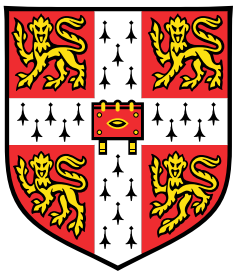
Mixed Selectivity via Unsupervised Learning in Neural Networks



Yan Wu

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Jesus College

April 2019

Declaration

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the Preface and specified in the text. It is not substantially the same as any that I have submitted, or, is being concurrently submitted for a degree or diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. I further state that no substantial part of my thesis has already been submitted, or, is being concurrently submitted for any such degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. It does not exceed the prescribed word limit for the relevant Degree Committee.

Yan Wu
April 2019

Abstract

Thesis Title: Mixed Selectivity via Unsupervised Learning in Neural Networks
Yan Wu

Mixed selectivity characterises neurons that simultaneously respond to different input stimuli. Neurons with mixed selectivity have been observed in multiple brain regions, and are hypothesised to play important roles in neural computation. Recently, both experimental and theoretical work demonstrated the importance of mixed selectivity in context-dependent decision tasks. This thesis extends existing theoretical work on mixed selectivity, arguing for a general and statistical role of mixed selectivity in learning complex dependencies of input stimuli. This role can be motivated from unsupervised learning of generative models, and is exhibited in increased mutual information between the stimuli and their neural representation. This argument is supported empirically using simulation of a sequence disambiguation task that incorporated key aspects of related behaviour experiments. Mixed selectivity neurons that resembled hippocampal place cells were emerged from models optimised only for behaviour. To understand these results, as well as to generalise the findings to a wider range of computations, I provided a formal connection between learning robust models and mixed selectivity.

I would like to dedicate this thesis to my beloved family.

Acknowledgements

First and foremost, I would like to thank my family for their support throughout the years of my studies. In particular, this thesis would not be possible without the company and encouragement of my wife, Yingjue, from the first day we met in Cambridge. I am grateful to the guidance of my supervisor Prof. Máté Lengyel, whose pursuit of quality and beauty in research shaped both this thesis and my research career beyond. I had the privilege to study in the the Computational and Biological Learning Lab (CBL), where I learned how to think and speak as a scholar from venues such as CBL reading club and tea talks, which have been filled with insightful and stimulating discussions. A special thank goes to my advisor Prof. Daniel Wolpert, who took the lead in curating such excellent academic and social atmosphere in CBL, and supported me at various important stages. This thesis benefited from thoughtful discussion with friends and colleagues at CBL, especially the wisdom from David Barrett, Daniel McNamee, Dylan Festa, Guillaume Hennequin, Sina Tootoonian and Yarin Gal. Further, I would like to thank my colleagues at DeepMind, in particular Jonathan Hunt, Sam Ritters and Tim Lillicrap for proof-reading or commenting on parts of the draft. Finally, I appreciate the feedback from my examiners Dr. Richard Turner and Dr. Gergő Orbán.

Table of contents

List of figures

List of tables

Symbols

Roman Symbols

$\mathcal{C}$	cost function
$\mathcal{E}$	error function measuring the distance between an output and its target
$\mathbf{f}(\cdot)$	recognition model
$\mathbf{g}(\cdot)$	generative model
H	entropy of a distribution
$\mathcal{L}$	variational lower-bound
$\mathcal{M}$	parameterised model
$\mathcal{N}(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Gaussian distribution with mean $\boldsymbol{\mu}$ and (co)variance $\boldsymbol{\Sigma}$
$p(\cdot)$	probability distribution of a random variable
$q(\cdot)$	approximate probability distribution
$\mathbf{r}$	latent variables or representations of data
$\mathbf{W}$	connection weight matrix of a neural network
$\mathbf{x}$	a data point or input of the neural network
$\hat{\mathbf{x}}$	estimated or reconstructed input
$\tilde{\mathbf{x}}$	noise corrupted input
$\mathbf{y}$	input passing the nonlinearity of a unit in the neural network
$\mathbf{z}$	output of the neural network

Greek Symbols

κ	mixing index (the measure in Rigotti et al. 2013)
ϕ	parameters of the recognition model
θ	parameters of the generative model
ξ	Noise added to input
ζ	mixed selectivity index (the measure in this work)

Subscripts

D	input dimensionality
K	number of (hidden) neurons
t	time step

Chapter 1

Introduction

1.1 From neural representation to computation

Despite recent rapid advances in artificial intelligence (AI), the brain remains the only instantiation of *general* intelligence. AI has achieved or surpassed human-level performance in an increasing range of tasks, including chess, video games (Mnih et al., 2015), and more recently the ancient game of Go (Silver et al., 2016). However, the human brain is much better at task generalisation, and can learn from far fewer data. Crucially, while AI is often specialised, only humans can master all these tasks simultaneously. Moreover, compared with state-of-the-art AI that relies on more than a dozen of graphical processing units (GPUs) and thousands of watts to perform a single task[1], the human brain is incredibly energy efficient — it weighs no more than a laptop and consumes about the same energy as an incandescent light[2] (Kandel et al., 2000). Understanding the brain, especially how intelligence emerges from the brain is one of the central questions in science.

One way to understand the brain is to understand how neurons represent. Neural representation, usually in the form of recorded neural activities, gives us important evidence on how an animal or human is solving a task. According to the normative perspective of computational neuroscience, the brain employs optimal neural representations for the underlying computations required to solve a task (Dayan and Abbott, 2001). However, it is challenging to infer the computation from merely observed neural activities. The neural systems are highly complex and nonlinear, and recordings of neural activities have large variance due to noise both the in the neural system and in the recording procedure. Abstraction is therefore a necessary tool to navigate these layers of complexity and focus on the specific level of investigation. For this purpose, I use David Marr's three levels (Marr, 1982) to abstract and

[1] A mini workstation with a stack of GPUs.

[2] Estimated at about 25 watts.

analyse neural representation: the computational level formulates the task of interest as a computational problem, the algorithmic level involves how the computational problem can be solved based on specific representations, and the implementational level considers the physical realisation of the system for solving this problem. These levels are closely coupled, for example in studying neural systems, limits of biological systems provide constraints at the implementation level which then restrict the algorithms that can be employed to solve the task's computation problem.

Interpreting neural representations in relation to task-dependent computations has the advantage that, instead of studying the phenomena in isolation, experimental observations are always related to their functional role. The experimental observations of particular interest in this thesis are neural activities that exhibit *mixed selectivity*, which have been observed in a variety of brain areas and computations that involve learning an internal statistical model of inputs (Fiser et al., 2010; Hinton and Ghahramani, 1997). Following Rigotti et al. (2013), mixed selectivity neurons are those that are "tuned to mixtures of multiple task-related aspects". Mixed selectivity has been shown to support complex decision tasks by expanding the dimensionality of representations (Rigotti et al., 2013). Notably, neural representation with elaborated mixed selectivity, such as grid cells, has been found recently from an end-to-end trained artificial system in navigation tasks (Banino et al., 2018).

Based on these observations, this thesis tries to advance theoretical understanding of mixed selectivity from the perspective of statistical modelling. I will demonstrate that mixed selectivity is an *emergent* property of neural networks trained to model external inputs. Such "generative models" can be trained with unsupervised learning, which does not require labelled data and is likely to be employed by the brain, given the large volume of unlabelled data available from perception. In addition to the expanded dimensionality at the *single neuron* level, as demonstrated in previous work (Rigotti et al., 2013), I further show that, at the *population* level, mixed selectivity increases the robustness of the neural network. This argument is supported by numerical simulations on context-dependent spatial prediction task, where we observed neurons with mixed selectivity for both location and context, and is justified by analytical results directly connecting learning generative models and mixed selectivity in more general scenarios.

