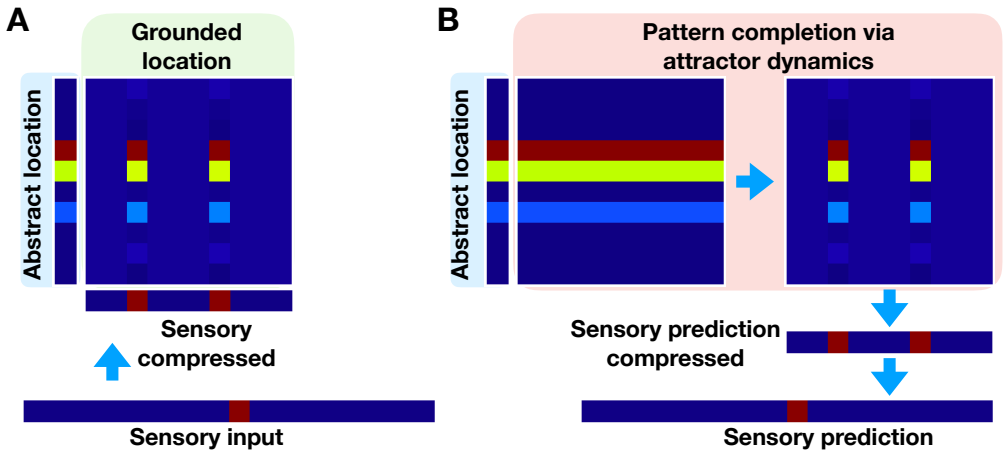

Figure 8.3: Schematic of conjunction and pattern completion. **A)** Schematic of conjunction: i.e. how grounded location (hippocampal) cell representations are formed as a conjunction between abstract location (medial entorhinal) and sensory experience (lateral entorhinal). Note the initial one-hot sensory representation is compressed to a two-hot representation. **B)** From just abstract location input alone, the correct grounded location representation can be retrieved via attractor dynamics - exciting co-active cells and inhibiting those that were not co-active. The retrieved grounded location can then predict the sensory experience (after summing over the entorhinal preference dimension and then being decompressed). This procedure can also be done 'the other way round' with sensory experience alone, in which case the abstract location is predicted (used in the inference model).

Hierarchies in the map

When representing tasks that have self repeating structure (as ours do), it becomes efficient to hierarchically organise your cognitive map. In this spirit, we separate our model into multiple parallel streams, each as described above, where each stream can learn weights to represent the world at different scales. These scales are then only combined when retrieving memories/ grounded locations in the attractor network. We provide further details on this is Section 8.2.10.

Model flow summary

The inference model is the one that sees sensory data $\mathbf{x}_t$ at each time-step t. It is 'awake' and transitions through time on its own inferring $\mathbf{g}_t$ and $\mathbf{p}_t$ at each time-step. The inference model infers the new abstract location $\mathbf{g}_t$ before inferring the new grounded location $\mathbf{p}_t$. In other words latent variables $\mathbf{g}$ and $\mathbf{p}$ are inferred

in the following order $\mathbf{g}_t$, $\mathbf{p}_t$, $\mathbf{g}_{t+1}$, $\mathbf{p}_{t+1}$, $\mathbf{g}_{t+2}\cdots$. This flow of information is shown in a schematic in Figure 8.2.

Independently, at each time-step, the generative model asks 'are the inferred variables from the inference model what I would have predicted given my current understanding of the world (weights) '. I.e. 1) Is the inferred $\mathbf{g}_t$ the one I would have predicted from $\mathbf{g}_{t-1}$. 2) Is the inferred $\mathbf{p}_t$ the one I would have predicted from $\mathbf{g}_t$. 3) Is $\mathbf{x}_t$ what I would have predicted from $\mathbf{p}_t$. This leads to errors (at each timestep) between inferred and generative variables $\mathbf{g}_t$ and $\mathbf{p}_t$, and between sensory data $\mathbf{x}_t$ and its prediction from the generative model.

At then end of a sequence, these errors are accumulated, with both inference and generative models updating their parameters along the gradient that matches each others variables and also matches the data.

Since the inference model runs along uninterrupted, it's activity at one time-step influence those at later time-steps. Thus when learning (using back-propagation through time - BPTT), gradient information flows backwards in time. This is important as, should a bad memory be formed at one-time step, it will have consequences for later predictions - thus BPTT allows us to learn how to form memories and latent representations such that they will be useful many steps into the future.

8.2.6 Detailed algorithmic description

8.2.7 Generative architecture

TEM has a generative model (Figure 8.2A) which factorises as

$$\mathrm{p}_\theta\left(\mathbf{x}_{\leq T}, \mathbf{p}_{\leq T}, \mathbf{g}_{\leq T}\right) = \prod_{t=1}^{T} \mathrm{p}_\theta\left(\mathbf{x}_t \mid \mathbf{p}_t\right) \mathrm{p}_\theta\left(\mathbf{p}_t \mid \mathrm{M}_{t-1}, \mathbf{g}_t\right) \mathrm{p}_\theta\left(\mathbf{g}_t \mid \mathbf{g}_{t-1}, \mathbf{a}_t\right) \tag{8.1}$$

M_{t-1} represents the agent's memory composed from past hippocampal representations $\mathbf{p}_{<t}$. θ are parameters of the generative model. The initial $\mathrm{p}_\theta\left(\mathbf{g}_1 \mid \mathbf{g}_0, \mathbf{a}_1\right) = \mathrm{p}_\theta\left(\mathbf{g}_1\right)$, i.e not conditioned on any prior variables. The model can be described by a sequence of computations represented by the following equations:

Transition sample	$\mathbf{g}_t \sim \mathcal{N}\left(\cdot \mid \mu = D_a\,\mathbf{g}_{t-1}, \sigma = f_{\sigma_g}(\mathbf{g}_{t-1})\right)$
Retrieve memory	$\mathbf{p}_t \sim \mathcal{N}\left(\mu = attractor(\mathbf{g}_t, \mathrm{M}_{t-1}), \sigma = f(\mu)\right)$
Observation sample	$\mathbf{x}_t \sim Cat\left(f_x(\mathbf{p}_t)\right)$
Repeat process for the next timestep	$\rightarrow \mathbf{g}_{t+1} \rightarrow \mathbf{p}_{t+1} \rightarrow \mathbf{x}_{t+1} \cdots$

We pictorially show these processes in Figure 8.4 (just consider the blue 'freq' stream initially, the second red stream will make sense in Section 8.2.10).

To predict where we will be, we can transition from our current location based on our heading direction (i.e. path integration). D_a is a set of learnable weights for each available action.

Once TEM has transitioned, it then retrieves a memory indexed by it's believed location. Memories are retrieved via an attractor network (details in a later section). f_{σ_g} is a simple multi layer perceptron (MLP).

After the memory has been retrieved, sensory information is extracted in order to predict the current observation. Our sensory sensory data is represented in a one-hot encoding, e.g. each element in the vector corresponds to a different sensory experience, and so we model it with a categorical distribution Cat. The function $f_x(\cdots)$ is $softmax\left(f_{c*}\left(w_x \sum_j p_{t,s,j} + b_x\right)\right)$, which looks intimidating but in words is simple: we take the retrieved memory, sum over the entorhinal preference cells

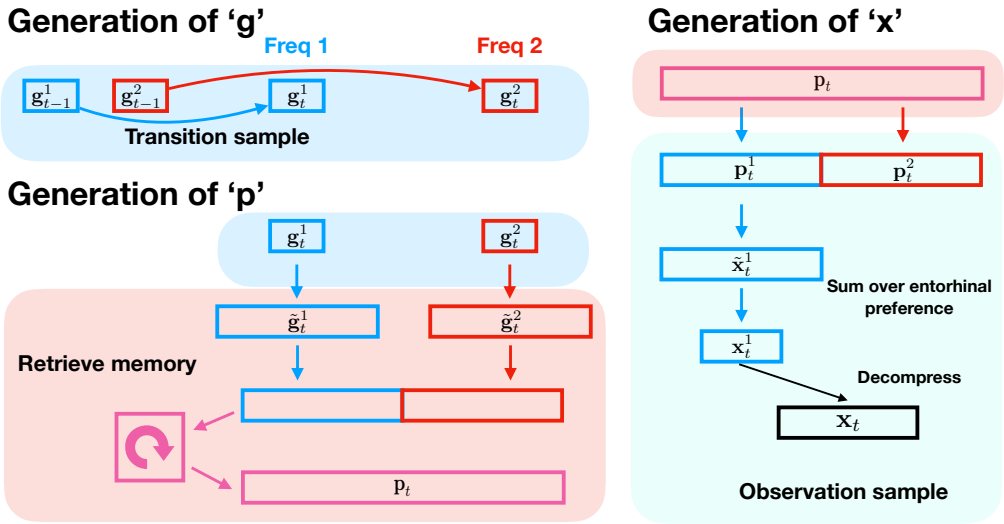

Figure 8.4: Schematic to show the generative process, making explicit the dimension changes e.g. the sensory decompression step and ensuring the entorhinal input to hippocampus has the appropriate dimension. Red/Blue boxes describe two different 'frequencies'.

j to produce a vector of the correct dimension. f_{c^*} is a MLP for 'decompressing' into the correct input dimensions. The 'summing over entorhinal preference j' is shown schematically in Figures 8.3B and 8.4, with subsequent decompression transformation, $f_{c*}(\cdots)$, to a one-hot sensory prediction.

8.2.8 Inference architecture

We have just defined the generative model, however to do anything interesting we need to be able to infer the posterior over the hidden variables. Unfortunately, due to the inclusion of memories, as well as other non-linearities, the posterior $p(\mathbf{g}_t, \mathbf{p}_t \mid \mathbf{x}_{\leq t}, \mathbf{a}_{\leq t})$ is intractable. We therefore turn to approximate inference, and in particular the variational autoencoder framework (Kingma and Welling, 2013; Rezende et al., 2014). Here the inference distribution is parameterized by a neural network, which during training learns how to infer.

The split between inference and generative networks is analogous to the idea of the sleep-wake algorithm. The inference network is 'awake' and observes the world, seeing each state as it transitions through the environment. The generative network is used during 'sleep' for *training* and where it compares 'sleep' generated variables to the inferred 'awake' ones. This allows training of *both* networks such that the inference network and generative network learn to align themselves i.e. the generative network learns to predict both sensory data and the variables inferred by the learned inference network (a.k.a recognition distribution) which, in turn, learns to appropriately map sensory events to latent variables.

In defining our approximate recognition distributions, $\mathrm{q}_\phi(\cdots)$, we make critical decisions that respect our proposal of map-structure information separated from sensory information as well as respecting certain biological considerations. We use a recognition distribution that factorises as

$$\mathrm{q}_\phi\left(\mathbf{g}_{\leq T}, \mathbf{p}_{\leq T} \mid \mathbf{x}_{\leq T}\right) = \prod_{t=1}^{T} \mathrm{q}_\phi\left(\mathbf{g}_t \mid \mathbf{x}_{\leq t}, \mathrm{M}_{t-1}, \mathbf{g}_{t-1}, \mathbf{a}_t\right) \mathrm{q}_\phi\left(\mathbf{p}_t \mid \mathbf{x}_{\leq t}, \mathbf{g}_t\right) \tag{8.2}$$

See Figure 8.2 for inference model schematic. ϕ denote parameters of the inference network. The variational posterior can be expressed by the following equations.

Compress sensory observation	$\mathbf{x}_t^c = f_c(\mathbf{x}_t)$
Temporally filter sensorium	$\mathbf{x}_t = (1-\alpha)\,\mathbf{x}_{t-1} + \alpha\,\mathbf{x}_t^c$
Sensory input to hippocampus	$\tilde{x}_t = W_{tile} w_p f_n(\mathbf{x}_t)$
Retrieve memory	$\mathbf{p}_t^x = attractor(\tilde{\mathbf{x}}_t, \mathrm{M}_{t-1})$
Infer entorhinal[1]	$\mathbf{g}_t \sim \mathrm{q}_\phi\left(\mathbf{g}_t \mid \mathbf{p}_t^x, \mathbf{g}_{t-1}, \mathbf{a}_t\right)$
Entorhinal input to hippocampus	$\tilde{g}_t = W_{repeat} f_g(\mathbf{g})$
Infer hippocampus	$\mathbf{p}_t \sim \mathcal{N}\left(\cdot \mid \mu = f_p(\tilde{\mathbf{g}}_t \cdot \tilde{\mathbf{x}}_t), \sigma = f(f_n(\mathbf{x}_t), \mathbf{g}_t)\right)$
Form memory	$\mathrm{M}_t = hebbian(\mathrm{M}_{t-1}, \mathbf{p}_t)$
Repeat process for next observation	$\rightarrow \mathbf{x}_{t+1} \rightarrow \mathbf{g}_{t+1} \rightarrow \mathbf{p}_{t+1} \cdots$

We pictorially show this process (with no Hebbian memory storage) in Figure 8.5 (just consider the blue stream initially, the second red stream will make sense in Section 8.2.10). We now explain step by step in words, offering further details and hopefully some intuition.

We take our input $\mathbf{x}_t$, which is a one-hot encoding of sensory experience (e.g. each element in the vector corresponds to a different sensory experience), and compress it via $f_c(\mathbf{x}_t)$. We compress from a one-hot to a two-hot encoding to reduce the size of the resulting network and ease computation (shown in Figure 8.3A and Figure 8.5).

We then smooth this compressed representation over time using exponential filtering. We note that although the exponential smoothing appears over-simplified, it approximates the Laplace transform with real coefficients. Cells of this nature have been discovered in LEC (Tsao et al., 2018).

Next, we normalise the representation using $f_n()$ which demeans then applies a rectified linear activation followed by unit normalisation. These representations are then scaled by w_p, and then multiplied by a fixed weight matrix W_{tile} (which gives the appropriate hippocampal dimension - all dimensions shown in Table C.1) to give $\tilde{\mathbf{x}}_t$ which is TEM's sensory input to hippocampus.

[1] $\mathbf{p}_t^x$ is an intermediary variable retrieved via the memory M_{t-1} from $\mathbf{x}_t$ - i.e. it represents $\mathbf{x}_{\leq t}$ and M_{t-1} in the posterior for $\mathbf{g}_t$.

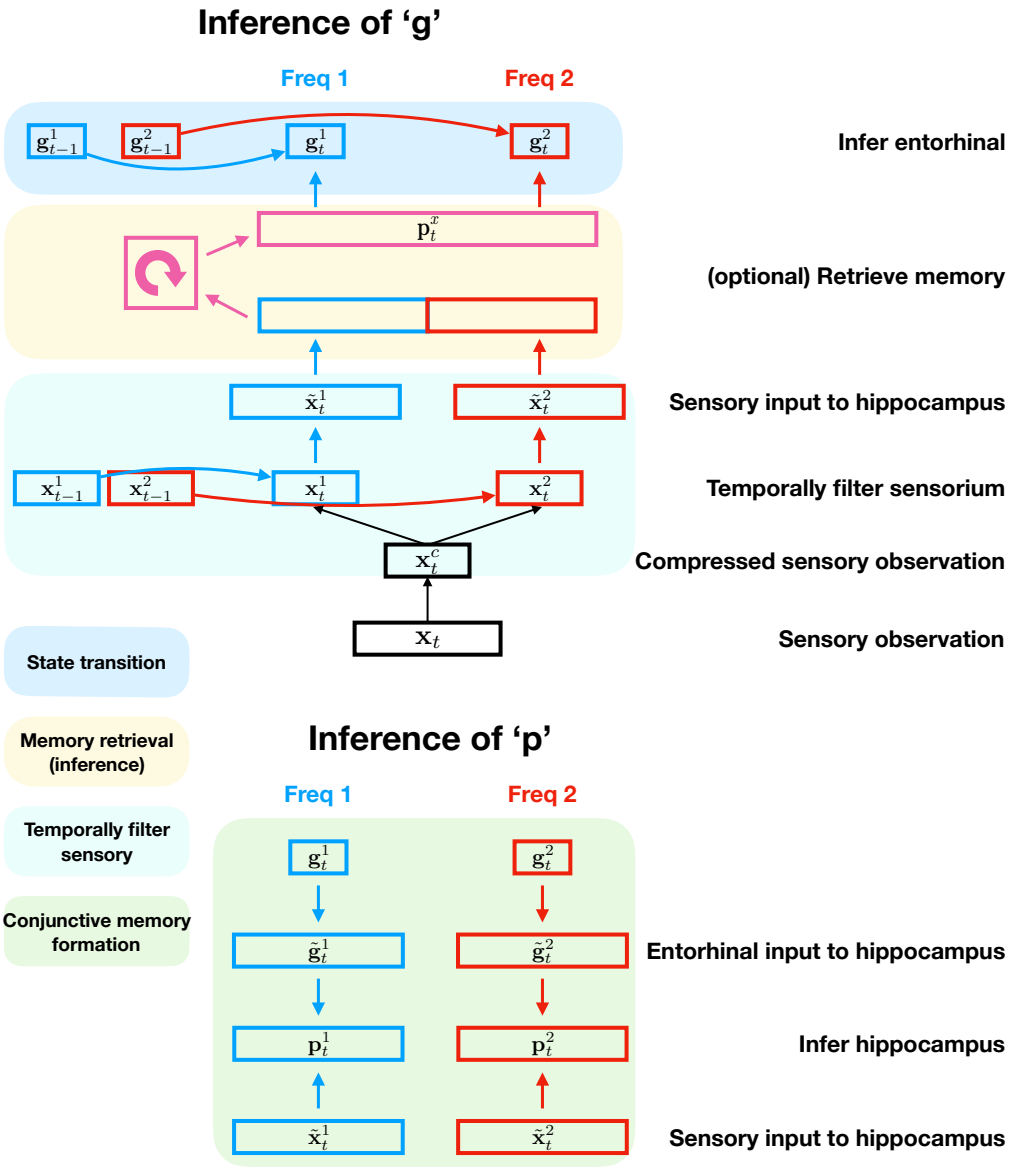

Figure 8.5: Schematic to show the inference process, making explicit the dimension changes e.g. ensuring the entorhinal and sensory input to hippocampus have the same dimension. We note that this could just be done with a simple linear transformation, but for interpretability we chose to repeat/tile the representations. Red/Blue boxes describe two different 'frequencies'.

We are now ready to infer where we are in the graph. We factorise our posterior on $\mathbf{g}$ as

$$\mathrm{q}_\phi\left(\mathbf{g}_t \mid \mathbf{x}_{\leq t}, \mathrm{M}_{t-1}, \mathbf{g}_{t-1}, \mathbf{a}_t\right) = \mathrm{q}_\phi\left(\mathbf{g}_t \mid \mathbf{g}_{t-1}, \mathbf{a}_t\right) \mathrm{q}_\phi\left(\mathbf{g}_t \mid \mathbf{x}_{\leq t}, \mathrm{M}_{t-1}\right) \tag{8.3}$$

To know where we are, we can path integrate (the first distribution, equivalent to the generative distribution described above) as well as use sensory information

that we may have seen previously (second distribution). The second distribution (optional) provides information on location given the sensorium. Since memories link location and sensorium, successfully retrieving a memory given sensory input allows us to refine our location estimate. We use $\tilde{\mathbf{x}}_t$ as the input to the attractor network to retrieve the memory associated with the current sensorium, $\mathbf{p}_t^x$. We use MLPs with input $\mathbf{p}_t^x$ to parameterized the mean and variance of the distribution. This factored distribution is a Gaussian with a precision weighted mean - i.e. we refine our generated location estimate with sensory information.

For Bayesian connoisseurs, we note that, unlike $\mathbf{p}_t$, these retrieved memories $\mathbf{p}_t^x$ are not random variables in the generative model and are therefore not inferred. Instead they are part of the function in the inference model that learns the approximate posterior posterior on $\mathbf{g}_t$. Nevertheless they share similarities to $\mathbf{p}_t$, e.g. they have the same dimension and are pressured to learn similar representations (see Section 8.3). Biologically, they can be thought of as memories cued only by sensory input, and not inferred from the combination of sensory and structural input.

Now that we have inferred where we are, we are ready to form a new memory - infer our hippocampal representation. After the the entorhinal representation is down-sampled using $f_g(\cdots)$, we then multiply by a fixed weight matrix W_{repeat} (which gives the appropriate hippocampal dimension - all dimensions shown in Table C.1) to give $\tilde{\mathbf{g}}_t$. We define the mean of the inferred hippocampal representation as the element wise multiplication of $\tilde{\mathbf{x}}_t$ and $\tilde{\mathbf{g}}_t$ followed by an activation function. We choose the *leaky relu* activation function to create sparsity and ensure the only active hippocampal cells are those that receive both map-structure and sensory information. We note that the two fixed weight matrices are designed such that their application, followed by an element wise product between $\tilde{\mathbf{x}}_t$ and $\tilde{\mathbf{g}}_t$, is equivalent to an outer product followed by reshaping to a vector (Figure 8.3A).

8.2.9 Memories

Storage using Hebbian learning

Memories of hippocampal cell representations are stored in Hebbian weights between hippocampal cells. We choose Hebbian learning, not only for its biological plausibility, but to also allow rapid learning when entering a new environment. We use the following learning rule to update the memory:

$$\mathrm{M}_t = \lambda \mathrm{M}_{t-1} + \eta(\mathbf{p}_t - \hat{\mathbf{p}}_t)(\mathbf{p}_t + \hat{\mathbf{p}}_t)^T \tag{8.4}$$

where $\hat{\mathbf{p}}_t$ represents place cells generated from inferred grid cells. λ and η are the rate of forgetting and remembering respectively. We note than many other types of Hebbian rules also work.

Notably, unlike the generative network, there is no requirement for a memory in the inference network. However, including such a memory allows the network to refine the path integration with landmark information *before* creating its place code and therefore speeds learning dramatically. However, representations in the main paper are observed both in networks that include an inference memory and those that do not.

In networks that do use an inference memory, we can either use the same memory matrix as the generative case (as the brain presumably does), or we can use a separate memory matrix. Best results (and those presented) were when two separate matrices were used. We used the following learning rule for the inference based matrix: $\mathrm{M}^x_t = \lambda \mathrm{M}^x_{t-1} + \eta(\mathbf{p}_t - \mathbf{p}^x_t)(\mathbf{p}_t + \mathbf{p}^x_t)^T$, where $\mathbf{p}^x_t$ is the retrieved memory with the sensorium as input to the attractor.

Retrieval using an attractor network

To retrieve memories, similarly to (Ba et al., 2016), we use an attractor network of the form

$$\mathbf{h}_\tau = f_p\left(\kappa \mathbf{h}_{\tau-1} + \mathrm{M}_{t-1}\,\mathbf{h}_{\tau-1}\right) \tag{8.5}$$

where τ is the iteration of the attractor network and κ is a decay term. The input to the attractor, $\mathbf{h}_0$, is from the grid cells or sensorium ($\tilde{\mathbf{g}}_t$ or $\tilde{\mathbf{x}}_t$) depending on whether memories are being retrieved for generative or inference purposes respectively. The output of the attractor is the retrieved memory.

8.2.10 Details about embedded hierarchy

Though not a requirement, we embed TEM with the notion of hierarchical scales. TEM abstract location and grounded location (memory) representations, $\mathbf{g}_t$ and $\mathbf{p}_t$ respectively, now come in different frequencies (hierarchies) indexed by superscript f - $\mathbf{p}_t^f$ and $\mathbf{g}_t^f$. This allows higher frequency statistics to be reused across lower frequency statistics, improving the speed of learning and reducing the number of weights that need to be learnt.

Implementation wise, this means our inference network has several parallel streams of procedure described above, each indexed by the superscript f. Each stream has its own learnable parameters (e.g. temporal filtering coefficients in the approximate Laplace transform α^f - a smaller α^f means a longer temporal smoothing window). Otherwise the inference procedure is the same. We schematically show an example of two separate streams in Figure 8.5. The place where the separate streams can interact is in the attractor network: when storing Hebbian memories, connections from high to low frequencies are set to zero, so that memories are retrieved hierarchically (low-resolution first). These connections are not set to zero when the memory M is used in inference.

The generative procedure is similar to as described above, except for state transitions, where connections in D_a (the action dependent transitions weight from $\mathbf{g}_{t-1}$ to $\mathbf{g}_t$) are from low frequency to the same or higher frequency only (or alternatively separate transition streams). There may be different numbers of TEM entorhinal cells in each frequency - this number is described by n^f. When making predictions of sensory experience, we only use $\mathbf{p}_t^{f=0}$ i.e. the highest frequency.

The separation into hierarchical scales helps to provide a unique code for each position, even if the same stimulus appears in several locations of one environment,

since the surrounding stimuli, and therefore the lower frequency hippocampal cells, are likely to be different. Since the hippocampal cell representations form memories, one can also utilise the hierarchical scales for memory retrieval.

TEMs hierarchy is consistent with the hierarchical scales observed across both grid cells (Stensola et al., 2012) and place cells (Kjelstrup et al., 2008), and with the entorhinal cortex receiving sensory information in hierarchical temporal scales (Tsao et al., 2018).

8.3 Optimisation

We wish to learn the parameters for both the generative model and inference network, θ and ϕ, by maximising the ELBO, a lower bound on $\ln \mathrm{p}_\theta\left(\mathbf{x}_{\leq T} \mid \mathbf{a}_{\leq T}\right)$. Following Gemici et al. (2017) (See appendix C or Whittington et al. (2018) supplementary material), we obtain a free energy

$$\mathcal{F} = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\mathrm{q}_\phi\left(\mathbf{g}_{<t}, \mathbf{p}_{<t} \mid \mathbf{x}_{<t}\right)} \left[J_t\right] \tag{8.6}$$

where

$$J_t = \mathbb{E}_{\mathrm{q}_\phi(\ldots)}[\ln \mathrm{p}_\theta\left(\mathbf{x}_t \mid \mathbf{p}_t\right) + \ln \frac{\mathrm{p}_\theta\left(\mathbf{p}_t \mid \mathrm{M}_{t-1}, \mathbf{g}_t\right)}{\mathrm{q}_\phi\left(\mathbf{p}_t \mid \mathbf{x}_{\leq t}, \mathbf{g}_t\right)} + \ln \frac{\mathrm{p}_\theta\left(\mathbf{g}_t \mid \mathbf{g}_{t-1}, \mathbf{a}_t\right)}{\mathrm{q}_\phi\left(\mathbf{g}_t \mid \mathbf{x}_{\leq t}, \mathrm{M}_{t-1}, \mathbf{g}_{t-1}, \mathbf{a}_t\right)}] \tag{8.7}$$

as a *per time-step* free energy. We use the variational autoencoder framework (Kingma and Welling, 2013; Rezende et al., 2014) to optimise this generative temporal model.

Up to this point the model definition is probabilistic and capable of a Bayesian treatment of uncertainty. However, in the tasks examined in this manuscript there is no uncertainty, so there is no need for this complexity. Hence, we use a network of identical architecture but only using the means of the above distributions - i.e. not sampling from the distributions. We then use the following surrogate

loss function to mirror the ELBO:

$$L_{total} = \sum_{t=1}^{T} L_{x_t} + L_{g_t} + L_{p_t} \tag{8.8}$$

with L_{x_t} being a cross entropy loss, and L_{p_t} and L_{g_t} are squared error losses between 'inferred' and 'generated' variables - in an equivalent way to the Bayesian energy function.

It is also possible to speed up learning with some augmented losses: As part of L_{x_t} we include losses between sensory experience and 3 different generative model predictions - those generated directly from the inferred $\mathbf{p}_t$, and also those generated ancestrally through the layers of the network i.e. $\mathbf{g}_t \rightarrow \mathbf{p}_t \rightarrow \mathbf{x}_t$ and $\mathbf{g}_{t-1} \rightarrow \mathbf{g}_t \rightarrow \mathbf{p}_t \rightarrow \mathbf{x}_t$. When using memory in inference, L_{p_t} includes a memory loss between the retrieved memory and the inferred $\mathbf{p}_t$.

We use backpropagation through time truncated to 25 steps - this means we roll forwards the inference network for 25 steps, collect the errors and then backpropagate. We then roll forwards the inference network from where we left off etc - i.e. we do not use a sliding window. Longer BPTT lengths are useful as getting an early $\mathbf{p}_t$ wrong will form a bad memory, which then influences the memory retrieval many timesteps into the future. We limit it to 25 for reasons of computational efficiency. We optimise with ADAM (Kingma and Ba, 2014) with a learning rate that is annealed from $1e-3$ to $1e-4$. Initially we down-weight costs not associated with prediction (L_{g_t} and L_{p_t}). We do not train on vertices that the agent has not seen before.

9

Neural predictions from TEM

Contents

The critical assumption that enables TEM's structural inferences is that the hippocampal representations of new events are **not random**. Instead they are constrained by learned structural representations in the entorhinal input. This assumption seems at odds with the commonly held belief that hippocampal place

cells remap randomly between environments. However, the structural representation for space is *periodic*. Thus, place cells can preserve structural information across environments without being spatial neighbours with the same cells in each environment. Instead, individual cells need only to retain their phases with respect to the grid code. Here, structural knowledge is retained but remapping still occurs because place cells might, in a new environment, move to the same phase but with respect to a **different grid peak** (see e.g. Figure 9.1). Together with the different sensory input between environments, this leads to remapping in TEM's conjunctive hippocampal cells.

Is this true in biological remapping? To uncover this, we tested data from two experiments in which both place and grid cells have been recorded whilst rats (Barry et al., 2012) and mice (Chen et al., 2018b) freely forage in multiple environments[1].

9.1 Experimental prediction

Our theoretical framework predicts place cells and grid cells retain their relationships across environments - despite place cell remapping - to allow generalisation of structural knowledge encoded by grid cells. More specifically, our framework predicts the following to be true: 1) As has been previously observed experimentally (Fyhn et al., 2007), our framework predicts that when an animal is moved from one environment to a structurally similar environment but with different sensory experiences, place cells will undergo remapping, and grid cells will realign. 2) As has also been previously observed experimentally (Fyhn et al., 2007), we predict the grid cell correlation structure (i.e. relationships between grid cells) within a module will be preserved across environments. 3) Despite realignment and remapping, we predict that, within a grid module, a given place cell will retain its relationship with a given grid cell across environments. For example, if a given place cell's firing field is in a given grid cell's firing field in one environment, it should remap to a location in a second structurally similar environment that is also in a firing field of that grid cell (Figure 9.1).

[1]This research was carried out in close collaboration with Tim Muller, a fellow DPhil student.

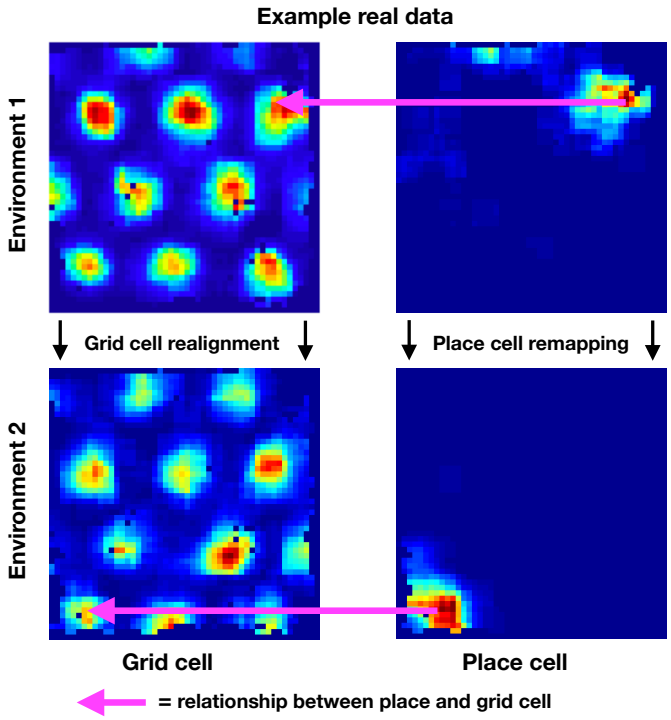

Figure 9.1: Structural knowledge is preserved over apparently random hippocampal remapping. TEM predicts that place cells remap to locations consistent with a grid code. A place cell co-active with a certain grid cell will be more likely to remap to locations that are also co-active with that grid cell.

We empirically test for a preserved place cell-grid cell relationship across environments in two datasets from different remapping experiments, in which both grid and place cells were recorded across different environments. We first briefly describe the experimental setup of the experiments, followed by the details of the analyses and results that support our prediction in both datasets. We additionally demonstrate that these results cannot be explained by the place and grid cells not remapping or realigning, and, as has been previously shown (Fyhn et al., 2007), that the correlation structure of grid cells is preserved across environments.

9.2 Experimental setup

9.2.1 Dataset 1 - Barry et al 2012

In the first dataset (Barry et al., 2012) - dataset 1 - both place and grid cells were recorded from rats in two different environments. The environments were

geometrically identical $1m^2$ arenas that were in distinct locations in the recording room, and differed in their sensory (texture/visual/olfactory) experiences. Each of seven rats had recordings from both environments in MEC and hippocampal CA1. Each recording day consisted of five twenty-minute trials, in which the rat free foraged in the environments. In between trials the rat was taken out of the arena. Of the five trials on a given day, trials 1 and 5 were in one environment, which the animal is familiar with (having spent at least 100 minutes in the environment), and trials 2-4 were exposures to a second, novel environment. We can therefore test for preserved place cell-grid cell relationships both within and across environments in this dataset.

Barry et al. (2012) sought to establish the effects of environmental novelty on grid and place cell properties, finding an increase in grid scale and decrease in grid score, as well as an increase in place cell field sizes in novel environments. This effect reduced with exposure to the novel environment over the course of trials 2-4, such that grid and place cells on trial 4 had properties most comparable to those on trials 1 and 5 (Barry et al., 2012). We therefore restrict our analyses of the second environment to trial 4. Further details about the experimental setup can be found in (Barry et al., 2012).

9.2.2 Dataset 2 - Chen et al 2018

We repeat our analyses in a second dataset (Chen et al., 2018a) - dataset 2. In dataset 2, both place and grid cells were recorded as mice free foraged in both real and virtual reality environments. These real and virtual environments provide the two different environments for the across environment measures of place cell-grid cell relationships. We do not have a 'within environment' condition for this dataset. As described in full in Chen et al. (2018a), in the virtual reality environment the mice were head-constrained such that head movements were constrained to rotations in the horizontal plane while the mouse runs on a Styrofoam ball. Screens and projectors projected a virtual environment around the mouse and onto the floor from a viewpoint that moves with the rotation of the ball. Hence this system allows expression of free foraging spatial navigation behaviour, analogous to that in the real world.

Both the real and virtual reality environments were square, and size $60cm^2$. Trials in the real and virtual environments were 20 and 40 minutes long, respectively. Recordings were made in MEC and hippocampal CA1. Chen et al. (2018a) showed that spatial neuronal cell types that typically characterise 2-dimensional real space, including place cells and grid cells, could be measured in the virtual environment. Of the eleven mice that were trained in the virtual reality system, four had recordings from both place and grid cells, and could therefore be included in our analyses. Further details about the experimental setup and virtual reality system can be found in Chen et al. (2018a).

Details of the number of cells recorded in each animal are found in Table 9.1.

9.3 Data analyses to test for preserved place cell-grid cell relationship

We tested the prediction that a given place cell maintains its relationship with a given grid cell across environments using two measures. First, whether grid cell activity at the position of the peak place cell activity is correlated across environments (gridAtPlace), and second, whether the minimum distance between the peak place cell activity and a peak of grid cell activity is correlated across environments (minDist; normalised to the corresponding grid scale).

9.3.1 Data pre-processing and critical controls

In the tests presented later, we show results for raw data where we take several steps (with different strictness levels) to avoid possible confounds. Results are shown for all combinations of these choices in Table 9.2. These include:

Defining a grid-score cut-off to ensure entorhinal cells were grid cells To ensure we are robust to the quality of grid cells entering the analysis, we consider several different grid score cut-offs. We use cut-offs of 0, 0.3 and 0.8. Using less stringent grid cut offs allows more cells and animals into the analysis (Table 9.1). We would expect our effect to be weaker when reducing the grid score cut off, as the

	Animal	Grid cells	Place cells	Place-grid pairs
Grid score ≥ 0.8				
Dataset 1	1	9	12	108
	6	1	5	5
	7	1	2	2
Dataset 2	1	5	8	40
	2	1	6	6
	4	3	6	18
Grid score ≥ 0.3				
Dataset 1	1	15	12	180
	2	3	10	30
	3	2	6	12
	6	3	5	15
	7	1	2	2
Dataset 2	1	10	8	80
	2	1	6	6
	3	2	4	8
	4	5	6	24
Grid score ≥ 0				
Dataset 1	1	15	12	180
	2	4	10	40
	3	2	6	12
	5	1	4	4
	6	3	5	15
	7	2	2	4
Dataset 2	1	13	8	104
	2	2	6	12
	3	2	4	8
	4	5	6	24

Table 9.1: Numbers of animals and cells that enter the across environments analyses. Table showing the total animal and cell numbers that survive the criteria and make it into the analyses where the grid score cut off is ≥ 0.8, ≥ 0.3 and ≥ 0 for both datasets. Note: although the animal numbers may be the same across the datasets, they are different animals.