In summary, this work proposes that mixed selectivity is a consequence of learning the statistical structure of inputs, which supports robust performance against unstructured noise. The statistical model can be learned easily with either denoise training, via adding noise to inputs, or a novel regulariser I developed. As a confirmation of my theory, the recurrent neural network in our experiments exhibited representation that resembled those recorded in multiple brain regions, and increased mixed selectivity was observed after training.

1.2 Mixed selectivity

Of the 100 billion neurons in the brain, their types and connectivities, consequently their firing patterns, vary vastly. For example, neurons in neocortex have complex spontaneous activities, hippocampal place cells have sparse and spatially sensitive firing patterns, while some retinal cells do not fire discrete spikes at all (Kandel et al., 2000).

To understand the computational role of a neural system, one first needs to define the input and output of this system (Dayan and Abbott, 2001). While the neural activities observed can be generally interpreted as the outputs of the system, the inputs are usually decided based on understanding of the system's function. For example, in studying the visual cortex in parsing a visual scene, one may consider individual pixels as the atomic input features, while as for deeper regions such as the hippocampus, more abstract features such as location and other environmental factors are more useful inputs to consider for studying spatial and navigation tasks. Practically, in an experiment, it is preferable if such inputs are easily measurable and controllable.

Once the inputs and outputs of of the system are determined, we can characterise the selectivity of this system, as how the outputs are sensitive to different inputs. For neurons as such systems, pure selective neurons are only sensitive to single inputs, while as mixed selective neurons are sensitive to combinations of different inputs. This intuitive definition of mixed selectivity is based on, but extended the scope of, the (model-based) mixed selectivity defined by Rigotti et al. (2013).

Mixed selectivity represents an important type of neural representations in the brain. Regardless of their exact functional roles, a large number of these neurons, found in different brain regions, are characterised by their selective firing to multiple input features, which I generally considered as having mixed selectivity. For the examples used earlier: at the level of sensory processing, complex cells in primary visual cortex are tuned to both the spatial orientation and movement direction of visual stimuli (Hubel and Wiesel, 1962); at a higher level, hippocampal cells use mixed selectivity by activities tuned to both spatial locations and non-spatial contexts (Wood et al., 2000).

Rigotti et al. (2013) argued that nonlinear mixed selectivity results in high dimensional neural representations. Neurons with mixed selectivity project originally separated input stimuli into higher dimensional *feature spaces*. Such high dimensional feature spaces may benefit down stream computation. For example, originally linearly non-separable inputs may become linearly separable features, which is visualised in Figure 1.1. Note that the projection needs to be nonlinear for this purpose; otherwise, the inputs are only projected to another hyper-plane without increasing dimensionality. Figure 1.1 **b** and **c** compare neurons with linear and nonlinear mixed selectivity.

Consequently, in neural systems, mixed selectivity may simplify the decoding task for down-stream neurons, since multiple task relevant factors, such as both the immediate stimuli the less immediate context, would already be represented together via mixed selectivity. Consistent with this theoretical argument, experimental evidence suggests that mixed-selectivity is strongly correlated with task performance, which makes it a promising interface for investigating the underlying computations supporting corresponding behavioural tasks. Rigotti et al. (2013) show that the degree of nonlinear mixed selectivity in monkey's PFC were correlated with the performance of context-dependent decisions. More recently, in a context-dependent conditioning task, increased errors were observed to be correlated with reduced encoding of context in the amygdala (Saez et al., 2015). In this experiment, neurons in the amygdala encoded both the identity of conditional stimuli and contexts, which exhibiting mixed selectivity. Therefore, it suggested a correlation between decreased performance and reduced mixed selectivity.

1.3 Sequence Disambiguation: a Testbed for Mixed Selectivity

In this thesis, I mainly investigate mixed selectivity via experiments on sequence learning. A variety of everyday activities, including language understanding and navigation, are examples of sequence learning. In particular, disambiguation of distinct sequences with overlapping parts (A-B-C-D vs. E-B-C-F) is involved in tasks ranging from finding the location of a parked car to appropriately chaining actions to prepare a breakfast.

This task is well suited as a test-bed for mixed selectivity, since different types of information are required *simultaneously* for disambiguating different sequences. For example, to predict the next element in a sequence that partially overlaps with another, both the current element that specifies the location within the sequence and the contextual information that uniquely identifies a sequence are required at the same time. Compared with another widely used sequence memory task (e.g., Rigotti et al., 2013), more neural data are available on behavioural experiments involving sequence learning, which I will review below.

1.3.1 Task

A classical sequence learning task that requires sequence disambiguation is the T-maze alternation task (Figure 1.2, Wood et al., 2000). In this task, a rat randomly started from either the left or right arm of a T-maze and moved down to the bottom of the maze. After moving along the central arm to the T-junction, the decision point, it needed to turn left

or right, depending on where it entered the central arm: if it entered from the right arm, it needed to turn left (left-turn trial), and *vice versa* (right-turn trial). A trial ended at the upper-left or upper-right corner of the T-maze. To reinforce this behaviour, cued reward was delivered when the rat correctly completed the trial. Rats were able to learn this task over repeated training sessions (Wood et al., 2000). The challenge of this task was at the decision point, where the decision of which arm to proceed had to be made using contextual information (left or right turn trial), which was determined before the rat entered the central arm.

1.3.2 Behavioural Experiments

Experiments in both humans and other animals have found that multiple brain regions, including the hippocampus, the parahippocampal area, and frontal cortical areas, may support disambiguation in sequence learning. For example, Wood et al. (2000) observed that, in the T-maze alternation task described in section 1.3.1, approximately $\frac{2}{3}$ of hippocampal cells with receptive fields on the shared central arm fired selectively, depending on whether the rat was in a left-turn or right-turn trial (figure 1.3). Thus, these cells exhibited mixed selectivity to both location and context, and were named "splitter cells" by Wood et al. (2000), as if they "split" left-turn and right-turn trials.

In another experiment, the rats were trained to run on a running wheel between left-turn and right-turn trials in a figure-eight maze as shown in figure 1.4 (Pastalkova et al., 2008). In this case, the wheel corresponded to the decision point in the original T-maze experiment (Figure 1.2), after which the rats needed to decide which arm to follow. By requiring the rats to run on the wheel, where their locations stayed the same, location information and the motor command of running were dissociated at the decision point. Nevertheless, internally generated hippocampal neural activities during wheel-running were selective to both the time spent on the wheel and the future choice of left-turn or right-turn (Figure 1.4). Importantly, such selectivity to trial types was highly correlated with task performance, and was only observed in correct trials.

In another T-maze based experiment, the choice was conditioned on an odour cue (Fujisawa et al., 2008). Instead of alternating between the left and right arm of the maze, an odour cue was presented at the beginning of a trial, specifying which arm was rewarded (Figure 1.5, **a**). This context information needed to be maintained along the central arm to the T-junction before the rat actually turned to one of the two arms. Consistent with this computational demand, medial prefrontal cortex (mPFC) neurons in layer 2/3 and layer 5 were found to respond to both locations and odour cues (Figure 1.5). More recently, Harvey et al. (2009) tested a similar perceptual decision task based on visual cues (Figure

1.6). Equipped with a fully controlled virtual reality environment and calcium imaging, they observed neurons in posterior parietal cortex (PPC) whose activities formed choice specific sequences during the task.