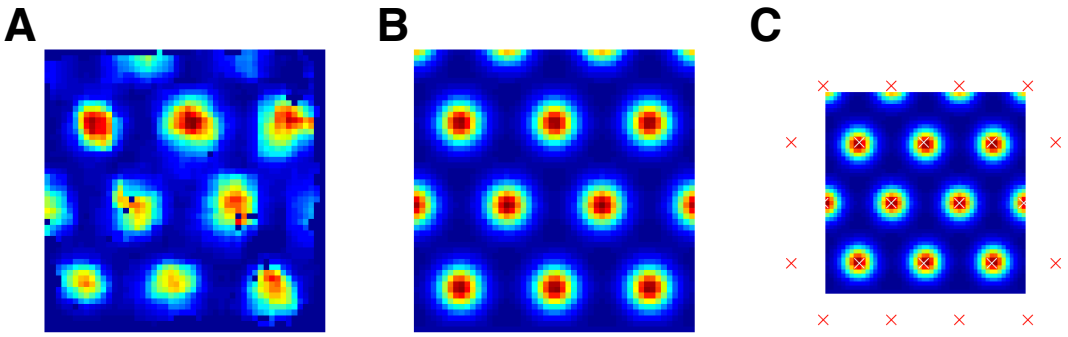

Figure 9.2: Fitting ideal grid maps A-C) Ideal grid. We fit an idealised grid rate map using the formula from (Stemmler et al., 2015) to the original grid cell rate maps to remove any possible confounds and to ensure that we obtain accurate grid cell peaks. **A** An example original grid cell rate map. **B)** An idealised rate map fit to that in **A)**. **C)** Accurate finding of grid cell peaks (white crosses) on the idealised grid rate map, which also allows peaks that extend outside the box to be used (red crosses).

resulting rate maps are likely to be less representative of the grid cell population. Both grid score and scale were computed as in Barry et al. (2012).

Fitting idealised grids to ensure grid-peaks were well-defined We fit the recorded grid cell rate maps to an idealised grid cell formula (equation 6 from Stemmler et al. (2015)), and use this ideal grid rate map to give grid cell firing rates and locations of grid peaks (Figure 9.2A,B,C). This leads to a very strenuous control as it ensures that results cannot be driven by *any* differences across grid cells apart from grid phase, grid scale and grid angle (which are the only fitted parameters). This additionally allowed us to use grid peaks that were outside the box. We only fitted idealised grids in situations where we also defined a grid-score cut off (g=0.8) to ensure good model fits.

Removing place cells at borders to ensure effects are not driven by border cells Here we removed all cells whose peaks were $\leq 10\%$ of environment width from the border. The reason we wish to account for border effects is because non-grid MEC cells (such as border cells) rather than grid cells may drive place cell remapping to the borders. We have this criteria for all our analyses.

Ensuring cells have actually remapped. Though not data-preprocessing, we ensure that any results could not be confounded by place cells and/or grid cells not remapping/ realigning (i.e. the animal thinking it was still in the same box!).

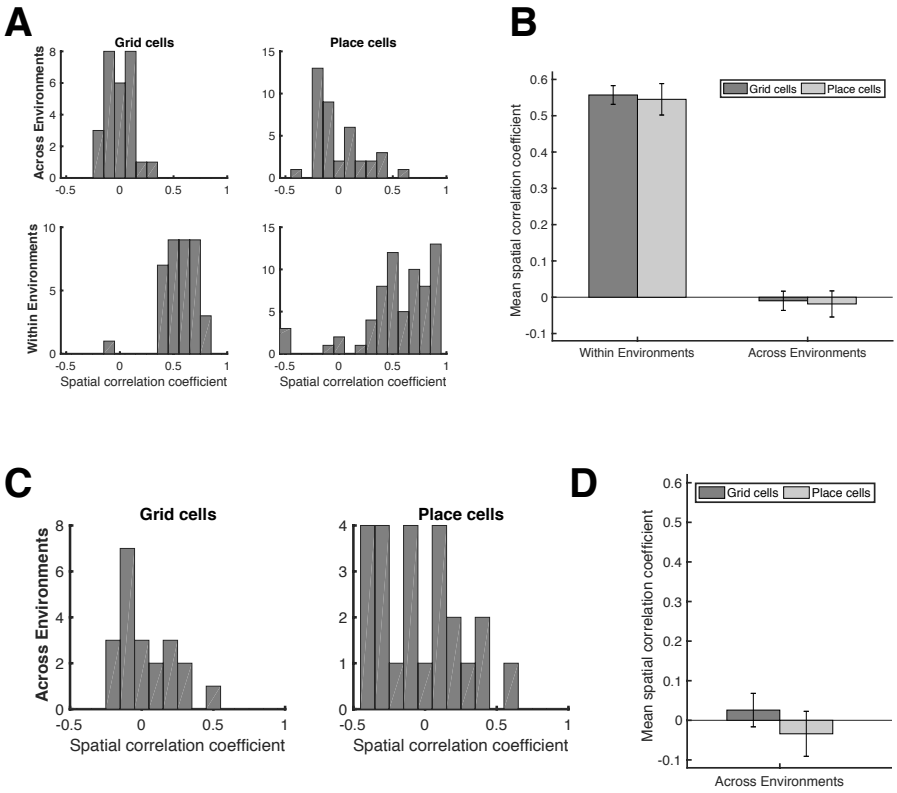

Figure 9.3: Grid cells reallign and place cells remap. A-B) Grid realignment and place cell remapping across environments in dataset 1. **A)** Histograms showing the distributions of spatial correlations for place and grid cells both within and across environments. **B)** Bar plots showing the mean (± SEM) of these distributions. **C-D)** Grid realignment and place cell remapping across environments in dataset 2. **C** and **D** are the same analyses as **A** and **B** but with dataset 2. They demonstrate distributions of spatial correlations near 0 for dataset 2. **D** has its axis locked to that of **B** for visualisation.

We test this by examining the distributions of spatial correlations obtained when correlating a given place or grid cell's rate map in one environment with its rate map in a second visit to that same environment (within environments; only possible in dataset 1) or its rate map in a different environment (across environments). In dataset 1, we found that all the grid cells realigned across environments and the place cells remapped, with spatial correlation coefficients around 0 and distributions similar to those observed in hippocampal global remapping experiments (Fyhn et al., 2007) (Figure 9.3A,B). On the other hand, spatial correlations were high upon a second visit to the same environment. Distributions of spatial correlations near 0 for both place and grid cells across environments were also found in dataset 2 (Figure 9.3C,D). These results suggest that, as expected, grid cells realigned across the

environments and the place cells accordingly underwent global remapping; global place cell remapping generally accompanies grid realignment (Fyhn et al., 2007). That the place and grid cell spatial correlations were near zero means it would be a non-trivial result should the place and grid cell relationship be preserved.

9.3.2 Computing the measures

We first perform the data-preprocessing, making each cell pass the appropriate checks.

We would like to know whether the relationship between place and grid cell pairs is preserved across environments. We propose 2 measures.

1) Does a given grid cell fire similarly at the respective peaks of a given place cell in both environments? We take a place cell and look at its peak in both environments, which we call P1 and P2. We then take a grid cell, and look at its firing rate at P1 in env1 - we call this X. We look at its firing rate in env2, we call that Y. This gives us a datapoint [X,Y]. We then do this again for the next grid cell, which gives another datapoint. We loop through all the grid cells and place cells for the same animal. Then start again for the next animal. We can then plot all these points on a graph, and find the correlation coefficient - this is the gridAtPlace measure (Figure 9.4)

2) Does a given grid cell peak at a similar distance from the respective peaks of a given place cell in both environments? For this "MinDist" measure, we do the same process as above, but X is now the minimum distance of a grid peak in env1 from P1, and Y is the minimum distance of a grid peak in env2 from P2. We normalise X,Y by grid scale of that grid cell. Note that the minDist measure is only calculated in analyses that fit idealised grids (to cells with grid score 0.8) to ensure that grid peaks are estimated effectively.

For the place cells, we analysed cells defined as place cells in Barry et al. (2012) and Chen et al. (2018a). Locations of place cell peaks were simply defined as the location of maximum activity in a given cell's rate map.

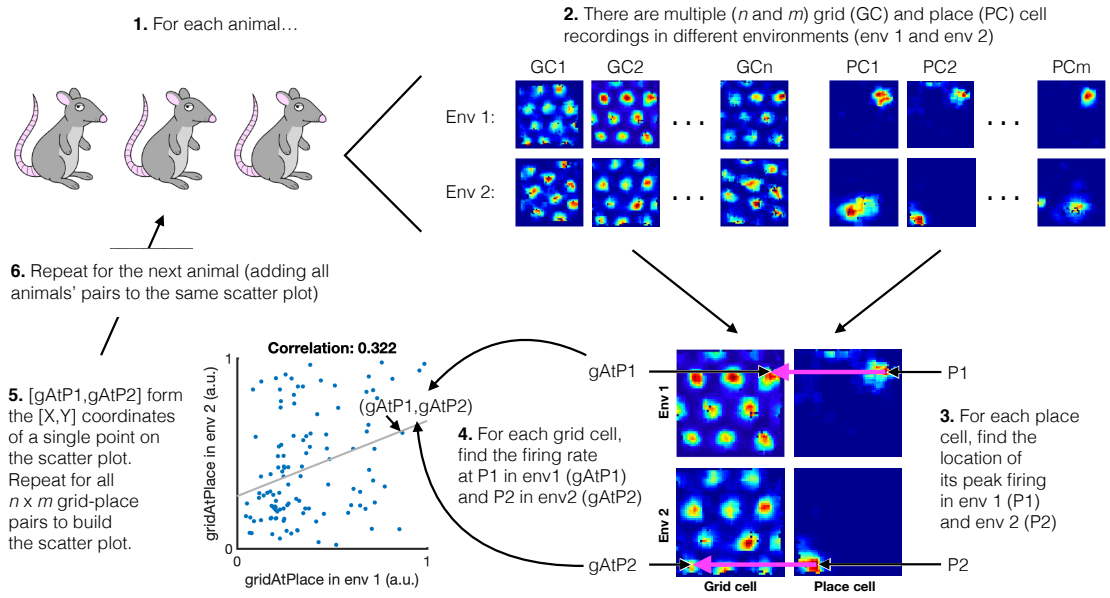

Figure 9.4: Schematic of analysis showing preserved grid-place relationships after remapping. Schematic explaining the gridAtPlace analysis. Specifically how the scatter plot is generated. Note that in this figure original grid cell rate maps are shown, rather than ideal grid cell rate maps (Figure 9.2A-C) that were used to generate the main text figures.

We require each place-grid pair to come from the same animal, but we do not require that the place and grid cells were simultaneously recorded i.e. a place cell may be paired with a grid cell from a different recording session.

Note: If there were only a single grid frequency (or module) in entorhinal cortex, TEM would predict a near perfect correlation across environments between gridAtPlace scores for each grid-cell place-cell pair. However, if either (1) place cells are influenced by phases of more than a single grid module or (2) place cells predominantly received input from a single grid module, but we (the experimenter) do not know which module (as is the case), then we should not predict perfect correlations, only non-zero correlations.

9.3.3 Statistical testing

To test the significance of this correlation and ensure it is not driven by bias in the data, we generated a null distribution by permuting the place cell peak (5000 times) and recomputing the measures and their correlation across trials. We use

two possible ways of permuting. Firstly, we choose a position randomly (but still passing our pre-processing steps). Secondly we choose a position from another recorded cell (cells from same and other animals to get enough combinations). We then examine where the correlation coefficient of the non-shuffled data lies relative to the null correlation coefficients to determine its statistical significance. These analyses were carried out separately for both datasets. Again, results from both procedures (for all tests) are reported in table 9.2.

9.4 Which cell types generalise their structure across environments?

As a brief interlude before the main result, we first test whether the correlation structure of each cell type generalises across environments.

9.4.1 Grid cells realign and keep their correlation structure

Indeed, although grid cells realign across environments, their correlation structure is preserved (Fyhn et al., 2007). Although this has been previously demonstrated, we also showed it to be true by demonstrating that the correlation structure between the grid cells was itself correlated (i.e. preserved) across environments. More specifically, we calculated the grid cell by grid cell spatial correlation matrix in one environment, and correlated its upper triangle with that of the correlation matrix in the other environment (a correlation matrix of the same grid cells, but computed in a different environment). We tested this in the single animal with the most recorded grid cells across both environments in each dataset (in a rat with 15 grid cells in dataset 1 [comparing trials 1 and 4], and a mouse with 13 grid cells in dataset 2). This was significant relative to a null distribution generated by permuting grid cell-grid cell pair correlations in both dataset 1 ($r=0.55$, $p<0.001$; Figure 9.5A) and dataset 2 ($r=0.95$, $p < 0.001$; Figure 9.5C). These results are expected if the grid cells encode knowledge that generalises across environments. A similar result has previously been reported in Yoon et al. (2013) based on data that was included in our analyses.

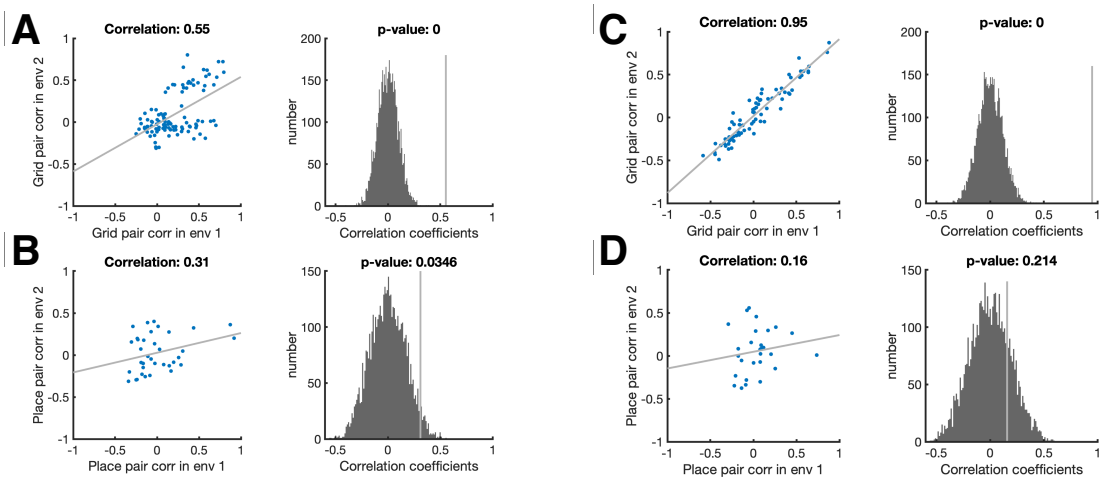

Figure 9.5: The grid cell correlation structure is preserved across environments. A-B) The grid cell correlation structure is preserved across environments in dataset 1. **A)** Dataset 1. Scatter plot shows the correlation across environments of the spatial correlations of grid cell-grid cell pairs (i.e. the correlation of the upper triangle of two grid cell by grid cell correlation matrices: one from environment 1 and one from environment 2). The histogram shows this correlation coefficient was significant relative to a null distribution of correlation coefficients obtained by permuting grid cell-grid cell pairs. **B)** Same as **A** for place cells. **C-D)** Replication of preserved grid cell correlation structure across environments in dataset 2. **C** and **D** are the same format as **B** and **C**.

9.4.2 Place cells remap, only weakly retaining correlation structure across environments

We also found this effect to be weakly significant in place cells in dataset 1 (r=0.31, p=0.035; Figure 9.5B) and not significant in dataset 2 (r=0.16, p=0.21; Figure 9.5D).

9.5 Preserved relationship between grid and place cells across environments

Back to our main results in examining whether grid-cell place cell relationships are preserved across environments using our two measures (gridAtPlace and MinDist).

9.5.1 Dataset 1 - Barry et al 2012

As a sanity check, we first confirmed these measures were significantly correlated within environments, i.e. correlated across two visits to the same environment (trials 1 and 5), when the cell populations have not remapped. We see that for both measures there is a significant correlation across the trials (the true correlation

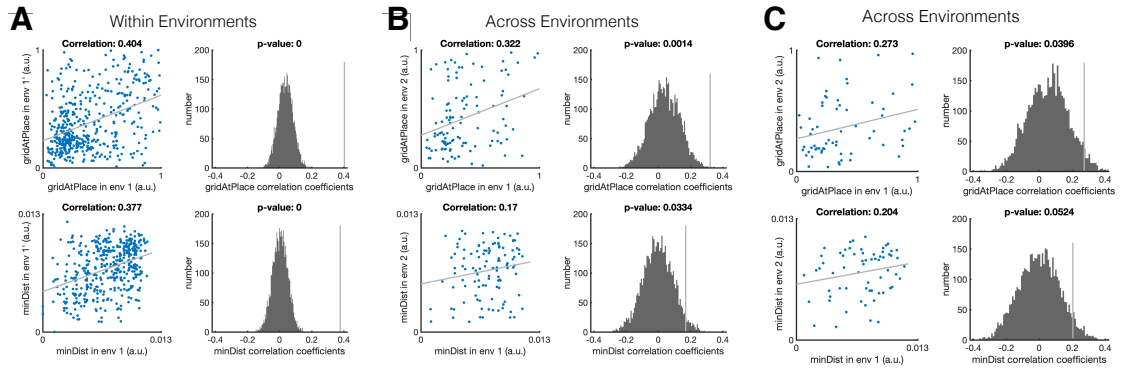

Figure 9.6: Structural knowledge is preserved over apparently random hippocampal remapping. Preserved relationship between place and grid cells across environments in dataset 1. The scatter plots show the correlation of a given measure across trials, where each point is a place cell-grid cell pair. The histogram plots show where this correlation (grey line) lies relative to the null distribution of correlation coefficients. The p value is the proportion of the null distribution that is greater than the unshuffled correlation. **A)** gridAtPlace (top) and minDist (bottom) measures are strongly significantly correlated over two trials within the same environment, as expected given the same place and grid code should be present. **B)** These measures are also significantly correlated across the two different environments, providing evidence that place and grid cells retain their relationship across environments. **C)** Replication of the preserved relationship between place and grid cells across environments in dataset 2. The gridAtPlace measure is significantly correlated at $p<0.05$ across real and virtual worlds and the minDist measure is trending very close to significance, replicating the preserved relationship between grid and place cells across environments.

coefficient is above 95% of the null distribution of correlation coefficients; Figure 9.6A), for 445 place cell-grid cell pairs. This indicates that upon returning to the same environment, place cells and grid cells have retained their relationship with each other, as expected.