In addition to spatial tasks, hippocampal cells have also been found to fire differentially for odours shared between different sequences (Ginther et al., 2011). More recently, increased activity in the hippocampus, parahippocampal cortex and orbitofrontal cortex were observed, on both rats and humans, during spatial disambiguation tasks in virtual reality (Brown et al., 2010; Harvey et al., 2009). In addition, the prefrontal cortex (PFC) was likely to play a role in sequence learning and disambiguation as indicated by electrophysiological as well as imaging data (Baeg et al., 2003; Euston et al., 2007; Fuster, 2001; Harvey et al., 2012; Huettel et al., 2002). In conclusion, neural representations in a variety of brain regions have been found to exhibit mixed selectivity during sequence disambiguation tasks.

1.3.3 Computational Models

How does the brain solve sequence learning tasks in these experiments? Theories and computational models suggest this is achieved by encoding context together with sequences of stimuli (e.g., locations). However, the exact mechanisms employed by the brain remains unclear. Here I review existing computational models focusing on sequence disambiguation. Although there are experimental data available in various cortical regions (section 1.3.2), most of the models were focused on the hippocampus, which has dense recurrent connections (in area CA3) and sparse firing that make it a suitable candidate for sequence coding (Eichenbaum and Cohen, 2001). Nevertheless, the computational principles of these models as well as the one I will present in this thesis, are not restricted to the hippocampus. This section reviews these historically important models at length. Although these algorithms are not immediately related to the model I am going to present, they provide important insights into sequence disambiguation at different levels.

Temporal Context Model (TCM)

To solve the sequence disambiguation task at the computational level, Howard and Kahana (2002) presented a minimalist model of context learning. It is a linear recurrent neural network that encodes context through time as continuously changing inputs. Importantly, it suggests sequences themselves can be used to encode the context for disambiguation, which is conceptually the same as more complex non-linear recurrent neural networks (Chapter 3). The temporal context model (TCM) is based on distributed representation of items as vectors

in a high-dimensional space. It provides a principled explanation of recency and contiguity effects in recall.

Assume the feature vector $\mathbf{f}_i$ is a function of external input at time i, and $\mathbf{t}_i$ is the context at time i. The connection matrix $\boldsymbol{M} = \sum_i \mathbf{f}_i \cdot \mathbf{t}_i^\intercal$, as the outer-product of $\mathbf{f}$ and $\mathbf{t}$, represents the association between $\mathbf{f}$ and $\mathbf{t}$. The input provided by a context $\mathbf{t}_j$ is defined as $\mathbf{f}^{IN} \equiv \boldsymbol{M}\,\mathbf{t}_j$. The TCM is based on the observation that, when $\mathbf{f}$'s form a set of orthogonal basis [3], the similarity between contexts at i and j, measured by their inner-product $\mathbf{t}_i^\intercal \cdot \mathbf{t}_j$, is equivalent to an activation based on the input defined as $\mathbf{a}_i \equiv \mathbf{f}^{IN\intercal} \cdot \mathbf{f}_i$. This suggests that the distributed features $\mathbf{f}$ encode contextual information, and such contextual information can be read-out easily using these features alone.

Howard and Kahana (2002) further propose that the encoded context can be made to evolve smoothly in time by an autoregressive process $\mathbf{t}_i = \rho\,\mathbf{t}_{i-1} + \mathbf{t}_i^{IN}$, where $\mathbf{t}^{IN}$ is the input at time i that changes the context, and $0 \leqslant \rho \leqslant 1$ determines how fast the context drifts. While $\mathbf{t}^{IN}$ can be anything that changes with time, the TCM uses a *retrieved context* to drive context drift, such that the retrieved context is more correlated with recent items. This input is defined as $\mathbf{t}_i^{IN} = \beta \cdot \boldsymbol{M}_i\,\mathbf{f}_i$, where β is another free parameter controlling the strength of the retrieved context. Therefore, the TCM has a recurrent structure, by which input features also serve as temporal context though the transform $\boldsymbol{M}$. While the TCM's simple and elegant way of modelling temporal context is inspiring, the computational power is constrained by the linear model, which may be problematic for complex tasks. For example, as a restriction of its linearity, it is unclear how the required orthogonal features can be generated efficiently for realistically changing input stimuli. It is also unclear how the TCM can disambiguate sequences depending on non-smooth and non-autoregressive context, as in the tasks I study here (section 1.3.1).

Models with Local Context Neurons

For neural network models, one way to represent context is to specify a population of neurons with pure selectivity for context. One of the first computational models based on such "local context neurons" was described by Levy (1996), and more biologically plausible models with detailed biophysical features were subsequently developed in Wallenstein and Hasselmo (1997) and Sohal and Hasselmo (1998). Since this thesis focuses on the computational mechanism rather than biologically plausible implementations, I will only review the work of Levy (1996) in more detail here.

[3] Howard and Kahana (2002) allow identical $\mathbf{f}$'s, but here I assume all $\mathbf{f}$'s are different to simplify the analysis.

Levy (1996) presented a minimal biologically plausible model of hippocampal CA3 that could solve various sequence learning tasks. It is a recurrent neural network trained on a local Hebbian learning rule. The external inputs **x** driving the network are sparse binary patterns postulated to come from the enthorinal cortex and the dentate gyrus. Importantly, only a fraction of the neurons **y** directly receive external input, while as other neurons, analogous to hidden neurons in modern neural networks, may develop into local context neurons through learning. The outputs **z** of each neuron feeds-back as input into **y** through sparse recurrent connections (with connectivities from 5% to 20%) that are randomly initialised.

More formally, this model operates according to

$$y_j = \frac{\sum_i w_{ij}\, c_{ij}\, z_i(t-1)}{\sum_i w_{ij}\, c_{ij}\, z_i(t-1) + K_I \sum_i x_i(t) + K_R \sum_i z_i(t-1)} \tag{1.1}$$

$$z_j(t) = \begin{cases} 1 \text{ if } y_j(t) \geqslant \theta \text{ or if } x_j(t) = 1 \\ 0 \text{ otherwise} \end{cases} \tag{1.2}$$

where K_I is the strength of feed-forward inhibition, K_R is the feedback inhibition scale constant, $c_{ij} \in \{0,1\}$ governs the (sparse) connectivity between neuron i and j, and w_{ij} is the excitatory weight between neuron i and j. The output from the previous step $t-1$ feeds-back via the recurrent connections (w and K_R). There are 512 excitatory neurons and 1 inhibitory neuron, implicitly modelled as the recurrent inhibition term. The local synaptic learning rule is

$$w_{ij}(t) \leftarrow w_{ij}(t-1) + \varepsilon z_j(t) \left[z_i(t-1) - w_{ij}(t-1) \right] \tag{1.3}$$

where ε is the learning rate.

Levy (1996) demonstrated that this model was able to learn a series of sequential tasks. In tasks requiring disambiguation, *local context neurons* that were selective to segments of sequences emerged after learning. This model relied on highly processed inputs in the form of sparse binary patterns. The inhibition parameters K_I and K_R had to be tuned manually, and the stability of the learning process was unclear, as the Hebbian learning rule was not derived from any specific objective function. Although it has been a pioneer of biologically plausible models that learn to solve a large range of sequential tasks, it is questionable how this model can scale-up for more realistic tasks without specifically tailored input/output structures, as well as the heuristic learning rule. Moreover, its localist approach precluded mixed selectivity for context.

Hasselmo and Eichenbaum (2005)

In a more comprehensive model, in which physiological properties of the whole hippocampal formation (including both the hippocampus and the entorhinal cortex) were considered, Hasselmo and Eichenbaum (2005) used different neural populations to model sequence association and context coding. In their model, the recurrent connections in entorhinal cortex layer III (EC III) encode the associations between consecutive time steps, while entorhinal cortex layer II (EC II) provides persistent and slowly changing temporal context similar to the more abstract TCM (Howard and Kahana, 2002). More specifically, the recurrent connections in EC III are modified by the correlation between the current and previous behavioural states ($\mathbf{b}_c$ and $\mathbf{b}_{c-1}$) through

$$\Delta \mathbf{W}_{\text{EC III}} = \mathbf{b}_c \, \mathbf{b}_{c-1}{}^{\mathsf{T}} \tag{1.4}$$

The activities of neurons in EC III evolve through time according to the equation

$$\mathbf{a}_{\text{EC III}}(t) = \mathbf{b}_c + \eta^{T/t} \, \mathbf{W}_{\text{EC III}} \, \text{ReLU}\left(\mathbf{a}_{\text{EC III}}(t-1) - \psi\right) \tag{1.5}$$

where the ReLU activation function is defined as $\text{ReLU}(\mathbf{x}) = \max(0, \mathbf{x})$, and ψ is a threshold. η is a positive constant smaller than 1 and $0 \leqslant t \leqslant T$ is the time step within a theta cycle, so $\eta^{T/t}$ represent the rhythmic activity of EC III neurons. The sustained activities in EC II represent temporal context through

$$\mathbf{a}_{c,\text{EC II}} = \mathbf{b}_c + \mu^{c-s} \, \mathbf{a}_{c-1,\text{EC II}} \tag{1.6}$$

where $0 < \mu < 1$ is a constant, and $s < c$ denotes the time step of a previous state. Expanding this recursive equation reveals that $a_{c,\text{EC II}} = \sum_{s=1}^{c} \mu^{c-s} b_s$. In other words, EC II activity represents the temporal context as a sum of the previous c behavioural states discounted by μ^{c-s}. The retrieved sequences from EC III (equation 1.5, possibly with ambiguity) and the temporal context from EC II are then combined in CA1 through CA3 to provide disambiguated sequences.

The simulation of splitter cells in a grid environment are illustrated in Figure 1.7. Although this model is based on observations across various regions of the hippocampal formation, the functional roles of these regions are less justified from a computational perspectives. For example, it is unclear why forward association and context coding need to be separated anatomically, and why the relatively simple role of synthesizing sequences and context needs both CA3 and CA1. From a model selection point of view, it is necessary to justify this additional structural complexity (MacKay, 2003). Additionally, this separation of functional roles into different areas is incompatible with *in vivo* firing patterns of hippocampal

and entorhinal cells that employ sparse distributed population codes (Smith and Mizumori, 2006).

Zilli and Hasselmo (2008)

In their models motivated by goal-directed behaviours, Zilli and Hasselmo (2008) assumed that contextual information was imported from other parts of the brain or memory systems. The model provided an explicit connection between reinforcement learning (RL) and memory systems, which has also seen in machine learning literature such as Blundell et al. (2016). Theory of RL (Sutton and Barto, 1998) is a framework for studying goal-directed behaviours. In a typical RL setting that assumes Markov decision processes (MDP), the optimal actions at time t, a_t^*, is only based on the current state s_t. This set-up is difficult to account for animal behaviours replying on past experience, without the extra complexity from assuming partial observability (as in partially observable Markov decision processes). For instance, MDP fails when the current state consist only of sensory input s_{sensory} that is ambiguous and cannot provide adequate information to support optimal actions.

Zilli and Hasselmo (2008) addressed this issue by augmenting the state with an episodic memory system and a working memory system, such that the state at time t became a tuple $s_t = \langle s_{\text{sensory}}, s_{\text{episodic}}, s_{\text{working}} \rangle$. s_{episodic} passively holds the most recent n sensory inputs. When these n sensory inputs are still not enough to disambiguate the current state — for example, when the sensory input that differentiate two otherwise overlapping sequences appeared before the previous n time step — an actively maintained s_{working}, which may buffer any previous input, comes to help. Although such augmented states are capable of disambiguating any sequences in theory, storing all the n previous sensory inputs may be inefficient. In addition, it is unclear what is the best strategy to buffer the sensory input in s_{working}. Zilli and Hasselmo (2008) demonstrated that this model could solve all except one of the sequential disambiguation tasks they simulated.

Rajan, Harvey and Tank (2016)

From a different perspective, Rajan et al. (2016) showed that recurrent neural networks directly trained to *reproduce* the observed neural representations could support context-dependent sequential decision tasks. They studied what kind of neural network structure gives rise to the sequential activity patterns observed in a variety of behavioural tasks. In particular, they consider the data collected from the sequential decision making task in Harvey et al. (2012), which I reviewed in the last section (see also figure 1.6). They proposed a measure *bVar* to compare different activity patterns. It measures the *stereotypy* of the

activity as the percentage of variance that can be explained by a moving template. The formal definition of bVar will be introduced in section 4.3, which I will use to analyse the simulated activities from my model. Rajan et al. (2016) showed that neither highly structured nor fully random neural networks could explain the PPC data from Harvey et al. (2012) — the former was too stereotypy (too high bVar) while the latter was too disordered (too low bVar). On the other hand, between these two extremes, they demonstrated that a recurrent neural network trained from randomly initialised weights could explain the data (showing similar bVar to that found experimentally, Figure 1.8).

In the notation of Rajan et al. (2016), the activation of each neuron $\mathbf{x}_i$ is determined by

$$\tau \frac{dx_i}{dt} = -x_i + \sum_{j=1}^{N} J_{ij}\,\phi(x_j) + h_i \tag{1.7}$$

where $\mathbf{J}$ is the recurrent connection matrix and $\phi(x) = \frac{1}{1+e^{-(x-\theta)}}$ is the sigmoid nonlinearity with bias (threshold) θ. The external input into each neuron h_i is the same series of Gaussian random noise that repeated from trial to trial, simulating sensory stimuli. The matrix $\mathbf{J}$ was trained by the FORCE algorithm (Sussillo and Abbott, 2009), so the activity x_i directly regressed neural activities extracted from the calcium imaging recorded in Harvey et al. (2012). Importantly, they showed that only a small portion (about 10%) of the recurrent connection weights needed to be modified to reproduce activities similar to data. They further demonstrated the computational function of this biologically plausible representation by computing the selectivity index during the delayed decision period as in Harvey et al. (2012), showing such patterns could support simulated sequential decision making involving disambiguation (Figure 1.8).

This work was the first to show a concrete connection between neural networks trained from random initialisation and *in vivo* recorded data during sequence learning tasks. Nevertheless, since the biologically plausible representation from this model was obtained from regressing neural data, it was less clear whether such representation, although sufficient, was also necessary, and could spontaneously emerge from behavioural tasks.

In summary, all these models include mechanisms that are either manually designed or explicitly trained to imitate observed representations, leaving open the question of what representation of context is most appropriate for sequence learning and disambiguation. On the other hand, results from experiments under different settings unanimously revealed representations with highly mixed selectivity (section 1.3.2), which are at odds with many of the models based on hand-crafted features that do not exhibit mixed selectivity.

It is worth mentioning that natural language processing (NLP), which involves parsing complex sequences of symbols, also addresses the challenge of sequence disambiguation.

NLP has been one of the most active research areas in machine learning, and a large number of exciting results using neural networks have been reported recently (Bahdanau et al., 2014; Mikolov et al., 2010; Sutskever et al., 2014). However, the challenge of sequence disambiguation is compounded by several other issues in natural language processing. Therefore, I skip this part of the literature from this review, and concentrate on models specifically dealing with sequence disambiguation and neural representations.