We then tested across environments, i.e. visits to two different environments (trials 1 and 4), to assess whether our predicted non-random remapping relationship between grid and place cells exists. Here we also find significant correlations for all combinations of measures, preprocessing decisions and statistical tests (Table 9.2). Data for the most stringent/conservative set of inclusion criteria (grid score $>$ 0.8, leaving 115 cell pairs) are shown in (Figure 9.6B, gridAtPlace $p < 0.005$, minDist$p < 0.05$)

Dataset	Cut-off	Ideal	Perm	GAP r	GAP p	MD r	MD p
1	0.8	1	0	0.322	**0.0014**	0.170	**0.0334**
1	0.3	0	0	0.654	**0.0094**	-	
1	0	0	0	0.630	**0.0318**	-	
1	0.8	1	1	0.322	**0.0038**	0.170	**0.0352**
1	0.3	0	1	0.654	**0.0356**	-	
1	0	0	1	0.630	0.0776	-	
2	0.8	1	0	0.273	**0.0396**	0.204	0.0524
2	0.3	0	0	0.554	**0.0282**	-	
2	0	0	0	0.544	**0.0404**	-	
2	0.8	1	1	0.273	**0.0468**	0.204	0.058
2	0.3	0	1	0.554	**0.0340**	-	
2	0	0	1	0.544	**0.0468**	-	

Table 9.2: Analysis results for different parameter settings. Ideal grid 1/0 is whether we used a fitted ideal grid ratemap or not respectively. Permutation type 0/1 is whether we sampled a place cell peaks randomly, or from another recorded place cells. GAP/MD is gridAtPlace and MinDIst respectively. We show the r value and the significant p for both. Results below a 0.05 significance threshold are shown in bold. Our analysis results are robust to parameter settings, with the non-significant results also trending.

9.5.2 Dataset 2 - Chen et al 2018

In this dataset, we only have measures for across environments, i.e. visits to the real and virtual worlds. We again found that the gridAtPlace measure was significant across all combinations of measures, preprocessing decisions and statistical tests (Table 9.2). Here the minDist measure is trending significant (but note this dataset has far fewer cell pairs (Table 9.2). Figure 9.6C shows data for the 64 pairs that survived the most stringent inclusion criteria (gridAtPlace $p < 0.05$, minDist $p = 0.0524$)

9.5.3 Remarks

These results are, to our knowledge, the first analyses demonstrating non-random place cell remapping based on neural activity, support a key prediction of our model TEM: that hippocampal place cells, despite remapping across environments, retain their relationship with the entorhinal grid, providing a substrate for structural inference.

10 Afterword

In this thesis we presented two strands of research united by a common theme of Bayesian reasoning. The first showed that framing supervised learning probabilistically allows for approximate inference to be implemented in biological neural networks. Furthermore, learning in these networks is equivalent to the famous back-propagation algorithm. This work demonstrated that algorithms as efficient as those in machine learning could be implemented in the brain. The second presented a framework for learning and generalising abstract relational knowledge. This framework unifies spatial cognition with relational memory in the hippocampal formation. A model implementation, TEM, learned to represent relational knowledge using neural representations that mirror those found in the brain. Furthermore, a prediction of TEM was verified in neural recordings. We now highlight some open questions and challenges for research.

Biological backprop

Though the models discussed in chapters 4 and 5, break the decodes old dogma that backprop is not possible in the brain, a full understanding of the brain's learning algorithm still present several important challenges.

Weight symmetry

Are biologically plausible deep learning implementations robust to the lack of symmetry between feedforward and feedback connections? The four models reviewed in chapter 5 use symmetric feedforward and feedback weights. In these models, both sets of weights are modified during learning, and the plasticity rules maintain the symmetry. Such symmetry does not exist in brain networks, so it is important to continue investigations into whether biologically plausible networks still perform robustly without weight symmetry. Early work along these lines include Lillicrap et al. (2016); Akrout et al. (2019).

Implementations that scale

How can researchers make biologically plausible deep learning implementations scale? Although the model of chapters 4 and 5 perform well on some tasks, it is unclear whether they scale to larger problems. This is in part due to the multiple iterations required to update node activity via network dynamics. The number of iterations required does not currently scale well for larger networks. Further work optimising this process is required if high depth networks are to be trained. Early work along these lines include Bartunov et al. (2018); Akrout et al. (2019).

Backprop through time

How can efficient learning of temporal sequences be implemented in biological networks? The models of chapters 4 and 5 focus on a case of static input patterns, but the sensory input received by the brain is typically dynamic, and the brain has to learn to recognise sequences of stimuli (e.g. speech). To describe learning in such tasks, artificial neural networks have been extended to include recurrent connections among hidden units, which provide a memory of the past. It is important to extend the models for learning through time. Early work along these lines include Bellec et al. (2019).

Optimising dynamics

How can the dynamics of neural circuits be optimised to support efficient learning? This question can be first studied in models of primary sensory areas predicting sensory input from its past values. In such tasks, the dynamics will play an important role, as networks need to generate their predictions at the right time to compare it with incoming sensory data.

The Tolman-Eichenbaum Machine

While TEM is a valiant first attempt to formalise generalisation as state space abstraction, it as yet only generalises in simple tasks where the state space is well defined. Humans and animals do much more than this.

In new settings, finding the appropriate state abstractions is hard, though should one be able to do so, understanding and new learning becomes easy. For this reason, this research direction is of great importance. Several key challenges remain.

Representing complex tasks

Tasks are often embedded within state spaces. In such cases, there is then a base state space, e.g. 2D space, and a task state space, e.g. cooking. These two state-spaces are factored from each other, since cooking can happen anywhere in space, but nevertheless cooking is still embedded in space. Building representations that build off each other compositionally offers an elegant form of task generalisation. Extensions to TEM that explicitly account for this may offer further explanations of neural representations including those in pre-frontal cortex.

Generalisation in non-transition based tasks

TEM requires transitions on graphs. This does not capture all relational knowledge. For example, we don't need transitions to generalise the line concept between a queue of people and a line of ducks, nor do we need to saccade between bees to realise a swarm. Processing such visual relational concepts is believed to rely on parietal cortex (Nieder, 2012; Summerfield et al., 2019). Extending TEM principles

(e.g. separate relational and sensory representations) could similarly be useful for generalisation, as well as offer explanations to parietal responses.

Representations of reinforcement learning

The presented framework brings together spatial cognition and relational memory though generalisation. These hippocampal functions and representations do not require rewards. This is unlike the majority of computational neuroscience experimental studies which involve analysis of neural representations from animals under reward-guided behaviours (reinforcement learning tasks; RL). RL has told us much about the algorithms that may take place in the brain (Schultz et al., 1997), but little about the representations that are used. Clearly humans and animals learn something general about state-spaces in RL tasks, but it is unknown how, or what they learn. We hypothesise that similar principles to those in TEM can provide explanations of the neural representation in RL tasks. This has the potential to explain a vast swath of experimental data under a single framework, and would solve a major problem in this history of neuroscience – unifying map-like learning and reinforcement learning.

Representing others

We are not alone in this world. Our representations of state must reflect this by including others, along with their intentions and emotions, so that we can reason over them. This constitutes a formidable challenge, and is an active area of research in (multi agent) RL (Foerster et al., 2016). Understanding recursive abstractions of others in a formal framework such as TEM may offer insights into theory of mind as well as the representations of social neuroscience e.g. Gallese et al. (1996).

Final remarks

These challenges present the neuroscience and machine learning community with difficult, yet interesting times ahead. Solving them would offer deep insights into the nature of intelligence, the human mind, and ultimately consciousness.

Appendices

A

Learning with Bayes

Contents

A.1 Graphical models

A graphical model is a probabilistic model that pictorially represents variables and their conditional dependencies. To see how, let us take an arbitrary joint distribution of variables (three here for clarity), and show that conditional independencies lead to a structured interpretation of the variables. For three variables, the conditional chain rule prescribes

$$P(x_1, x_2, x_3) = P(x_1 \mid x_2, \mathbf{x}_3)P(x_2 \mid x_3)P(x_3) \tag{A.1}$$

Here an arbitrary permutation of the indices on the right hand side also holds. We may, however, have additional knowledge (/assumptions / restrictions) about

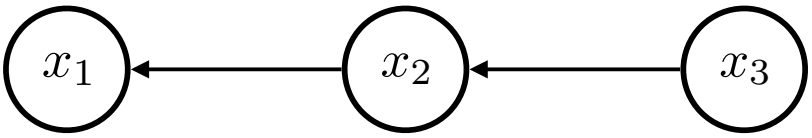

Figure A.1: Arrows show conditional dependencies of variables.

how these variables impact each other. For example, knowing x_3 does not tell us anything more about x_1 should we have already observed x_2. More formally, x_1 is conditionally independent of x_3 given x_2, and thus $P_\theta(x_1 \mid x_2, x_3) = P(x_1 \mid x_2)$. Knowing such properties simplifies the joint distribution

$$P(x_1, x_2, x_3) = P(x_1 \mid x_2)P(x_2 \mid x_3)P(x_3) \tag{A.2}$$

Not only simpler, but also we can see a hierarchical ordering of the variables - x_3 generates x_2 which in turn generates x_1. This sort of structuring of variables is exactly what graphical models capture (Figure A.1). Rather than starting with the conditional independencies leading to the graphical model, we can equally start with a graphical model and then read off conditional independencies.

A.2 Bayesian approximations

In chapter 2, we describe the difficulty of calculating the posterior distribution. Fortunately all is not lost, and there are many well developed techniques to proceed with. These include sampling based techniques (MCMC), approximate Bayesian computation (ABC) and variational inference. We focus on variational inference here as it is relevant to this thesis.

A.2.1 Variational methods

Rather than attempting to compute the exact posterior, variational methods instead compute an approximation. A good approximate posterior is one that matches the true posterior as much as possible. To quantify how well these two distributions match we consider a distance metric between distributions

$$\mathrm{D_{KL}}(\mathrm{Q}_\phi\left(\mathbf{z}^{1:\mathrm{N}} \mid \mathbf{x}^{1:\mathrm{N}}\right) \| \mathrm{P}_\theta\left(\mathbf{z}^{1:\mathrm{N}} \mid \mathbf{x}^{1:\mathrm{N}}\right)) = \int \mathrm{Q}_\phi\left(\mathbf{z}^{1:\mathrm{N}} \mid \mathbf{x}^{1:\mathrm{N}}\right) \ln \frac{\mathrm{Q}_\phi\left(\mathbf{z}^{1:\mathrm{N}} \mid \mathbf{x}^{1:\mathrm{N}}\right)}{\mathrm{P}_\theta\left(\mathbf{z}^{1:\mathrm{N}} \mid \mathbf{x}^{1:\mathrm{N}}\right))} d\,\mathbf{z}^{1:\mathrm{N}} \tag{A.3}$$

To get the best possible approximate distribution we want the following

$$\mathrm{Q}_\phi\left(\mathbf{z}^{1:\mathrm{N}} \mid \mathbf{x}^{1:\mathrm{N}}\right)^* = \underset{\mathrm{Q}_\phi(\mathbf{z}^{1:\mathrm{N}}|\mathbf{x}^{1:\mathrm{N}})}{\arg\min}\, \mathrm{D_{KL}}(\mathrm{Q}_\phi\left(\mathbf{z}^{1:\mathrm{N}} \mid \mathbf{x}^{1:\mathrm{N}}\right) \| \mathrm{P}_\theta\left(\mathbf{z}^{1:\mathrm{N}} \mid \mathbf{x}^{1:\mathrm{N}}\right)) \tag{A.4}$$

This equation is still difficult to compute, again because the posterior still appears. After a little manipulation, in particular substituting Bayes theorem, we obtain

$$\mathrm{D_{KL}}(\mathrm{Q}_\phi\left(\mathbf{z}^{1:\mathrm{N}} \mid \mathbf{x}^{1:\mathrm{N}}\right) \| \mathrm{P}_\theta\left(\mathbf{z}^{1:\mathrm{N}} \mid \mathbf{x}^{1:\mathrm{N}}\right)) = \ln \mathrm{P}_\theta\left(\mathbf{x}^{1:\mathrm{N}}\right) + \mathbb{E} \ln \mathrm{Q}_\phi\left(\mathbf{z}^{1:\mathrm{N}} \mid \mathbf{x}^{1:\mathrm{N}}\right) - \mathbb{E} \ln \mathrm{P}_\theta\left(\mathbf{x}^{1:\mathrm{N}}, \mathbf{z}^{1:\mathrm{N}}\right) \tag{A.5}$$

Where $\mathbb{E}$ is the expectation over the distribution $\mathrm{Q}_\phi\left(\mathbf{z}^{1:\mathrm{N}} \mid \mathbf{x}^{1:\mathrm{N}}\right)$. Even though the intractable $\ln \mathrm{P}_\theta\left(\mathbf{x}^{1:\mathrm{N}}\right)$ appears here, we are making progress. This is because, though we can't directly optimise $\mathrm{D_{KL}}(\cdots)$, we can instead optimise the second two terms which are equivalent to $\mathrm{D_{KL}}(\cdots)$, up to the additive constant $\ln \mathrm{P}_\theta\left(\mathbf{x}^{1:\mathrm{N}}\right)$. This is the key to the variational approach - since we are optimising and $\ln \mathrm{P}_\theta\left(\mathbf{x}^{1:\mathrm{N}}\right)$ is a constant with respect to latent variables, we can ignore it! The second two terms together are known as the free energy F or $ELBO$ - the evidence lower bound.

$$ELBO = \mathbb{E} \ln \mathrm{Q}_\phi\left(\mathbf{z}^{1:\mathrm{N}} \mid \mathbf{x}^{1:\mathrm{N}}\right) - \mathbb{E} \ln \mathrm{P}_\theta\left(\mathbf{x}^{1:\mathrm{N}}, \mathbf{z}^{1:\mathrm{N}}\right) \tag{A.6}$$

It is given this name as it lower bounds $\ln \mathrm{P}_\theta\left(\mathbf{x}^{1:\mathrm{N}}\right)$ since D_{KL} is always positive

$$\ln \mathrm{P}_\theta\left(\mathbf{x}^{1:\mathrm{N}}\right) = \mathrm{D_{KL}}(\mathrm{Q}_\phi\left(\mathbf{z}^{1:\mathrm{N}} \mid \mathbf{x}^{1:\mathrm{N}}\right) \| \mathrm{P}_\theta\left(\mathbf{z}^{1:\mathrm{N}} \mid \mathbf{x}^{1:\mathrm{N}}\right)) + ELBO \tag{A.7}$$

A.3 Optimising the ELBO

The great benefit of the variational method is that it translates an integration problem into an optimisation problem, and optimisation problems are generally easier that integration problems. The only thing we then need to consider is how to optimise the $ELBO$, and, if necessary, how to specify the approximate posterior $\mathrm{Q}_\phi\left(\mathbf{z}^{1:\mathrm{N}} \mid \mathbf{x}^{1:\mathrm{N}}\right)$. When we choose $\mathrm{Q}_\phi\left(\mathbf{z}^{1:\mathrm{N}} \mid \mathbf{x}^{1:\mathrm{N}}\right)$, there is naturally there is a trade-off between the expressiveness of the posterior distribution, and the ease of optimisation.

There are numerous ways to approach these problems - from expectation maximisation (Dempster et al., 1977), variational Bayes (Jordan et al., 1999), stochastic variational inference (Hoffman et al., 2013), black-box variational inference (Ranganath et al., 2014), and variational autoencoders (Kingma and Welling, 2013).

We will discuss the expectation-maximisation algorithm (following the discussion in (Dayan and Abbott, 2001)) as it is relevant to chapter 4, and variational autoencoders as they are relevant to chapters 7 and 8.

A.3.1 Expectation Maximisation

Expectation maximisation (EM) takes the approach of optimising the $ELBO$ iteratively during two phases - the expectation phase (E) and the maximisation phase (M). In the E phase, the $ELBO$ is increased with respect to Q, keeping θ constant. In the M phase, the $ELBO$ is increased with respect to θ while keeping Q fixed. There are a number of different ways to proceed with the E phase, two of which we discuss now.

Firstly, for **invertible models**, noting that the $ELBO$ is maximised with respect to Q, when $Q = P$, should we be able to compute P, the posterior over causes then this is the optimum Q. This completes the E step. For the M step, we simply substitute $Q = P$ in equation A.8, and then maximise with respect to the parameters.

Secondly, for the much wider class of **non-invertible models**, we will not find the E step so easy. Here, we parameterize, with ϕ, an approximate recognition distribution - $\mathrm{Q}_\phi\left(\mathbf{z}^{1:\mathrm{N}} \mid \mathbf{x}^{1:\mathrm{N}}\right)$. The $ELBO$ then becomes a function of ϕ and θ.

The E phase now becomes optimising the $ELBO$ with respect to ϕ, this has the effect of making $\mathrm{Q}_\phi\left(\mathbf{z}^{1:\mathrm{N}} \mid \mathbf{x}^{1:\mathrm{N}}\right)$ more similar to $\mathrm{P}_\theta\left(\mathbf{z}^{1:\mathrm{N}} \mid \mathbf{x}^{1:\mathrm{N}}\right)$. The M phase, as before, maximises the $ELBO$, with respect to θ.

The procedure is to perform an E phase, then an M phase, then and E phase and so on iteratively. For non-invertible models, the E and M phases are also often individually iterative. This means that the EM algorithm can become computationally very expensive. To alleviate this, it is not always necessary to maximise θ (and sometimes ϕ) fully at each iteration. Instead gradient ascent in small amounts can be taken at each step.

A.3.2 Variational auto-encoders

Variational autoencoders are a follow on work from the wake-sleep algorithm (Hinton et al., 1995) and the helmholtz machines (Dayan et al., 1995) as discussed in chapter 2. Variational autoencoders assume a parameterized form of both $\mathrm{P}_\theta(\mathbf{x}, \mathbf{z})$ and $\mathrm{Q}_\phi(\mathbf{z} \mid \mathbf{x})$, in particular the sufficient statistics of the distributions are parameterized by neural networks.

Assuming a factorisation over data samples

$$ELBO = \sum_{\mathbf{x}\, in\, \mathcal{D}} \mathbb{E}\left[\ln \mathrm{Q}_\phi(\mathbf{z} \mid \mathbf{x}) - \ln \mathrm{P}_\theta(\mathbf{x}, \mathbf{z})\right] \tag{A.8}$$

Taking gradients with respect to θ is easy enough

$$\nabla_\theta \mathbb{E} \ln \mathrm{P}_\theta(\mathbf{x}, \mathbf{z}) = \mathbb{E}\, \nabla_\theta \ln \mathrm{P}_\theta(\mathbf{x}, \mathbf{z}) \tag{A.9}$$

Of which we can easily take unbiased estimators. The difficulty lies with taking gradient with respect to ϕ, where

$$\nabla_\phi \mathbb{E}\left[\ln \mathrm{Q}_\phi(\mathbf{z} \mid \mathbf{x}) - \ln \mathrm{P}_\theta(\mathbf{x}, \mathbf{z})\right] \neq \mathbb{E}\, \nabla_\phi \left[\ln \mathrm{Q}_\phi(\mathbf{z} \mid \mathbf{x}) - \ln \mathrm{P}_\theta(\mathbf{x}, \mathbf{z})\right] \tag{A.10}$$

As such, obtaining unbiased gradients is more difficult. This is what led Hinton and colleagues to optimise the reverse-KL for the sleep phase in the wake-sleep

algorithm. This is why the overall objective that gets optimised is no longer the *ELBO* in the wake-sleep algorithm or the Helmholtz machine.