1.4 Structure of this thesis

Chapter 2 reviews artificial neural networks, with an emphasis on their connections with probabilistic generative models, which leads to a novel interpretation of denoising training as introducing a smooth prior as a constrained Gaussian mixture model. It explains how assumptions and statistical properties of such models affect the neural representations developed during learning. A class of neural networks called autoencoders can be interpreted as generative models, under which their representations are naturally explained as latent variables in these models.

Chapter 3 then extends the probabilistic interpretation to recurrent neural networks (RNNs), after reviewing common training methods for RNNs. Due to the additional complexities involved in RNNs, more technical analysis of RNNs and sequence learning are presented separately in Appendix A.

In chapter 4, I present an experiment using recurrent neural networks (RNNs) for a sequence learning task. This task requires disambiguating overlapping sequences of locations, which requires the neural population to represent both the locations within sequences and contexts distinguishing different sequences (at even overlapping locations) simultaneously. However, it remains an open question whether the location and contextual information shall be mixed at single neuron level, resulting in neurons with mixed selectivity. To answer this question, I trained RNNs for one-step sequence prediction. The trained RNNs were competent in both predicting and recalling sequences. In addition, they spontaneously developed biologically plausible neural representations showing mixed selectivity. I then explain this performance from the perspective of dynamical systems.

Chapter 5 further discusses mixed selectivity. There I propose a new information theoretical measure of mixed selectivity and compare it with existing measure of mixed selectivity from Rigotti et al. (2013). I try to argue from both theoretical and simulation analysis that the measure of mixed selectivity proposed here is more relevant for understanding the connection between neural representation and task performance.

Chapter 6 concludes the analysis with the main theoretical result of this thesis — training a robust generative model implicitly increases mixed selectivity. This suggests that the emergence of mixed selectivity after training the model for sequence disambiguation, as well as the observed correlation between mixed selectivity and performance, were not coincidences. Instead, mixed selectivity is a consequence of learning robust generative models. The wider implications of this work are discussed in Chapter 7.

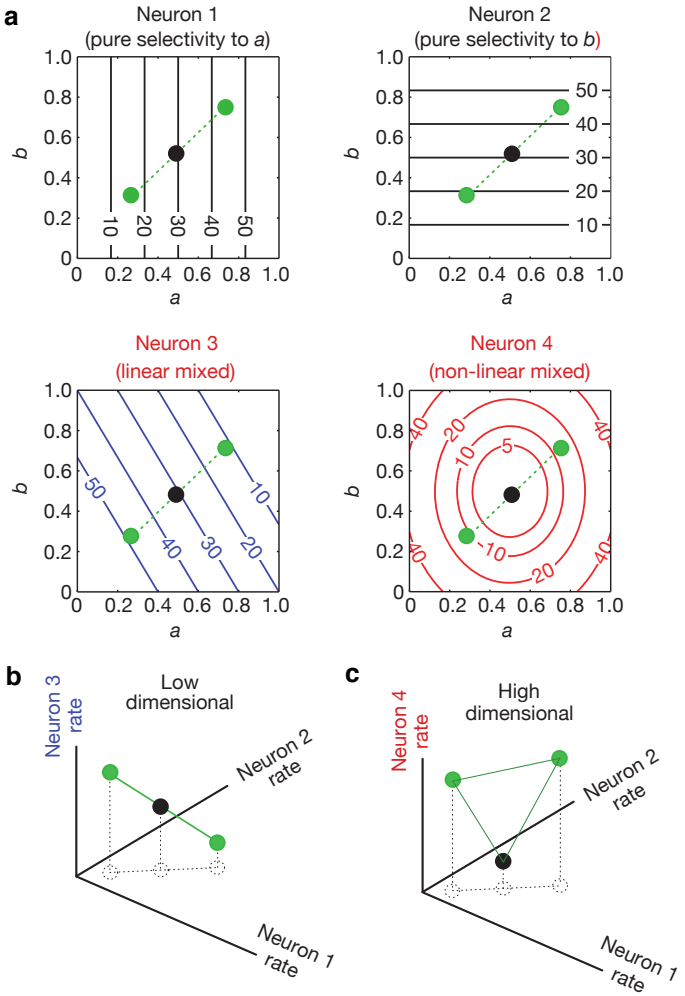


Fig. 1.1 Low and high-dimensional neural representations, and mixed selectivity (reproduced from Rigotti et al., 2013). (**a**) Contour plots of the responses (spikes per second) of four hypothetical neurons to two continuous stimuli (a and b). Neurons 1, 2 were pure selectivity neurons, selective to individual stimuli (a and b, respectively). Neuron 3 is a linear mixed selectivity neuron. Neuron 4 is a nonlinear mixed selectivity neuron. The green circles indicate the responses to three sensory stimuli parametrised by different a, b combinations. (**b**) The responses of the pure and linear mixed selectivity neurons from **a** in the space of activity patterns elicited by the three stimuli indicated by the green circles in **a**. (**c**) As in **b**, with the third neuron being the nonlinear mixed selectivity Neuron 4 in **a**. This higher dimensionality has an important role when the activity is read out by linear classifiers. For example, in **b** it is impossible for any linear classifier to respond to the darker central circle and not to one of the other two. But it is possible in **c**, for instance for a linear classifier corresponding to an appropriately positioned horizontal plane.

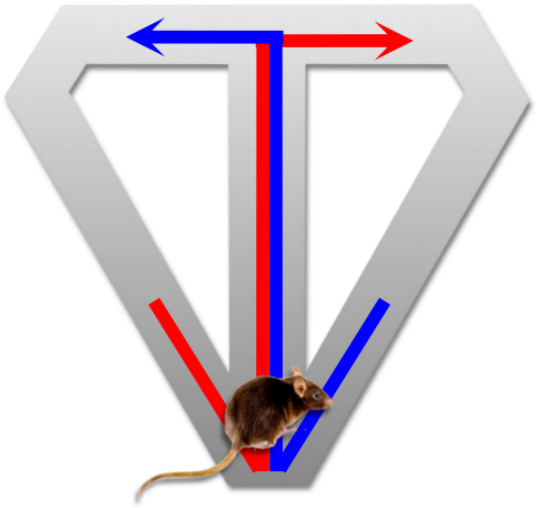


Fig. 1.2 The T-maze alternation task adapted from Wood et al. (2000). A rat randomly started from either one of the two side-arms. At the decision point (the T-shaped junction), the rat needed to turn left if it entered from the right arm (left-turn trial, blue line); it needed to turn right if it entered from the left arm (right-turn trial, red line). In animal experiments, this behaviour was reinforced by rewarding at the end of correct trials. Here the rewards are omitted for simplicity, thus the task is interpreted as unsupervised sequence learning.