The trick that was found in both Kingma and Welling (2013); Rezende et al. (2014), was that (for certain distributions) a change of variables allows both θ and ϕ to be optimised. This is known as the reparameterization trick. The change of variables is

$$\mathbf{z} = g(\boldsymbol{\varepsilon}, \mathbf{x}, \phi) \tag{A.11}$$

Where $\boldsymbol{\varepsilon}$ is a random variable with distribution π independent of ϕ and θ. Distributions that allow reparameterization include the normal distribution. With this trick, an expectation $\mathbb{E}_{\mathbf{z}\sim \mathrm{Q}_\phi(\mathbf{z}|\mathbf{x})}$ is equivalent to the expectation $\mathbb{E}_{\varepsilon\sim\pi}$. Back to equation A.10, this can be rewritten as

$$\begin{aligned}\nabla_\phi \mathop{\mathbb{E}}_{\mathbf{z}\sim \mathrm{Q}_\phi(\mathbf{z}|\mathbf{x})} \left[\ln \mathrm{Q}_\phi\left(\mathbf{z} \mid \mathbf{x}\right) - \ln \mathrm{P}_\theta\left(\mathbf{x}, \mathbf{z}\right)\right] &= \nabla_\phi \mathop{\mathbb{E}}_{\varepsilon\sim\pi} \left[\ln \mathrm{Q}_\phi\left(\mathbf{z} \mid \mathbf{x}\right) - \ln \mathrm{P}_\theta\left(\mathbf{x}, \mathbf{z}\right)\right] \\ &= \mathop{\mathbb{E}}_{\varepsilon\sim\pi} \nabla_\phi \left[\ln \mathrm{Q}_\phi\left(\mathbf{z} \mid \mathbf{x}\right) - \ln \mathrm{P}_\theta\left(\mathbf{x}, \mathbf{z}\right)\right]\end{aligned} \tag{A.12}$$

Where $\mathbf{z} = g(\boldsymbol{\varepsilon}, \mathbf{x}, \phi)$. We can easily get unbiased estimates of the above gradient.

B

Further predictive coding details.

Contents

B.1 Variational derivation of supervised predictive coding

Here we derive supervised predictive coding networks using a variational approximation (similar to chapter 2).

Let us start with the joint distribution of all variables conditioned on the input $\mathbf{x}_1$. We assume it factorises as follows

$$P_\theta(\mathbf{x}_L, ..., \mathbf{x}_2 \mid \mathbf{x}_1) = \prod_{l=2}^{L} P(\mathbf{x}_l \mid \mathbf{x}_{l-1}) \tag{B.1}$$

Where $P(\mathbf{x}_l \mid \mathbf{x}_{l-1}) = \mathcal{N}(\mathbf{x}_l; \boldsymbol{\mu}_l, \Sigma_l)$ and $\boldsymbol{\mu}_l = \Theta_{l-1} f(\mathbf{x}_{l-1}) + \mathbf{b}_l$. We leave Σ_l as a learnable set of parameters.

We want to use variational inference. We assume a recognition density

$$Q_\phi(\mathbf{x}_L, ..., \mathbf{x}_2) = \prod_{l=2}^{L} \delta(\mathbf{x}_l - \boldsymbol{\phi}_l) \tag{B.2}$$

Substituting this into equation 2.15, and (with a serious abuse of notation) relabelling ϕ_l as $\mathbf{x}_l$, we obtain

$$F = -\frac{1}{2} \sum_{l=2}^{L} (\mathbf{x}_l - \boldsymbol{\mu}_l)^T \Sigma_l^{-1} (\mathbf{x}_l - \boldsymbol{\mu}_l) - \frac{1}{2} \ln(|\Sigma_l|) + Const \tag{B.3}$$

Which is the desired result. Since we are just finding the most likely set of variables $\mathbf{x}$, this is equivalent to maximum a posteriori inference and learning.

A slight admission is that we have hidden included in *Const* terms such as

$$\int \delta(\mathbf{x}_l - \boldsymbol{\phi}_l) \ln \delta(\mathbf{x}_l - \boldsymbol{\phi}_l) \tag{B.4}$$

Which are infinite, though nevertheless don't change things when taking gradients.

B.2 Learning covariance terms

We can, if we wish, also lean the covariance parameter. We would then obtain the following update rule

$$\Delta \boldsymbol{\Sigma_a} = \frac{1}{2} \left(\boldsymbol{\varepsilon}_a \, \boldsymbol{\varepsilon}_a^T - \boldsymbol{\Sigma_a}^{-1} \right) \tag{B.5}$$

This is now an issue for Hebbian learning, as computing the matrix inverse term cannot be done using local information alone. This has been investigated before Bogacz (2017) where it has been shown that the covariance parameter can be learned in a Hebbian manner by adding an extra interneuron, however at the cost of more complex network dynamics.

C

The Tolman-Eichenbaum Machine

Contents

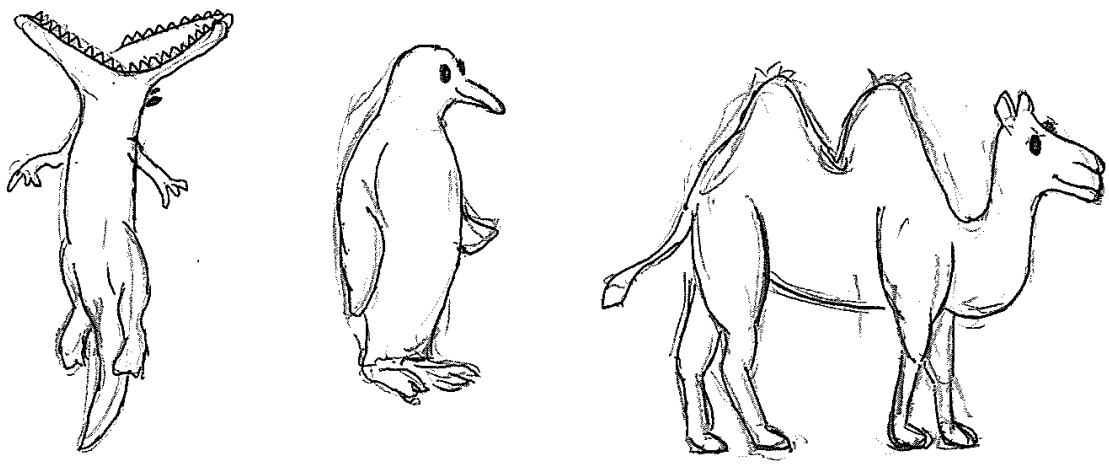

Figure C.1: TEM in animal form by Jacob Bakermans.

C.1 Simulation and analysis details

All the tasks described below are best 'solved' if the underlying structure is learned, even though each structure is different. We now describe the details for the types of graphs we considered, as well as the simulation details.

For all simulations presented above, we use the additional memory module (two separate memory matrices) in grid cell inference. Each time the agent enters a new environment, both memory matrices, M, are reset (all weights zero). Asides from when otherwise stated, the agent randomly diffuses through each environment.

The agent is initially randomly placed in each environment. The agent changes to a completely new environment after a certain number of steps ($\sim$ 2000-5000 for the 2D graph worlds, lower for smaller environments/ tasks). For 2D graph worlds, typically after $200-300$ environments, the agent has fully learned the structure and how to address memories. This equates to ~ 50000 gradient updates (1 gradient update per block of 25 steps). For smaller worlds, learning is much faster.

We now describe the dimensions of variables (summarised in Table C.1). We use $n_s = 45$ (the number of different sensory objects), $n_{s^*} = 10$ (the compressed sensory dimension) and 5 different frequencies. The number of TEM entorhinal cells in each frequency are $[30, 30, 24, 18, 18]$, and the number of TEM entorhinal cells that project to TEM hippocampus, n^f are $[10, 10, 8, 6, 6]$ (i.e. the first 1/3 entorhinal cells in each frequency). Thus the number of hippocampal cells in each

Variable	Freq 1	Freq 2	Freq 3	Freq 4	Freq 5	Total
$\mathbf{g}$	30	30	24	18	18	120
$f_g(\mathbf{g})$	10	10	8	6	6	120
$\mathbf{p}$	100	100	80	60	60	400
$\mathbf{x}^f$	10	10	10	10	10	50
$\mathbf{x}^c$	10	-	-	-	-	-
$\mathbf{x}$	45	-	-	-	-	-

Table C.1: Table showing the number of neurons for each variable; Entorhinal $\mathbf{g}$, Entorhinal subsampled $f_g(\mathbf{g})$, Hippocampus $\mathbf{p}$, Temporally filtered sensorium $\mathbf{x}^f$, Compressed sensory observation $\mathbf{x}^c$, Sensory observation $\mathbf{x}$. Note the hippocampal dimensions, per frequency, are the multiplication of entorhinal and sensory inputs - coming from the outer product of the two representations

fequence are $[100, 100, 80, 60, 60]$ i.e. n_{s^*} multiplied by each n^f. λ and η are set to 0.9999 and 0.5 respectively. We note that, like Cueva and Wei (2018), a higher ratio of grid to band cells is observed if additional l2 regularisation (penalisation of euclidean norm) of grid cell activity is used.

As mentioned in Section 8.1, for each task we train on environments of different sizes - this means a true abstract representation must be learned and not just one that is a template map. The learned map must generalise to different sized worlds.

C.1.1 Transitive inference

The hippocampus is crucial for problems of transitive inference with animals solving novel tasks on first presentation Bunsey and Eichenbaum (1996). And so analogously we test whether TEM can learn about line structures and orderings i.e. if apple is one more than pear and pear is more than monkey, what is 2 bigger than monkey?

To do so we use fully connected graphs, and order the nodes on a line i.e. label each node from 1 to K, where K is the number of nodes in the graph (Figure C.2A). Each edge describes an action, e.g. the edge from 5 to 2 describes 'below by 3', the edge 4 to 14 describe 'higher by 10' etc. This structure and labelling of nodes and edges creates an implicit transitive hierarchy. We use lines of length $\{4,\ 5,\ 6\}$ i.e. number of states $\{4,\ 5,\ 6\ \}$).

When TEM navigates these line graph, the actions, $\mathbf{a}$, are two dimensional with the first element describing higher/lower and the second element by how much.

C.1.2 Social hierarchy

The hippocampus is known to be involved in reasoning over social hierarchies Kumaran et al. (2012), and again we want to examine whether TEM is capable of learning the abstract set of relationships that govern social hierarchies.

We consider the graph of a family tree (Figure C.2B). We limit ourselves to the case where each node has two children. We also eliminate the notion of gender - i.e. aunt/uncle is the same relationship, as is mother/father etc. Each edge corresponds to a family relationship i.e. 'grandfather of...'. We use 10

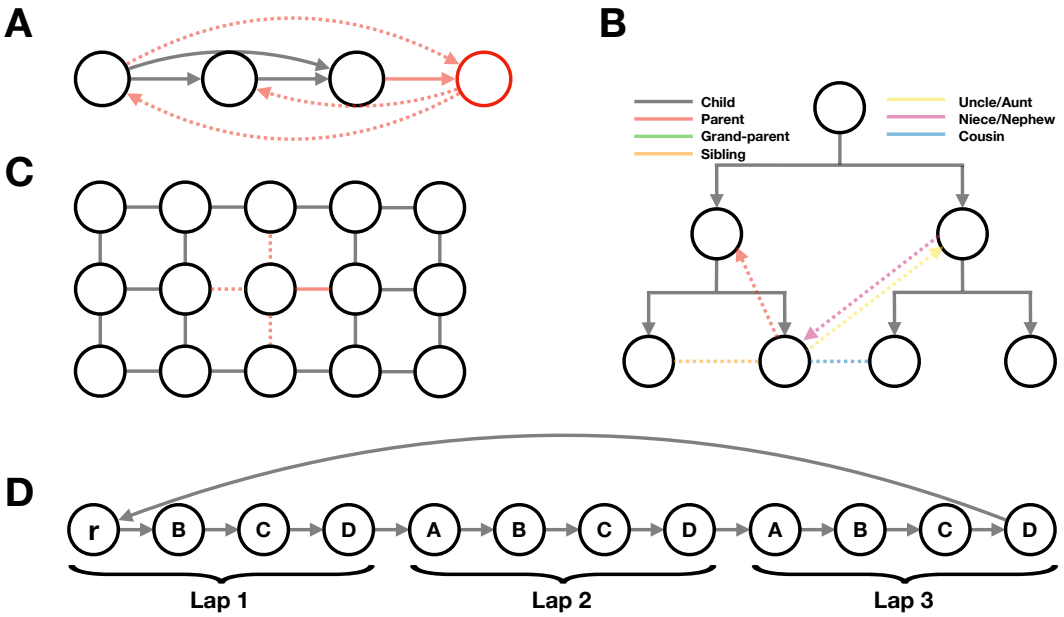

Figure C.2: Taks types in TEM. **A)** Transitive inference graph. When a new node (red) is seen to be one higher, all other (dotted) relations can be inferred i.e. 3 higher. **B)** Example graph for a social hierarchy. **C)** Example graph for 2D structure. **D)** A complex task embedded in a spatial world. This is a representation of the state space for the task in Sun et al. (2019). Each lap is of length 4 as the sensory objects (A, B, C, D) repeat every 4 nodes. There are 3 laps in total, and that defines the true state-space as a reward, r, is given every 3 laps.

types of relationships: {sibling, parent, grandparent, child 1, child 2, aunt/uncle, niece/nephew 1, niece/nephew 2, cousin 1 cousin 2}. We use {3, 4} levels of hierarchy i.e. number of states: {15, 31}.

When TEM navigates these graph, the actions, $\mathbf{a}$, are a one-hot encoding of relations such as 'child of', 'grand-parent of', 'sibling of' etc. There are 10 available actions overall.

C.1.3 2D graphs

The hippocampus and entorhinal system has produced many famous cells, most notably those that have characteristic responses to space Hafting et al. (2005); O'Keefe and Dostrovsky (1971). Thus we consider graphs with spatial properties (e.g. Figure C.2C). We consider both 4-connected and 6-connected graphs i.e. those with square or hexagonal symmetries. We use square environments of width 8-11

(number of states: {64, 81, 100, 121}), and hexagonal environments of edge width {5,6,7} (number of states: {61, 91, 127}).

We run simulations in either 6-connected graph worlds, or 4-connected graph worlds. The action is a one-hot encoding - either 4 or 6 dimensional depending on square or hexagonal worlds respectively.

Diffusive behaviour

For diffusive behaviour, the agent has a slight bias for straight paths to facilitate exploration in these larger worlds.

We show all TEM learned entorhinal cells in Figure C.3B and all hippocampal cells in Figure C.3A. We note that even in hexagonal worlds TEM sometimes learns hexagonal grid-like cells and sometimes square grid-like cells.

Non-diffusive behaviour

For non-diffusive behaviour (e.g. simulations involving object vector cells), we bias the agents transition behaviours to head towards shiny objects (for object vector cells) or spend more time near boundaries (for border cell representations).

For object-vector cell simulations, we also use an additional distribution in grid cell inference: $\mathrm{q}_\phi\left(\mathbf{g}_t \mid s_t\right)$, where s_t is an indicator saying whether the agents is at the location of a 'shiny' state. This means that the entorhinal cells can know when it is at a 'shiny' object state. From this, the network builds its own representation encoding vectors from the shiny states. We make one further change to the generative network to encourage the learning of vector representations, by not telling the generative model what action, $\mathbf{a}_t$, was taken. This encourages it to build representations of what actions will likely be taken (as governed by behaviour). Interestingly, this phenomena is observed in sleep (generative) replay - sequences of head direction activations are divorced from replay of sequences of awake activity locations Brandon et al. (2012).

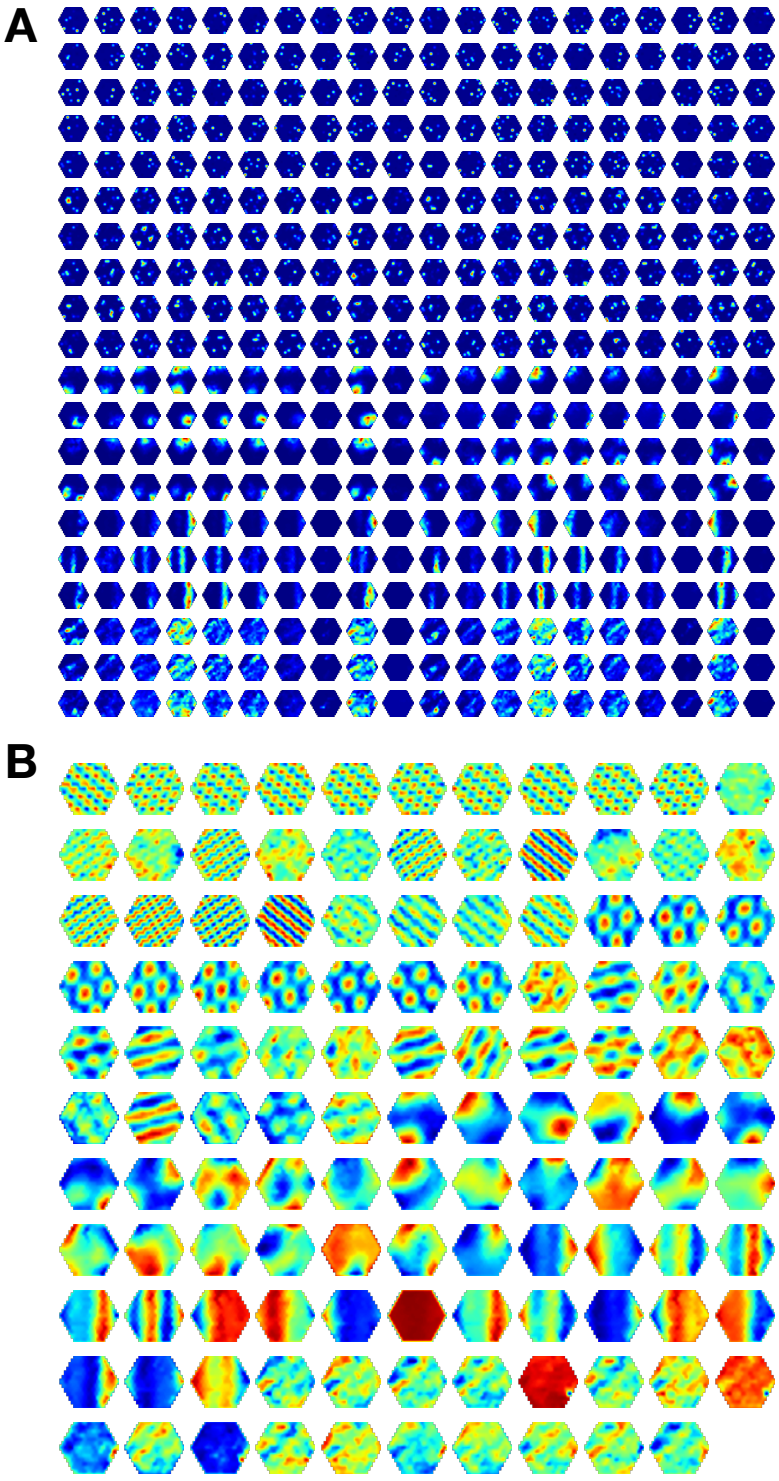

Figure C.3: Further TEM hippocampal and entorhinal cell representations. **A)** Hippocampal cells, **p**, learned by TEM during diffusive behaviour. **B)** Structural cells, **g**, learned by TEM in diffusive behaviour.

C.1.4 Complex spatial tasks

Finally we consider non-spatial tasks embedded in a spatial world. We use the task set-up from Sun et al. (2019), where rodents perform laps of a circular track. Notably they are only rewarded every 4 laps. Thus the 'true' state space of the task is 4 laps not a single lap as space would suggest. This is a non-spatial task (every 4) embedded in a spatial world (circular track). We mimic this task on a loop graph of length $l * n$, with l the lap length and n the number of laps (e.g. Figure C.2D). The sensory observations are identical on each lap, however every n laps (i.e. every whole loop of the graph), the first state is a 'reward' state - where the reward is a unique sensory observation per environment. We use $n = 3$ laps of length $l = 12$. We increase the backpropagation through time truncation to 60 so that gradient information has access to the whole state space.

We show additional examples of cells that 'chunk' as well as those that don't, from TEM's hippocampal and entorhinal layers in Figure C.4.

C.2 Variational derivation

We follow the derivation from Gemici et al. (2017). Exploiting Jensen's inequality, we can re-write as the following

$$\ln P\left(\mathbf{x}_{\leq t}\right) \geq \underset{Q\left(\mathbf{p}_{\leq t}, \mathbf{g}_{\leq t} | \mathbf{x}_{\leq t}\right)}{\mathbb{E}} \ln \frac{P\left(\mathbf{x}_{\leq t}, \mathbf{p}_{\leq t}, \mathbf{g}_{\leq t}\right)}{Q\left(\mathbf{p}_{\leq t}, \mathbf{g}_{\leq t} \mid \mathbf{x}_{\leq t}\right)} \tag{C.1}$$

Should we factorise both out generative and recognition distribution temporally as follows

$$P\left(\mathbf{x}_{\leq t}, \mathbf{p}_{\leq t}, \mathbf{g}_{\leq t}\right) = \prod_{t=1}^{T} P\left(\mathbf{x}_t \mid \mathbf{p}_t\right) P\left(\mathbf{p}_t \mid \mathbf{g}_t, M_{t-1}\right) P\left(\mathbf{g}_t \mid \mathbf{g}_{t-1}\right) \tag{C.2}$$

$$Q\left(\mathbf{p}_{\leq t}, \mathbf{g}_{\leq t} \mid \mathbf{x}_{\leq t}\right) = \prod_{t=1}^{T} Q\left(\mathbf{p}_t, \mathbf{g}_t \mid \mathbf{x}_{\leq t}, \mathbf{p}_{<t}, \mathbf{g}_{<t}\right) \tag{C.3}$$

We can then write things as the following

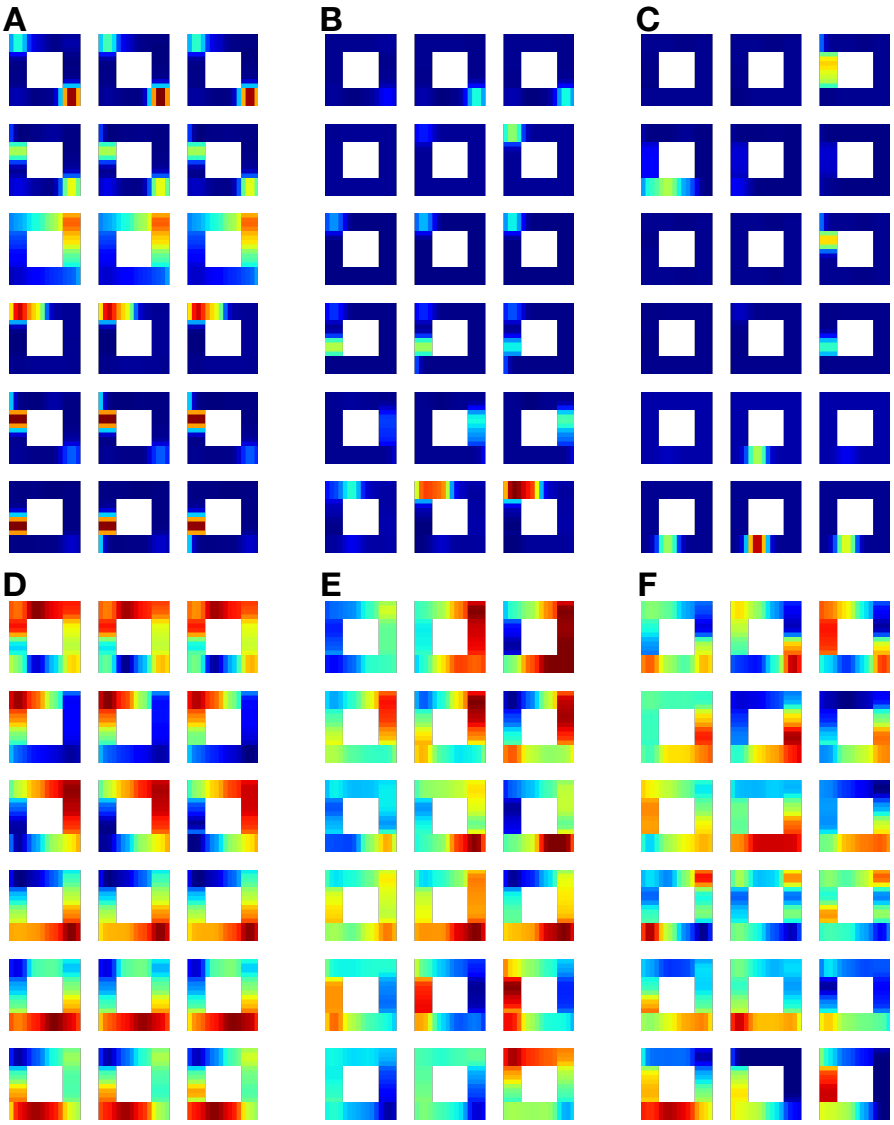

Figure C.4: Further TEM cell in complex tasks] **C-H)** Additional cells rate maps from TEM when performed in the task from Sun et al. (2019). **A-C)** TEM hippocampal cells. **A)** Cells that respond to location on lap - i.e. place cells. **B)** Cells that appear to 'count' the laps. **C)** Chunking cells. **D-F)** TEM entorhinal cells. **D)** Cells that respond to location on lap. **E)** Cells that appear to 'count' laps. **F)** Cells that chunk the laps.

$$\ln P\left(\mathbf{x}_{\leq t}\right) \geq \mathop{\mathbb{E}}_{Q\left(\mathbf{p}_{\leq t}, \mathbf{g}_{\leq t} | \mathbf{x}_{\leq t}\right)} \sum_{t=1}^{T} J_t \tag{C.4}$$

Where

$$\ln \frac{P\left(\mathbf{x}_t \mid \mathbf{x}_{<t}, \mathbf{p}_{\leq t}, \mathbf{g}_{\leq t}\right) P\left(\mathbf{p}_t \mid \mathbf{x}_{<t}, \mathbf{p}_{<t}, \mathbf{g}_{\leq t}\right) P\left(\mathbf{g}_t \mid \mathbf{x}_{<t}, \mathbf{p}_{<t}, \mathbf{g}_{<t}\right)}{Q\left(\mathbf{p}_t, \mathbf{g}_t \mid \mathbf{x}_{\leq t}, \mathbf{p}_{<t}, \mathbf{g}_{<t}\right)} = J_t \tag{C.5}$$

Thus

$$\begin{aligned}\ln P\left(\mathbf{x}_{\leq t}\right) &\geq \mathop{\mathbb{E}}_{Q\left(\mathbf{p}_{\leq t},\mathbf{g}_{\leq t}|\mathbf{x}_{\leq t}\right)} \sum_{t=1}^{T} J_t \\ &= \int Q\left(\mathbf{p}_t,\mathbf{g}_t \mid \mathbf{x}_1\right) \int ... \int Q\left(\mathbf{p}_T,\mathbf{g}_T \mid \mathbf{x}_{\leq T},\mathbf{p}_{<T},\mathbf{g}_{<T}\right) \sum_{t=1}^{T} J_t\end{aligned} \tag{C.6}$$

Since J_t is not a function of elements from the set $\{\mathbf{p}_{t+1},\mathbf{g}_{t+1},\mathbf{p}_{t+2},\mathbf{g}_{t+2}...\mathbf{p}_T,\mathbf{g}_T\}$, we can rewrite the above equation as the following:

$$\begin{aligned}&= \int Q\left(\mathbf{p}_t,\mathbf{g}_t \mid \mathbf{x}_1\right) J_1 \int Q\left(\mathbf{p}_2,\mathbf{g}_2 \mid \mathbf{x}_{\leq 2},\mathbf{p}_1,\mathbf{g}_1\right) ... \int Q\left(\mathbf{p}_T,\mathbf{g}_T \mid \mathbf{x}_{\leq T},\mathbf{p}_{<T},\mathbf{g}_{<T}\right) \\ &+ \int Q\left(\mathbf{p}_t,\mathbf{g}_t \mid \mathbf{x}_1\right) \int Q\left(\mathbf{p}_2,\mathbf{g}_2 \mid \mathbf{x}_{\leq 2},\mathbf{p}_1,\mathbf{g}_1\right) J_2 ... \int Q\left(\mathbf{p}_T,\mathbf{g}_T \mid \mathbf{x}_{\leq T},\mathbf{p}_{<T},\mathbf{g}_{<T}\right) \\ &+ ... \\ &+ \int Q\left(\mathbf{p}_t,\mathbf{g}_t \mid \mathbf{x}_1\right) \int Q\left(\mathbf{p}_2,\mathbf{g}_2 \mid \mathbf{x}_{\leq 2},\mathbf{p}_1,\mathbf{g}_1\right) ... \int Q\left(\mathbf{p}_T,\mathbf{g}_T \mid \mathbf{x}_{\leq T},\mathbf{p}_{<T},\mathbf{g}_{<T}\right) J_T\end{aligned} \tag{C.7}$$

All inner integrals integrate to 1, and so we are left with the following:

$$\mathcal{F} = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\prod_{\tau=1}^{t} Q\left(\mathbf{p}_\tau,\mathbf{g}_\tau|\mathbf{x}_{\leq\tau},\mathbf{p}_{<\tau},\mathbf{g}_{<\tau}\right)} [J_t] \tag{C.8}$$

This can all we rewritten as:

$$\begin{aligned}\mathcal{F} = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\prod_{\tau=1}^{t-1} Q\left(\mathbf{p}_\tau,\mathbf{g}_\tau|\mathbf{x}_{\leq\tau},\mathbf{p}_{<\tau},\mathbf{g}_{<\tau}\right)} [\mathop{\mathbb{E}}_{Q\left(\mathbf{p}_t,\mathbf{g}_t|\mathbf{x}_{\leq t},\mathbf{p}_{<t},\mathbf{g}_{<t}\right)} [&\ln P\left(\mathbf{x}_t \mid \mathbf{x}_{<t},\mathbf{p}_{\leq t},\mathbf{g}_{\leq t}\right) \\ &+ \ln P\left(\mathbf{p}_t \mid \mathbf{x}_{<t},\mathbf{p}_{<t},\mathbf{g}_{\leq t}\right) \\ &+ \ln P\left(\mathbf{g}_t \mid \mathbf{x}_{<t},\mathbf{p}_{<t},\mathbf{g}_{<t}\right) \\ &- \ln Q\left(\mathbf{p}_t,\mathbf{g}_t \mid \mathbf{x}_{\leq t},\mathbf{p}_{<t},\mathbf{g}_{<t}\right)]]\end{aligned} \tag{C.9}$$

We can see that this is now an per time-step cost function that we can optimise. We now use add in our choice of distributions. First out generative distribution:

$$Q\left(\mathbf{p}_t,\mathbf{g}_t \mid \mathbf{x}_{\leq t},\mathbf{p}_{<t},\mathbf{g}_{<t}\right) = Q\left(\mathbf{p}_t \mid \mathbf{x}_{\leq t},\mathbf{g}_t\right) Q\left(\mathbf{g}_t \mid \mathbf{x}_{\leq t},M_{t-1},\mathbf{g}_{t-1}\right) \tag{C.10}$$

and now our recognition distribution:

$$P\left(\mathbf{x}_{\leq t}, \mathbf{p}_{\leq t}, \mathbf{g}_{\leq t}\right) = P\left(\mathbf{x}_t \mid \mathbf{p}_t\right) P\left(\mathbf{p}_t \mid \mathbf{g}_t, M_{t-1}\right) P\left(\mathbf{g}_t \mid \mathbf{g}_{t-1}\right) \tag{C.11}$$

With M_{t-1} being the memory (stored in synaptic weights).

We can now simplify to the following:

$$\begin{aligned}
\mathcal{F} = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\prod_{\tau=1}^{t-1} Q\left(\mathbf{p}_\tau, \mathbf{g}_\tau \mid \mathbf{x}_{\leq \tau}, \mathbf{p}_{<\tau}, \mathbf{g}_{<\tau}\right)} \Big[& \\
& + \mathop{\mathbb{E}}_{Q\left(\mathbf{p}_t, \mathbf{g}_t \mid \mathbf{x}_{\leq t}, \mathbf{p}_{<t}, \mathbf{g}_{<t}\right)} \left[\ln P\left(\mathbf{x}_t \mid \mathbf{p}_t\right)\right] \\
& - \mathop{\mathbb{E}}_{Q\left(\mathbf{p}_t, \mathbf{g}_t \mid \mathbf{x}_{\leq t}, \mathbf{p}_{<t}, \mathbf{g}_{<t}\right)} D_{\mathrm{KL}}(Q\left(\mathbf{p}_t \mid \mathbf{x}_{\leq t}, \mathbf{g}_t\right) \| P\left(\mathbf{p}_t \mid \mathbf{x}_{<t}, \mathbf{g}_t\right)) \\
& - D_{\mathrm{KL}}(P\left(\mathbf{g}_t \mid \mathbf{x}_{<t}, M_{t-1}, \mathbf{g}_{t-1}\right) \| Q\left(\mathbf{g}_t \mid \mathbf{x}_{\leq t}, M_{t-1}, \mathbf{g}_{t-1}\right)) \Big] \Big]
\end{aligned} \tag{C.12}$$

Bibliography

Abbott, L. F. and Nelson, S. B. (2000). Synaptic plasticity: taming the beast. *Nature Neuroscience*, 3(S11):1178–1183.

Ackley, D., Hinton, G., and Sejnowski, T. (1985). A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1):147–169.

Advani, M. S. and Saxe, A. M. (2017). High-dimensional dynamics of generalization error in neural networks. pages 1–32.

Akrout, M., Wilson, C., Humphreys, P. C., Lillicrap, T., and Tweed, D. (2019). Using Weight Mirrors to Improve Feedback Alignment.

Anderson, J. R., Peterson, C., and Anderson, J. R. (1987). A Mean Field Theory Learning Algorithm for Neural Networks. *Complex Systems*, 1:995–1019.

Anderson, M. I. and Jeffery, K. J. (2003). Heterogeneous modulation of place cell firing by changes in context. *Journal of Neuroscience*, 23(26):8827–8835.

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., and de Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. *Nips*, (Nips):1–16.

Aronov, D., Nevers, R., and Tank, D. W. (2017). Mapping of a non-spatial dimension by the hippocampal–entorhinal circuit. *Nature*, 543(7647):719–722.

Attinger, A., Wang, B., and Keller, G. B. (2017). Visuomotor Coupling Shapes the Functional Development of Mouse Visual Cortex. *Cell*, 169(7):1291–1302.e14.

Ba, J., Hinton, G., Mnih, V., Leibo, J. Z., and Ionescu, C. (2016). Using Fast Weights to Attend to the Recent Past. *Advances in Neural Information Processing Systems 29*, 29:4331–4339.

Baldi, P. and Pineda, F. (1991). Contrastive Learning and Neural Oscillations. *Neural Computation*, 3(4):526–545.

Baldi, P. and Sadowski, P. (2016). A theory of local learning, the learning channel, and the optimality of backpropagation. *Neural Networks*, 83:51–74.

Balduzzi, D., Vanchinathan, H., and Buhmann, J. (2015). Kickback cuts backprop's red-tape: Biologically plausible credit assignment in neural networks. *Proceedings of the National Conference on Artificial Intelligence*, 1:485–491.

Banino, A., Barry, C., Uria, B., Blundell, C., Lillicrap, T., Mirowski, P., Pritzel, A., Chadwick, M. J., Degris, T., Modayil, J., Wayne, G., Soyer, H., Viola, F., Zhang, B., Goroshin, R., Rabinowitz, N., Pascanu, R., Beattie, C., Petersen, S., Sadik, A., Gaffney, S., King, H., Kavukcuoglu, K., Hassabis, D., Hadsell, R., and Kumaran, D. (2018). Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557(7705):429–433.

Baram, A. B., Muller, T. H., Whittington, J. C. R., and Behrens, T. E. (2018). Intuitive planning: global navigation through cognitive maps based on grid-like codes. *bioRxiv*, 0:421461.

Barry, C., Ginzberg, L. L., O'Keefe, J., and Burgess, N. (2012). Grid cell firing patterns signal environmental novelty by expansion. *Proceedings of the National Academy of Sciences*, 109(43):17687–17692.

Bartunov, S., Santoro, A., Hinton, G. E., Richards, B. A., Marris, L., and Lillicrap, T. P. (2018). Assessing the scalability of biologically-motivated deep learning algorithms and architectures. *Advances in Neural Information Processing Systems*, 2018-Decem(NeurIPS):9368–9378.

Bastos, A. M., Usrey, W. M., Adams, R. A., Mangun, G. R., Fries, P., and Friston, K. J. (2012). Canonical Microcircuits for Predictive Coding. *Neuron*, 76(4):695–711.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gulcehre, C., Song, F., Ballard, A., Gilmer, J., Dahl, G., Vaswani, A., Allen, K., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., and Pascanu, R. (2018). Relational inductive biases, deep learning, and graph networks. pages 1–38.

Bayes, T. (1763). An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, 53:370–418.

Beck, J. M., Ma, W. J., Kiani, R., Hanks, T., Churchland, A. K., Roitman, J., Shadlen, M. N., Latham, P. E., and Pouget, A. (2008). Probabilistic Population Codes for Bayesian Decision Making. *Neuron*, 60(6):1142–1152.

Behrens, T. E. (2004). MRI diffusion tractography: Methods and Applications.