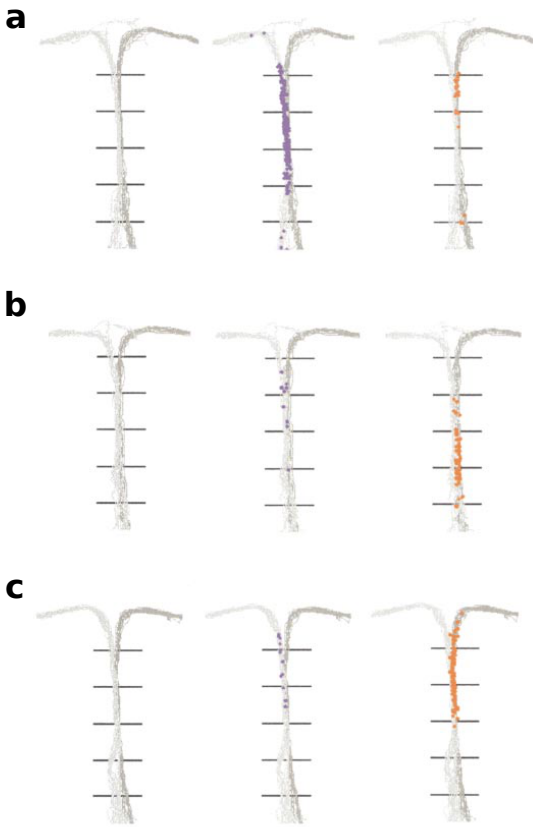


Fig. 1.3 Examples of splitter cells that were active while the rat traversed the central stem of the T-maze (reproduced from Wood et al., 2000, task is illustrated in Figure 1.2). These cells fired almost exclusively during either left-turn or right-turn trials. In each example, the paths taken by the animals on the central stem are plotted in the left panel (light grey, left-turn trial; dark grey, right-turn trial). In the middle and right panels, the locations of the rat when individual spikes occurred in hippocampal CA1 place cells are indicated separately for left-turn trials (purple dots) and right-turn trials (orange dots). (**a**) A cell that fired almost exclusively on left-turn trials as the rat traversed later sectors of the stem. (**b**) A cell that fired almost exclusively on right-turn trials as the rat traversed early sectors of the stem. (**c**) A cell that fired almost exclusively on right-turn trials as the rat traversed later sectors of the stem.

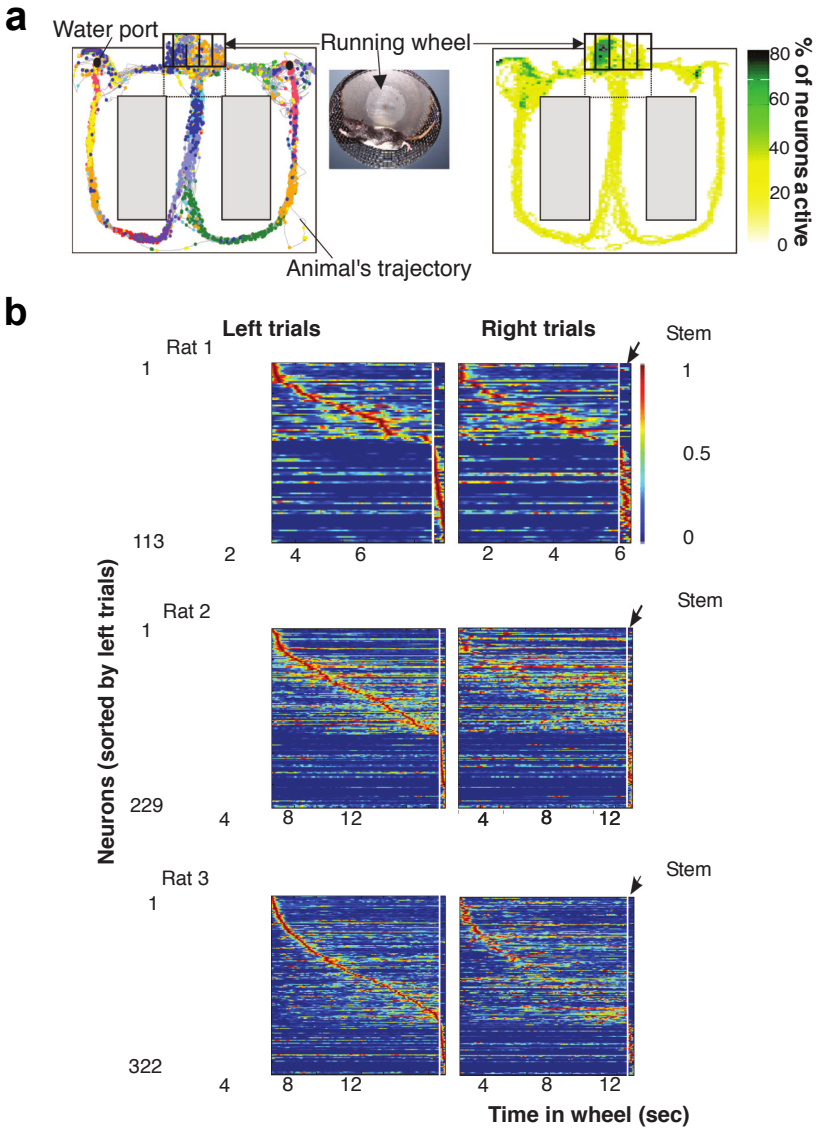


Fig. 1.4 Cell-assembly activities in the wheel predicted the future choice of the rat in the maze (reproduced from Pastalkova et al., 2008). (**a**) Illustration of the experimental scheme. The rats were trained to alternate between the left and right arm of the maze to receive reward (water) after correct trials. Before making the decision, the rats were running in the same direction in a wheel. Left: colour-coded spikes (dots) of simultaneously recorded hippocampal CA1 pyramidal neurons. The rat was required to run in the wheel facing to the left during the delay between the runs in the maze. Right: percent of neurons firing >0.2 Hz within each pixel. The highest percentage of neurons was active when rats were running in the wheel. (**b**) Normalised firing rate profiles of neurons during wheel running and in the stem of the maze, ordered by the latency of their peak firing rates during trials (each line is a single cell; cells are combined from all sessions). White line shows the time gap between the end of wheel running and the initiation of maze stem traversal.

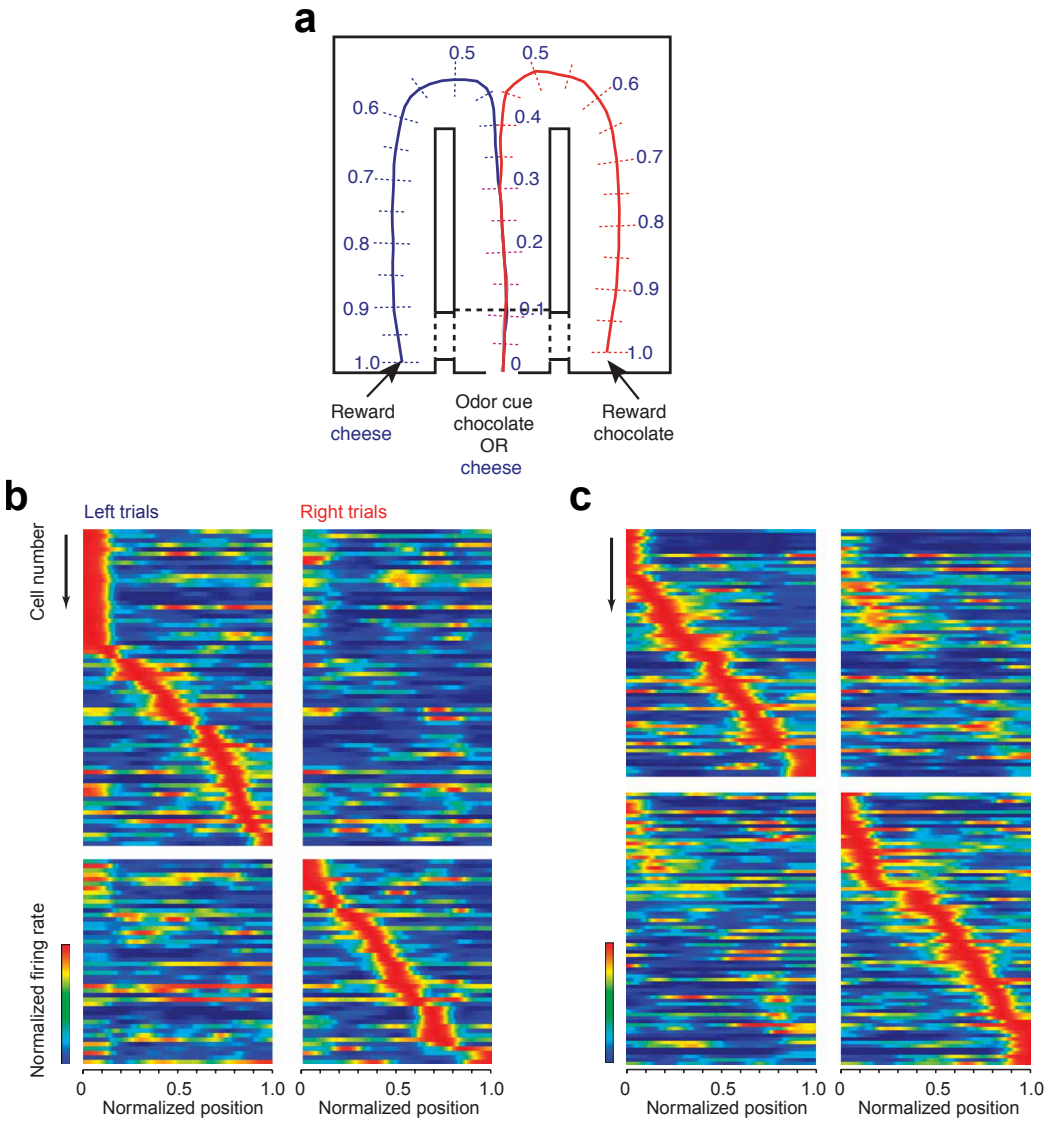


Fig. 1.5 Behaviour- and position-selective firing activity of PFC single neurons (reproduced from Fujisawa et al., 2008). (**a**) Illustration of odour-based matching-to-sample task. An odor cue (chocolate or cheese) is presented following a nose-poke in a start box (position 0). Cheese or chocolate odour signals the availability of cheese or chocolate reward in the left or right goal area (position 1), respectively. Travel trajectories were linearised and represented parametrically as a continuous, one-dimensional line for each trial (blue and red lines). Firing patterns of neurons recorded simultaneously in either layer 2/3 (**b**) or layer 5 (**c**) in two rats. Each row represents the position-dependent firing rate of a single neuron (normalized relative to its peak firing rate). Neurons were ordered by the location of their peak firing rates relative to the rat's position in the maze.

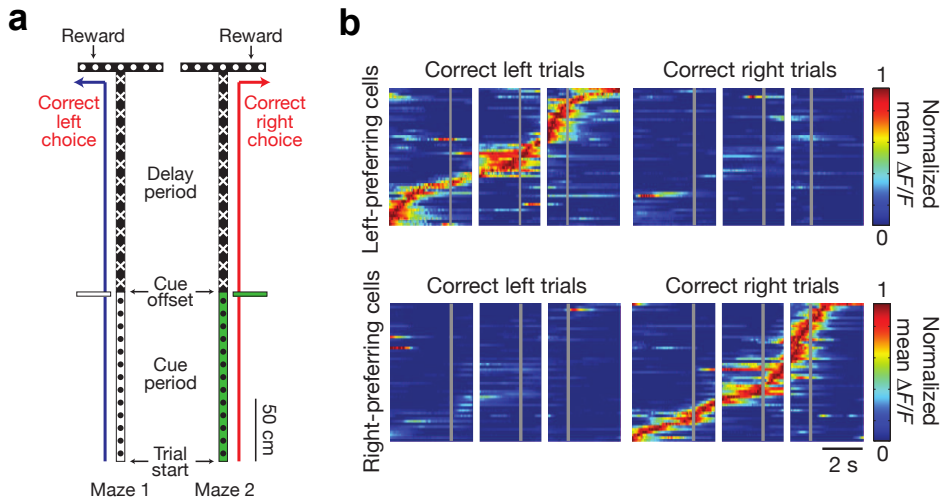


Fig. 1.6 Imaging PPC neuronal activity during the T-maze task (reproduced from Harvey et al., 2012). (**a**) Diagram of the two trial types of the virtual T-maze that differed only in the cue period and the reward location. Patterns in the diagram reflect the patterns present on the virtual maze walls. (**b**) Normalised mean $\Delta F/F$ traces for all the choice-specific, task-modulated cells (one cell per row) imaged in a single mouse and divided by left-preferring and right-preferring cells. Traces were normalised to the peak of each cell's mean $\Delta F/F$ trace on preferred trials and sorted by the peak time.

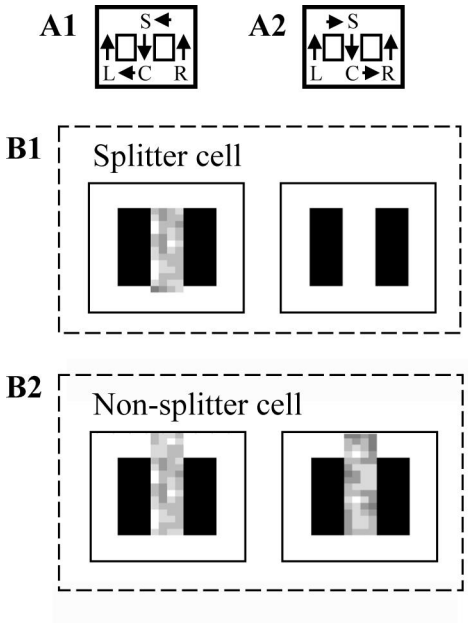


Fig. 1.7 Simulation of a hippocampal "splitter cell" (**B1**) that was sensitive to both location and context (trial type). In contrast, the "non-splitter cell" (**B2**) was not selective to the context (reproduced from Hasselmo and Eichenbaum, 2005). The two white boxes in panels from A1 / A2, as well as the black boxes in panels from B1 / B2, represent walls separating the environment into one middle and two side lanes. **A1 / A2** Behavioural context showing two trial types (L, R, C, S represent different stages in the simulated environment). **B1**. A splitter cell representing the location just to the right of the choice point will fire as part of sequences retrieved in the stem after the virtual rat performs a right turn response (left side), but will not fire after a left turn response (right side). Gray scale represents rates of spikes (darker for higher) fired by simulated cells when the virtual rat is in particular locations. **B2**. A non-splitter cell representing the choice point will fire after visiting either arm.