Behrens, T. E. J., Muller, T. H., Whittington, J. C. R., Mark, S., Baram, A. B., Stachenfeld, K. L., and Kurth-nelson, Z. (2018). What Is a Cognitive Map? Organizing Knowledge for Flexible Behavior. *Neuron*, 100(2):490–509.

Behrens, T. E. J., Woolrich, M. W., Walton, M. E., and Rushworth, M. F. S. (2007). Learning the value of information in an uncertain world. *Nature neuroscience*, 10(9):1214–21.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences of the United States of America*, 116(32):15849–15854.

Bell, A. J. and Sejnowski, T. J. (1997). The "independent components" of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338.

Bellec, G., Scherr, F., Hajek, E., Salaj, D., Legenstein, R., and Maass, W. (2019). Biologically inspired alternatives to backpropagation through time for learning in recurrent neural nets. pages 1–37.

Bengio, Y. (2014). How Auto-Encoders Could Provide Credit Assignment in Deep Networks via Target Propagation. pages 1–34.

Bengio, Y. (2017). The Consciousness Prior. (1):1–4.

Bengio, Y., Courville, A., and Vincent, P. (2012). Representation learning: A review and new perspectives. *arXiv preprint arXiv ...*, (1993):1–34.

Bi, G. Q. and Poo, M. M. (1998). Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of Neuroscience*, 18(24):10464–10472.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bliss, T. and Collingridge, G. (1993). A synaptic model of memory: LTP in the hippocampus. *Nature*, 361:31–39.

Bogacz, R. (2017). A tutorial on the free-energy framework for modelling perception and learning. *Journal of Mathematical Psychology*, 76:198–211.

Bostock, E., Muller, R. U., and Kubie, J. L. (1991). Experience-dependent modifications of hippocampal place cell firing. *Hippocampus*, 1(2):193–205.

Bowers, J. S. (2017). Parallel Distributed Processing Theory in the Age of Deep Networks. *Trends in Cognitive Sciences*, 21(12):950–961.

Brading, K. and Castellani, E. (2003). *Symmetries in Physics: Philosophical Reflections*. Cambridge University Press.

Brandon, M. P., Bogaard, A. R., Andrews, C. M., and Hasselmo, M. E. (2012). Head direction cells in the postsubiculum do not show replay of prior waking sequences during sleep. *Hippocampus*, 22(3):604–618.

Brun, V. H., Solstad, T., Kjelstrup, K. B., Fyhn, M., Witter, M. P., Moser, E. I., and Moser, M. B. (2008). Progressive increase in grid scale from dorsal to ventral medial entorhinal cortex. *Hippocampus*, 18(12):1200–1212.

Buckmaster, C. A., Eichenbaum, H., Amaral, D. G., Suzuki, W. A., and Rapp, P. R. (2004). Entorhinal Cortex Lesions Disrupt the Relational Organization of Memory in Monkeys. *Journal of Neuroscience*, 24(44):9811–9825.

Bunsey, M. and Eichenbaum, H. (1996). Conservation of hippocampal memory function in rats and humans. *Nature*, 379(6562):255–257.

Burak, Y. and Fiete, I. R. (2009). Accurate Path Integration in Continuous Attractor Network Models of Grid Cells. *PLoS Computational Biology*, 5(2):e1000291.

Burt, C. (1911). Experimental tests of higher mental processes and their relation to general intelligence.

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

Chang, L. and Tsao, D. Y. (2017). The Code for Facial Identity in the Primate Brain. *Cell*, 169(6):1013–1028.e14.

Chen, G., King, J. A., Lu, Y., Cacucci, F., and Burgess, N. (2018a). Spatial cell firing during virtual navigation of open arenas by head-restrained mice. *eLife*, 7.

Chen, G., Lu, Y., King, J. A., Cacucci, F., and Burgess, N. (2019). Differential influences of environment and self-motion on place and grid cell firing. *Nature Communications*, 10(1):1–11.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. (2018b). Neural Ordinary Differential Equations. (Nips):1–18.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). InfoGAN: Interpretable Representation Learning. *30th Conference on Neural Information Processing Systems (NIPS 2016)*, (Nips):2172–2180.

Clark, A. (2013). Whatever next? Predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences*, 36(3):181–204.

Cohen, N. J. and Eichenbaum, H. (1993). *Memory, amnesia, and the hippocampal system*. MIT Press.

Cohen, N. J. and Squire, L. R. (1980). Preserved learning and retention of pattern-analyzing skill in amnesia: Dissociation of knowing how and knowing that. *Science*, 210(4466):207–210.

Constantinescu, A. O., OReilly, J. X., Behrens, T. E. J., O'Reilly, J. X., and Behrens, T. E. J. (2016). Organizing conceptual knowledge in humans with a gridlike code. *Science*, 352(6292):1464–1468.

Crick, F. (1989). The recent excitement about neural networks.

Cueva, C. J. and Wei, X.-X. (2018). Emergence of grid-like representations by training recurrent neural networks to perform spatial localization. *International Conference on Learning Representations*, 0.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314.

Danjo, T., Toyoizumi, T., and Fujisawa, S. (2018). Spatial representations of self and other in the hippocampus. *Science*, 359(6372):213–218.

Dayan, P. (1993). Improving Generalization for Temporal Difference Learning: The Successor Representation. *Neural Computation*, 5(4):613–624.

Dayan, P. and Abbott, L. F. (2001). *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. MIT Press.

Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. (1995). The Helmholtz machine. *Neural computation*, 7(5):889–904.

de Lange, F. P., Heilbron, M., and Kok, P. (2018). How Do Expectations Shape Perception? *Trends in Cognitive Sciences*, 22(9):764–779.

de Sa, V. R. and Ballard, D. H. (1998). Perceptual Learning From Cross-Modal Feedback. In *Psychology of Learning and Motivation - Advances in Research and Theory*, volume 36, pages 309–351.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data Via the EM Algorithm . *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

Deshmukh, S. S. and Knierim, J. J. (2011). Representation of Non-Spatial and Spatial Information in the Lateral Entorhinal Cortex. *Frontiers in Behavioral Neuroscience*, 5(OCTOBER).

Deshmukh, S. S. and Knierim, J. J. (2013). Influence of local objects on hippocampal representations: Landmark vectors and memory. *Hippocampus*, 23(4):253–67.

Doeller, C. F., Barry, C., and Burgess, N. (2010). Evidence for grid cells in a human memory network. *Nature*, 463(7281):657–61.

Dordek, Y., Soudry, D., Meir, R., and Derdikman, D. (2016). Extracting grid cell characteristics from place cell inputs using non-negative principal component analysis. *eLife*, 5(MARCH2016):1–36.

Dusek, J. A. and Eichenbaum, H. (1997). The hippocampus and memory for orderly stimulus relations. *Proceedings of the National Academy of Sciences*, 94(13):7109–7114.

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

Ernst, M. O. and Banks, M. S. (2002). Humans integrate visual and haptic information in a statistically optimal fashion. *Nature*, 415(6870):429–433.

Etienne, A. S. and Jeffery, K. J. (2004). Path integration in mammals. *Hippocampus*, 14(2):180–192.

Ferster, C. B. and Skinner, B. F. (1957). *Schedules of reinforcement*. Appleton-Century-Crofts, East Norwalk.

Feynman, R. (1972). *Statistical Mechanics; A Set of Lectures*. Massachusetts.

Field, D. J. (1987). Relations between the statistics of natural images and the response properties of cortical cells. *Journal of the Optical Society of America A*, 4(12):2379.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks.

Foerster, J. N., Assael, Y. M., de Freitas, N., and Whiteson, S. (2016). Learning to Communicate with Deep Multi-Agent Reinforcement Learning. pages 1–13.

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

Friston, K. (2003). Learning and inference in the brain. *Neural Networks*, 16(9):1325–1352.

Friston, K. (2005). A theory of cortical responses. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 360(1456):815–36.

Friston, K. (2010). The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11(2):127–138.

Friston, K. (2013). Life as we know it. *Journal of the Royal Society, Interface / the Royal Society*, 10(86):20130475.

Friston, K. and Herreros, I. (2016). Active Inference and Learning in the Cerebellum. *Neural Computation*, 28(9):1812–1839.

Friston, K. J., Daunizeau, J., and Kiebel, S. J. (2009). Reinforcement learning or active inference? *PLoS ONE*, 4(7).

Friston, K. J., Harrison, L., and Penny, W. (2003). Dynamic causal modelling. *NeuroImage*, 19(4):1273–1302.

Fuhs, M. C. and Touretzky, D. S. (2006). A spin glass model of path integration in rat medial entorhinal cortex. *Journal of Neuroscience*, 26(16):4266–4276.

Fyhn, M., Hafting, T., Treves, A., Moser, M. B., and Moser, E. I. (2007). Hippocampal remapping and grid realignment in entorhinal cortex. *Nature*, 446(7132):190–194.

Gallese, V., Fadiga, L., Fogassi, L., and Rizzolatti, G. (1996). Action recognition in the premotor cortex. *Brain*, 119(2):593–609.

Garvert, M. M., Dolan, R. J., and Behrens, T. E. E. (2017). A map of abstract relational knowledge in the human hippocampal–entorhinal cortex. *eLife*, 6:1–20.

Gauthier, J. L. and Tank, D. W. (2018). A Dedicated Population for Reward Coding in the Hippocampus. *Neuron*, 99(1):179–193.e7.

Gell-Mann, M. (1962). Symmetries of Baryons and Mesons. *Physical Review*, 125(3):1067–1084.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural Networks and the Bias/Variance Dilemma. *Neural Computation*, 4(1):1–58.

Gemici, M., Hung, C.-C., Santoro, A., Wayne, G., Mohamed, S., Rezende, D. J., Amos, D., and Lillicrap, T. (2017). Generative Temporal Models with Memory. *arXiv preprint arXiv:1702.04649*, 0.

Gershman, S. J. (2019). What does the free energy principle tell us about the brain? pages 1–10.

Gershman, S. J., Horvitz, E. J., and Tenenbaum, J. B. (2015). Computational rationality: A converging paradigm for intelligence in brains, minds, and machines. *Science*, 349(6245):273–278.

Gibson, J. J. (1977). The theory of affordances.

Gilboa, A., Sekeres, M., Moscovitch, M., and Winocur, G. (2014). Higher-order conditioning is impaired by hippocampal lesions. *Current Biology*, 24(18):2202–2207.

Goodman, N. D. (1955). *Fact Fiction and Forecast*.

Griffiths, T. L., Lieder, F., and Goodman, N. D. (2015). Rational use of cognitive resources: Levels of analysis between the computational and the algorithmic. *Topics in Cognitive Science*, 7(2):217–229.

Griffiths, T. L. and Tenenbaum, J. B. (2006). Optimal Predictions in Everyday Cognition. *Psychological Science*, 17(9):767–773.

Grossberg, S. (1987). Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 11(1):23–63.

Guanella, A. and Verschure, P. F. M. J. (2006). A Model of Grid Cells Based on a Path Integration Mechanism. In *Artificial Neural Networks – ICANN 2006*, pages 740–749. Springer Berlin Heidelberg.

Guergiuev, J., Lillicrap, T. P., Richards, B. A., Guerguiev, J., Lillicrap, T. P., Richards, B. A., Guergiuev, J., Lillicrap, T. P., and Richards, B. A. (2017). Towards deep learning with segregated dendrites. *eLife*, 6:1–41.

Gustafson, N. J. and Daw, N. D. (2011). Grid Cells, Place Cells, and Geodesic Generalization for Spatial Reinforcement Learning. *PLoS Computational Biology*, 7(10):e1002235.

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

Harlow, H. F. (1949). The formation of learning sets. *Psychological Review*, 56(1):51–65.

Hassabis, D., Kumaran, D., Vann, S. D., and Maguire, E. A. (2007). Patients with hippocampal amnesia cannot imagine new experiences. *Proceedings of the National Academy of Sciences*, 104(5):1726–1731.

Hebb, D. O. (1949). The Organization of Behavior; A Neuropsychological Theory.

Higgins, I., Amos, D., Pfau, D., Racaniere, S., Matthey, L., Rezende, D., and Lerchner, A. (2018). Towards a Definition of Disentangled Representations. pages 1–29.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017a). β-VAE: Learning basic visual concepts with a constrained variational framework. *International Conference on Learning Representations*, 0.

Higgins, I., Pal, A., Rusu, A. A., Matthey, L., Burgess, C. P., Pritzel, A., Botvinick, M., Blundell, C., and Lerchner, A. (2017b). DARLA: Improving Zero-Shot Transfer in Reinforcement Learning.

Hillis, J. M., Watt, S. J., Landy, M. S., and Banks, M. S. (2004). Slant from texture and disparity cues: Optimal cue combination. *Journal of Vision*, 4(12):967–992.

Hinton, G., Dayan, P., Frey, B., and Neal, R. (1995). The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161.

Hinton, G. E., Osindero, S., and Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–54.

Hinton, G. E. and van Camp, D. (1993). Keeping neural networks simple by minimizing the description length of the weights. pages 5–13.

Hodgkin, A. L. and Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500–544.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347.

Hopfield, J. J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences*, 81(10):3088–3092.

Høydal, Ø. A., Skytøen, E. R., Andersson, S. O., Moser, M.-B., and Moser, E. I. (2019). Object-vector coding in the medial entorhinal cortex. *Nature*, 568(7752):400–404.

Høydal, Ø. A., Skytøen, E. R., Moser, M. B., and Moser, E. I. (2018). Object-vector cells in the medial entorhinal cortex. *bioRxiv*, 0.

Hoyer, P. O. and Hyvärinen, A. (2003). Interpreting neural response variability as monte carlo sampling of the posterior. *Advances in neural information processing systems*, (1):293–300.

Hubel, D. H. and Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, 148(3):574–591.

Hyvarinen, A. (1999). Regression using independent component analysis, and its connection to multi-layer perceptrons. In *9th International Conference on Artificial Neural Networks: ICANN '99*, volume 1999, pages 491–496. IEE.

Jacobs, R. A. and Fine, I. (1999). Experience-dependent integration of texture and motion cues to depth. *Vision Research*, 39(24):4062–4075.

Jolliffe, I. T. (1986). Principal Component Analysis and Factor Analysis. pages 115–128.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). Introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233.

Julian, J. B., Keinath, A. T., Frazzetta, G., and Epstein, R. A. (2018). Human entorhinal cortex represents visual space using a boundary-anchored grid. *Nature Neuroscience*, 21(2):191–194.

Jung, M. W., Wiener, S. I., and McNaughton, B. L. (1994). Comparison of spatial firing characteristics of units in dorsal and ventral hippocampus of the rat. *The Journal of Neuroscience*, 14(12):7347–7356.

Kemp, C. and Tenenbaum, J. B. (2008). The discovery of structural form. *Proceedings of the National Academy of Sciences*, 105(31):10687–10692.

Ketz, N., Morkonda, S. G., and O'Reilly, R. C. (2013). Theta Coordinated Error-Driven Learning in the Hippocampus. *PLoS Computational Biology*, 9(6).

Killian, N. J. and Buffalo, E. A. (2018). Grid cells map the visual world. *Nature Neuroscience*, 21(2):161–162.

Kim, S., Sapiurka, M., Clark, R. E., and Squire, L. R. (2013). Contrasting effects on path integration after hippocampal damage in humans and rats. *Proceedings of the National Academy of Sciences of the United States of America*, 110(12):4732–4737.

Kingma, D. P. and Ba, J. L. (2014). Adam: A Method for Stochastic Optimization. *arXiv preprint arxiv:1412.6980*, 0.

Kingma, D. P. and Welling, M. (2013). Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*, 0.

Kjelstrup, K. B., Solstad, T., Brun, V. H., Hafting, T., Leutgeb, S., Witter, M. P., Moser, E. I., and Moser, M.-B. (2008). Finite Scale of Spatial Represenation in the Hippocampus. *Science*, 321(July):140 – 143.

Knill, D. C. (2003). Mixture models and the probabilistic structure of depth cues. *Vision Research*, 43(7):831–854.

Knill, D. C. and Pouget, A. (2004). The Bayesian brain: The role of uncertainty in neural coding and computation. *Trends in Neurosciences*, 27(12):712–719.

Knill, D. C. and Saunders, J. A. (2003). Do humans optimally integrate stereo and texture information for judgments of surface slant? *Vision Research*, 43(24):2539–2558.

Kok, P. and Lange, F. P. D. (2015). Predictive Coding in Sensory Cortex. pages 221–244.

Komorowski, R. W., Manns, J. R., and Eichenbaum, H. (2009). Robust Conjunctive Item-Place Coding by Hippocampal Neurons Parallels Learning What Happens Where. *Journal of Neuroscience*, 29(31):9918–9929.

Körding, K. P. and König, P. (2001). Neurons with Two Sites of Synaptic Integration Learn Invariant Representations. *Neural Computation*, 13(12):2823–2849.

Kording, K. P. and Konig, P. (2001). Supervised and Unsupervised Learning with Two Sites of Synaptic Integration. *Journal of Computational Neuroscience*, 11:207–215.

Körding, K. P. and Wolpert, D. M. (2004). Bayesian integration in sensorimotor learning. *Nature*, 427(6971):244–247.

Krotov, D. and Hopfield, J. J. (2019). Unsupervised learning by competing hidden units. *Proceedings of the National Academy of Sciences of the United States of America*, 116(16):7723–7731.

Krupic, J., Burgess, N., and O'Keefe, J. (2012). Neural Representations of Location Composed of Spatially Periodic Bands. *Science*, 337(6096):853–857.

Kubota, Y. (2014). Untangling GABAergic wiring in the cortical microcircuit. *Current Opinion in Neurobiology*, 26:7–14.

Kullback, S. and Leibler, R. A. (1951). On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86.

Kumaran, D., Melo, H. L., and Duzel, E. (2012). The Emergence and Representation of Knowledge about Social and Nonsocial Hierarchies. *Neuron*, 76(3):653–666.

Kuśmierz, Ł., Isomura, T., and Toyoizumi, T. (2017). Learning with three factors: modulating Hebbian plasticity with errors. *Current Opinion in Neurobiology*, 46:170–177.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Laplace, P. S. (1774). Memoir on the Probability of the Causes of Events. *Statistical Science*, 3:364–378.

Larkum, M. E., Zhu, J. J., and Sakmann, B. (1999). A new cellular mechanism for coupling inputs arriving at different cortical layers. *Nature*, 398(6725):338–341.

Lecun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *proc. OF THE IEEE*.

Lee, D. H., Zhang, S., Fischer, A., and Bengio, Y. (2015). Difference target propagation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9284:498–515.

Lee, T. S. and Mumford, D. (2003). Hierarchical Bayesian inference in the visual cortex. *Journal of the Optical Society of America A*, 20(7):1434.

Leinweber, M., Ward, D. R., Sobczak, J. M., Attinger, A., and Keller, G. B. (2017). A Sensorimotor Circuit in Mouse Cortex for Visual Flow Predictions. *Neuron*, 95(6):1420–1432.e5.

Leutgeb, S., Leutgeb, J. K., Barnes, C. A., Moser, E. I., McNaughton, B. L., and Moser, M.-B. (2005). Independent Codes for Spatial and Episodic Memory in Hippocampal Neuronal Ensembles. (July):619–624.

Lever, C., Burton, S., Jeewajee, A., O'Keefe, J., and Burgess, N. (2009). Boundary vector cells in the subiculum of the hippocampal formation. *Journal of Neuroscience*, 29(31):9771–9777.

Liao, Q., Leibo, J. Z., and Poggio, T. (2016). How important is weight symmetry in backpropagation? *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, pages 1837–1844.

Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. (2016). Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7:1–10.

Loewi, O. (1921). Über humorale übertragbarkeit der Herznervenwirkung. *Pflügers Archiv für die Gesamte Physiologie des Menschen und der Tiere*, 189(1):239–242.

Ma, W. J., Beck, J. M., Latham, P. E., and Pouget, A. (2006). Bayesian inference with probabilistic population codes. *Nature Neuroscience*, 9(11):1432–1438.

MacDonald, C. J., Lepage, K. Q., Eden, U. T., and Eichenbaum, H. (2011). Hippocampal "time cells" bridge the gap in memory for discontiguous events. *Neuron*, 71(4):737–749.

MacKay, D. J. C. (1992). *Bayesian methods for adaptive models*. PhD thesis, California Institute of Technology.

MacKay, D. J. C. (2003). *Information theory, inference and learning algorithms*, volume 1. Cambridge University Press.

Manns, J. R. and Eichenbaum, H. (2006). Evolution of declarative memory. *Hippocampus*, 16(9):795–808.

Marblestone, A. H., Wayne, G., and Kording, K. P. (2016). Toward an integration of deep learning and neuroscience. *Frontiers in Computational Neuroscience*, 10(SEP):1–41.

Mayor, M. and Queloz, D. (1995). A jupiter-mass companion to a solar-type star. *Nature*, 378(6555):355–359.

Mazzoni, P., Andersen, R. a., and Jordan, M. I. (1991). A more biologically plausible learning rule for neural networks. *Proceedings of the National Academy of Sciences*, 88(10):4433–4437.

McClelland, J. L., McNaughton, B. L., and O'Reilly, R. C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102(3):419–457.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133.

Mcgonigle, B. O. and Chalmers, M. (1977). Are monkeys logical? *Nature*, 267(5613):694–696.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. *Advances In Neural Information Processing Systems*, pages 1389–1399.

Mitchell, T. M. (1980). The Need for Biases in Learning Generalizations. *Rutgers CS tech report CBM-TR-117*, (May).

Mittelstaedt, M. L. and Mittelstaedt, H. (1980). Homing by path integration in a mammal. *Naturwissenschaften*, 67(11):566–567.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Marc G, B., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Momennejad, I., Russek, E. M., Cheong, J. H., Botvinick, M. M., Daw, N. D., and Gershman, S. J. (2017). The successor representation in human reinforcement learning. *Nature Human Behaviour*, 1(9):680–692.

Mostafa, H., Ramesh, V., and Cauwenberghs, G. (2018). Deep supervised learning using local errors. *Frontiers in Neuroscience*, 12(AUG).

Murphy, K. (2014). Machine Learning, a Probabilistic Perspective. *CHANCE*, 27(2):62–63.

Nakazawa, K., Quirk, M. C., Chitwood, R. A., Watanabe, M., Yeckel, M. F., Sun, L. D., Kato, A., Carr, C. A., Johnston, D., Wilson, M. A., and Tonegawa, S. (2002). Requirement for hippocampal CA3 NMDA receptors in associative memory recall. *Science*, 297(5579):211–218.

Nau, M., Navarro Schröder, T., Bellmund, J. L. S., and Doeller, C. F. (2018). Hexadirectional coding of visual space in human entorhinal cortex. *Nature Neuroscience*, 21(188).

Neal, R. M. and Hinton, G. E. (1998). A View of the Em Algorithm that Justifies Incremental, Sparse, and other Variants. In *Learning in Graphical Models*, pages 355–368. Springer Netherlands, Dordrecht.

Neunuebel, J. P., Yoganarasimha, D., Rao, G., and Knierim, J. J. (2013). Conflicts between local and global spatial frameworks dissociate neural representations of the lateral and medial entorhinal cortex. *Journal of Neuroscience*, 33(22):9246–9258.

Nieder, A. (2012). Supramodal numerosity selectivity of neurons in primate prefrontal and posterior parietal cortices. *Proceedings of the National Academy of Sciences of the United States of America*, 109(29):11860–11865.

Noether, E. (1920). The finiteness theorem for invariants of a finite group. *Mathematische Annalen*, 77:89–92.

O'Keefe, J. and Dostrovsky, J. (1971). The hippocampus as a spatial map. Preliminary evidence from unit activity in the freely-moving rat. *Brain Research*, 34(1):171–175.

O'Keefe, J. and Nadel, L. (1978). *The Hippocampus as a Cognitive Map*.

Ólafsdóttir, H. F., Barry, C., Saleem, A. B., Hassabis, D., and Spiers, H. J. (2015). Hippocampal place cells construct reward related sequences through unexplored space. *eLife*, 4(JUNE2015):1–17.

Olshausen, B. A. and Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images.

Omer, D. B., Maimon, S. R., Las, L., and Ulanovsky, N. (2018). Social place-cells in the bat hippocampus. *Science*, 359(6372):218–224.

O'Neill, J., Boccara, C., Stella, F., Schoenenberger, P., and Csicsvari, J. (2017). Superficial layers of the medial entorhinal cortex replay independently of the hippocampus. *Science*, 355(6321):184–188.

O'Reilly, R. C. (1996). Biologically Plausible Error-Driven Learning Using Local Activation Differences: The Generalized Recirculation Algorithm. *Neural Computation*, 8(5):895–938.

O'Reilly, R. C. and Munakata, Y. (2000). *Computational explorations in cognitive neuroscience.* MIT Press., Cambridge, MA.

Pavlov, I. P. (1927). Conditioned reflexes.

Pike, F. G., Meredith, R. M., Olding, A. W. A., and Paulsen, O. (1999). Postsynaptic bursting is essential for 'Hebbian' induction of associative long-term potentiation at excitatory synapses in rat hippocampus. *The Journal of Physiology*, 518(2):571–576.

Pineda, F. J. (1987). Generalization of back-propagation to recurrent neural networks. *Physical Review Letters*, 59(19):2229–2232.

Pouget, A., Beck, J. M., Ma, W. J., and Latham, P. E. (2013). Probabilistic brains: Knowns and unknowns. *Nature Neuroscience*, 16(9):1170–1178.

Purcell, E. M. (1977). Life at low Reynolds number. *American Journal of Physics*, 45(1):3–11.

Rahnev, D. and Denison, R. N. (2018). Suboptimality in Perceptual Decision Making. *Behavioral and Brain Sciences*, pages 1–107.

Ranganath, R., Gerrish, S., and Blei, D. M. (2014). Black box variational inference. *Journal of Machine Learning Research*, 33:814–822.

Rao, R. P. and Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2:79–87.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *arXiv preprint arXiv:1401.4082*, 0.

Richards, B. A. and Lillicrap, T. P. (2019). Dendritic solutions to the credit assignment problem. *Current Opinion in Neurobiology*, 54(September):28–36.

Roelfsema, P. R. and Holtmaat, A. (2018). Control of synaptic plasticity in deep cortical networks. *Nature Reviews Neuroscience*, 19(3):166–180.

Roitman, J. D. and Shadlen, M. N. (2002). Response of neurons in the lateral intraparietal area during a combined visual discrimination reaction time task. *Journal of Neuroscience*, 22(21):9475–9489.

Rumelhart, D., Hinton, G., and Williams, R. (1988). Learning Internal Representations by Error Propagation. In *Readings in Cognitive Science*, pages 399–421. Elsevier.

Rumelhart, D. E., Hinton, G. E., and McClelland, J. L. (1986). A General Framework for Parallel Distributed Processing. *Parallel Distributed Processing: Explorations in the microstructure of cognition. Volume I*, (1982):45–76.

Sacramento, J., Bengio, Y., Costa, R. P., and Senn, W. (2018). Dendritic cortical microcircuits approximate the backpropagation algorithm. *Advances in Neural Information Processing Systems*, 2018-Decem(NeurIPS):8721–8732.

Sarel, A., Finkelstein, A., Las, L., and Ulanovsky, N. (2017). Vectorial representation of spatial goals in the hippocampus of bats. *Science*, 355(6321):176–180.

Scellier, B. and Bengio, Y. (2017a). Equilibrium Propagation: Bridging the Gap Between Energy-Based Models and Backpropagation. 11(May).

Scellier, B. and Bengio, Y. (2017b). Equivalence of Equilibrium Propagation and Recurrent Backpropagation.

Schiess, M., Urbanczik, R., and Senn, W. (2016). Somato-dendritic Synaptic Plasticity and Error-backpropagation in Active Dendrites. *PLoS Computational Biology*, 12(2):1–18.

Schmidhuber, J. (1992). Learning Factorial Codes by Predictability Minimization. *Neural Computation*, 7.

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

Scoville, W. B. and Milner, B. (1957). Loss of recent memory after bilateral hippocampal lesions. *J.Neurol. Neurosurg.Psychiat.*, 20:11–21.

Seung, H. (2003). Learning in Spiking Neural Networks by Reinforcement of Stochastic Synaptic Transmission. *Neuron*, 40(6):1063–1073.

Sherrington, C. S. (1907). The Integrative Action of the Nervous System. *JAMA: The Journal of the American Medical Association*, XLVIII(12):1055.

Silberberg, G. and Markram, H. (2007). Disynaptic Inhibition between Neocortical Pyramidal Cells Mediated by Martinotti Cells. *Neuron*, 53(5):735–746.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., Van Den Driessche, G., Graepel, T., and Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359.

Singer, Y., Teramoto, Y., Willmore, B. D., Schnupp, J. W., King, A. J., and Harper, N. S. (2018). Sensory cortex is optimized for prediction of future input. *eLife*, 7:1–31.

Skinner, B. F. (1938). *The behavior of organisms*. Appleton-Century.

Solstad, T., Boccara, C. N., Kropff, E., Moser, M.-B., and Moser, E. I. (2008). Representation of Geometric Borders in the Entorhinal Cortex. *Science*, 322(5909):1865–1868.

Song, S., Sjöström, P. J., Reigl, M., Nelson, S., and Chklovskii, D. B. (2005). Highly nonrandom features of synaptic connectivity in local cortical circuits. *PLoS Biology*, 3(3):0507–0519.

Sorscher, B., Mel, G. C., Ganguli, S., and Ocko, S. A. (2019). A unified theory for the origin of grid cells through the lens of pattern formation. *Advances in Neural Information Processing Systems 32*, 32(NeurIPS):10003–10013.

Sporea, I. and Grüning, A. (2013). Supervised Learning in Multilayer Spiking Neural Networks. *Neural Computation*, 25(2):473–509.

Sreenivasan, S. and Fiete, I. (2011). Grid cells generate an analog error-correcting code for singularly precise neural computation. *Nature Neuroscience*, 14(10):1330–1337.

Srivastava, N. and Salakhutdinov, R. (2014). Multimodal learning with Deep Boltzmann Machines. *Journal of Machine Learning Research*, 15:2949–2980.

Stachenfeld, K. L. K. L., Botvinick, M. M., and Gershman, S. J. (2017). The hippocampus as a predictive map. *Nature Neuroscience*, 20(11):1643–1653.

Stemmler, M., Mathis, A., and Herz, A. V. M. (2015). Connecting multiple spatial scales to decode the population activity of grid cells. *Science Advances*, 1(11):e1500816.

Stensola, H., Stensola, T., Solstad, T., FrØland, K., Moser, M.-B. B., and Moser, E. I. (2012). The entorhinal grid map is discretized. *Nature*, 492(7427):72–78.

Strange, B. A., Witter, M. P., Lein, E. S., and Moser, E. I. (2014). Functional organization of the hippocampal longitudinal axis. *Nature Reviews Neuroscience*, 15(10):655–669.

Summerfield, C. and de Lange, F. P. (2014). Expectation in perceptual decision making: neural and computational mechanisms. *Nature Reviews Neuroscience*, 15(11):745–756.

Summerfield, C., Egner, T., Greene, M., Koechlin, E., Mangels, J., and Hirsch, J. (2006). Predictive Codes for Forthcoming Perception in the Frontal Cortex. *Science*, 314(5803):1311–1314.

Summerfield, C., Luyckx, F., and Sheahan, H. (2019). Structure Learning and the Parietal Cortex. *bioRxiv*, 0.

Summerfield, C., Trittschuh, E. H., Monti, J. M., Mesulam, M.-m. M., and Egner, T. (2008). Neural repetition suppression reflects fulfilled perceptual expectations. *Nature neuroscience*, 11(9):1004–1006.

Summerfield, C. and Tsetsos, K. (2015). Do humans make good decisions? *Trends in Cognitive Sciences*, 19(1):27–34.

Sun, C., Yang, W., Martin, J., and Tonegawa, S. (2019). CA1 pyramidal cells organize an episode by segmented and ordered events. *bioRxiv*, 0:565689.

Taube, J., Muller, R., and Ranck, J. (1990). Head-direction cells recorded from the postsubiculum in freely moving rats. I. Description and quantitative analysis. *The Journal of Neuroscience*, 10(2):420–435.

Tavares, R. M., Mendelsohn, A., Grossman, Y., Williams, C. H., Shapiro, M., Trope, Y., and Schiller, D. (2015). A Map for Social Navigation in the Human Brain. *Neuron*, 87(1):231–243.

Tenenbaum, J. B., Kemp, C., Griffiths, T. L., and Goodman, N. D. (2011). How to grow a mind: Statistics, structure, and abstraction. *Science*, 331(6022):1279–1285.

Thorndike, E. L. (1898). Animal intelligence: An experimental study of the associative processes in animals. *The Psychological Review: Monograph Supplements*, 2(4):i–109.

Tikhonov, A. N. (1943). On the stability of inverse problems. *Doklady Akademii Nauk SSSR*, 39(5):195–198.

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

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

Tolman, E. C., Ritchie, B. F., and Kalish, D. (1946). Studies in spatial learning. I. Orientation and the short-cut. *Journal of Experimental Psychology*, 36(1):13–24.

Treichler, F. R. and Van Tilburg, D. (1996). Concurrent conditional discrimination tests of transitive inference by macaque monkeys: list linking. *Journal of experimental psychology. Animal behavior processes*, 22(1):105–17.

Tsao, A., Sugar, J., Lu, L., Wang, C., Knierim, J. J., Moser, M.-B., and Moser, E. I. (2018). Integrating time from experience in the lateral entorhinal cortex. *Nature*, 561(7721):57–62.

Unnikrishnan, K. P. and Venugopal, K. P. (1994). Alopex: A Correlation-Based Learning Algorithm for Feedforward and Recurrent Neural Networks. *Neural Computation*, 6(3):469–490.

van der Meer, M., Kurth-Nelson, Z., and Redish, A. D. (2012). Information Processing in Decision-Making Systems. *The Neuroscientist*, 18(4):342–359.

Vogels, T. P., Sprekeler, H., Zenke, F., Clopath, C., and Gerstner, W. (2011). Inhibitory Plasticity Balances Excitation and Inhibition in Sensory Pathways and Memory Networks. *Science*, 334(6062):1569–1573.

von Helmholtz, H. (1896). Handbuch der physiologischen Optik. *Monatshefte für Mathematik und Physik*, 7(1):A60–A61.

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. (2016). Learning to reinforcement learn. *arXiv*, pages 1–17.

Weinberg, S. (1967). A Model of Leptons. *Physical Review Letters*, 19(21):1264–1266.

Weiss, Y., Simoncelli, E. P., and Adelson, E. H. (2002). Motion illusions as optimal percepts. *Nature Neuroscience*, 5(6):598–604.

Werbos, P. J. (1994). *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting.*

Werfel, J., Xie, X., and Seung, H. S. (2005). Learning curves for stochastic gradient descent in linear feedforward networks. *Neural Computation*, 17(12):2699–2718.

Whittington, J. C. R. and Bogacz, R. (2015). Learning in cortical networks through error back-propagation. *bioRxiv*, 44(0):035451.

Whittington, J. C. R. and Bogacz, R. (2017). An Approximation of the Error Backpropagation Algorithm in a Predictive Coding Network with Local Hebbian Synaptic Plasticity. *Neural Computation*, 29(5):1229–1262.

Whittington, J. C. R. and Bogacz, R. (2019). Theories of Error Back-Propagation in the Brain. *Trends in Cognitive Sciences*, xx:1–16.

Whittington, J. C. R., Muller, T. H., Mark, S., Barry, C., and Behrens, T. E. J. (2018). Generalisation of structural knowledge in the hippocampal-entorhinal system. *Advances in Neural Information Processing Systems 31*, 31:8493–8504.

Whittington, J. C. R., Muller, T. H., Mark, S., Barry, C., Burgess, N., and Behrens, T. E. (2019). The Tolman-Eichenbaum Machine: Unifying space and relational memory through generalisation in the hippocampal formation. *bioRxiv*, 0:770495.

Wikenheiser, A. M. 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.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256.

Wills, T. J., Lever, C., Cacucci, F., Burgess, N., and O'Keefe, J. (2005). Attractor dynamics in the hippocampal representation of the local environment. *Science*, 308(5723):873–876.

Woloszyn, L. and Sheinberg, D. L. (2012). Effects of Long-Term Visual Experience on Responses of Distinct Classes of Single Units in Inferior Temporal Cortex. *Neuron*, 74(1):193–205.

Wood, E. R., Dudchenko, P. A., and Eichenbaum, H. (1999). The global record of memory in hippocampal neuronal activity. *Nature*, 397(6720):613–616.

Xie, J. and Padoa-Schioppa, C. (2016). Neuronal remapping and circuit persistence in economic decisions. *Nature Neuroscience*, 19(6):855–861.

Yamins, D. L. K. and DiCarlo, J. J. (2016). Using goal-driven deep learning models to understand sensory cortex. *Nature Neuroscience*, 19(3):356–365.

Yoon, K., Buice, M. A., Barry, C., Hayman, R., Burgess, N., and Fiete, I. R. (2013). Specific evidence of low-dimensional continuous attractor dynamics in grid cells. *Nature Neuroscience*, 16(8):1077–1084.

Zemel, R. S., Dayan, P., and Pouget, A. (1998). Probabilistic Interpretation of Population Codes. *Neural Computation*, 10(2):403–430.

Zenke, F. and Ganguli, S. (2018). SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks. *Neural Computation*, 30(6):1514–1541.

Zhang, K. (1996). Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble: a theory. *The Journal of Neuroscience*, 16(6):2112–2126.