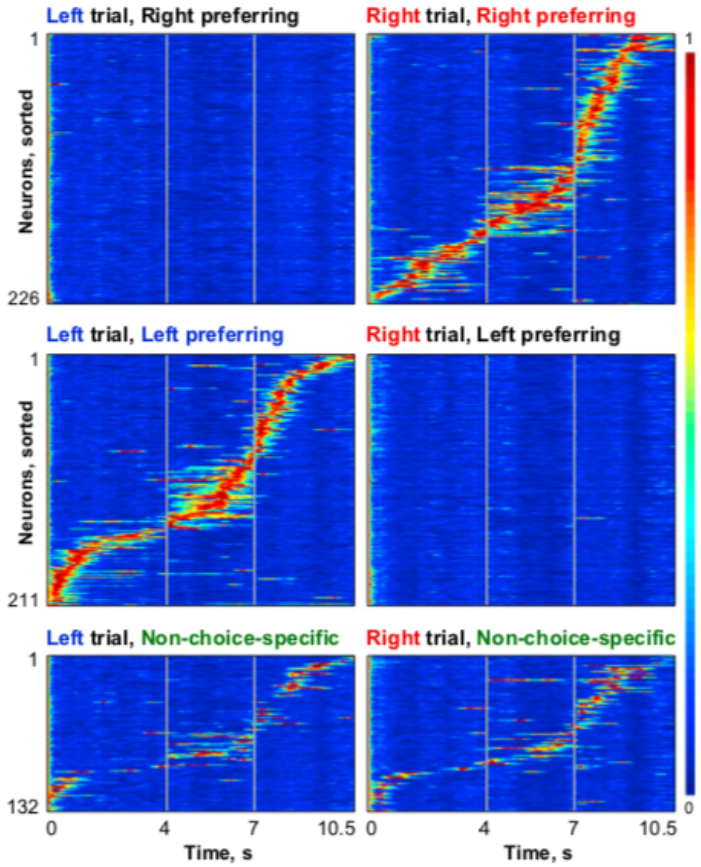


Fig. 1.8 Simulation of PPC neural representations (reproduced from Rajan et al., 2016). The sparsely connected recurrent neural network was trained to imitate the neural representation recorded by Harvey et al. (2012). The outputs of the 569-neuron network with p = 16% plastic synapses, sorted and normalized by the peak of the output of each neuron, showed a match with the data (Figure 1.6). For this network, pVar = 85%.

Chapter 2

Neural networks: a probabilistic view

This chapter introduces artificial neural networks, with an emphasise on the probabilistic interpretation of these computational models. In addition to dealing with uncertainty, this view provides useful tools for understanding the behaviours of neural networks, which enables us to recast training a broad range of neural network models as building statistical models of data. This chapter ends with introducing a regulariser for a practical approximation of training generative models with Gaussian mixture model prior. This regulariser provides a steppingstone for latter chapters that further extend the analysis of neural networks using information theory.

2.1 Neural networks

Artificial neural networks abstract neural circuits in the brain into interconnected simple computational units. These computational units usually have monotonic nonlinear activation functions, so that they can be combined to approximate complex functions (Minsky and Papert, 1969). The outputs from these computational units can be interpreted as firing rates of neurons, so we simply call these computational units "neurons" in the following text. From this view, the output of a neuron as a function of the external input variable is a model of the tuning curve of a real neuron. Similarly, the connections between these units are analogous to synapses. Rudimentary though such models are, artificial neural networks are able to develop biologically plausible neural representations, including those with mixed selectivity; as expressive computational models, they are able to solve challenging tasks including sequence learning in the face of ambiguity (section 3.1). Neural networks have achieved state-of-the-art performance in a number of real-world challenging tasks (LeCun et al., 2015; Schmidhuber, 2015).

Neural networks may develop representations resembling those in the brain, after being optimised for computational or behavioural task performance. For example, firing patterns similar to those from the primary visual cortex emerged after training on natural images (Cadena et al., 2019; Glorot et al., 2011; Olshausen and Field, 1996). When trained for solving motor control tasks, neurons with tuning curves similar to those in motor cortex were observed (Sussillo et al., 2015). More recently, representations similar to hippocampal place cells and enthorinal grid cells were observed after training neural networks on spatial navigation tasks (Banino et al., 2018).

Even in more abstract tasks where clear corresponding neural representations have yet to be observed experimentally, the representations developed in neural network models exhibit mixed selectivity. For example, since the early years of connectionism, neural representations that emerged during simple semantic learning showed mixed selectivity by correlating concepts with closely related semantic meanings (Rumelhart et al., 1987, 1986). In the more complex task of language learning, neurons selective to context dependent semantic units at different levels (e.g., sentence, phases) were found in recurrent neural networks (Karpathy et al., 2015).

2.2 Generative models

Generative models, as the name suggests, can generate samples similar to the observed data $\mathbf{x} \in \mathcal{X}$, which intuitively suggests these models *understand* the dataset $\mathcal{X}$ (Hinton, 2007). More formally, a generative model learns the distribution of data $\mathcal{X}$ (defined by its probabilistic density function $p^*(\mathbf{x})$), either explicitly or implicitly (Mohamed and Lakshminarayanan, 2016), by minimising a discrepancy measure between the data distribution $p^*(\mathbf{x})$ and the model distribution $p_\theta(\mathbf{x})$ — a density estimation problem. Here I focus on parametric models that explicitly model such probability density functions, and use KL-divergence as the probability distance measure.

In this case, training a generative model involves adjusting its parameters θ, so that the observed data is *likely* to have been generated from the model, which can be formally expressed as increasing the likelihood of the model given data (MacKay, 2003). This is a general task that can be a stepping stone for many other tasks, which makes it an ideal *a priori* candidate of neural computation from synaptic to cognitive levels (Berkes et al., 2011; Griffiths et al., 2010; Pfister et al., 2010; Tenenbaum et al., 2011; Tootoonian and Lengyel, 2014).

Learning a generative model provides a principled way for learning representations. In particular, I will look at generative models with *latent variables*, which explicitly model

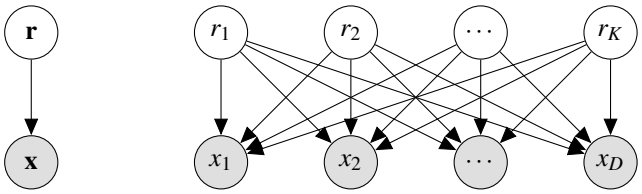


Fig. 2.1 The directed graphical model illustration of generative models discussed in this section. To generate data $\mathbf{x}$, we first sample $\mathbf{r}$ from the prior distribution of the from $p_\theta(\mathbf{r})$, then sample $\mathbf{x}$ from the conditional distribution $p_\theta(\mathbf{x}|\mathbf{r})$.

unobserved factors behind observed data. A generative model with latent variables is illustrated in figure 2.1 (left). While the variable $\mathbf{x}$ represents observable data, the latent variable $\mathbf{r}$ models hidden factors assumed to generate these data. This *directed graphical model* can be interpreted as latent variable *representing* the observed data $\mathbf{x}$ via the conditional distribution $p_\theta(\mathbf{x}|\mathbf{r})$, where θ is the parameter vector. To sample data from this model, we first sample the latent variable from the prior $p_\theta(\mathbf{r})$, then the observable data from $p_\theta(\mathbf{x}|\mathbf{r})$.

Inference is the inverse of generation, which computes the latent variables $\mathbf{r}$ given observation $\mathbf{x}$. This process is of particular interest for studying neural representation: we can treat a neural system as an inference model (also called *recognition model*, Dayan et al. (1995); Hinton et al. (1995)), so that neural activities $\mathbf{r}$, as output of this model, represent the hidden factors of its inputs $\mathbf{x}$. Therefore, the latent variables $\mathbf{r}$ can be interpreted as the neural representation of input $\mathbf{x}$. Under this statistical framework, we can further study the *mutual information* between inputs and their neural representation, which quantifies the information passed from inputs to outputs of the inference model.

This statistical structure makes the latent variable interpretable. For example, $\mathbf{r}$ could be the location of the animal underlying its noisy sensory input $\mathbf{x}$, or the category of a hand-written digit when $\mathbf{x}$ is an image of the hand-written digit. In addition to their explanatory ability, generative models naturally support unsupervised learning. While labelled data are usually scarce for both artificial and biological agents, unsupervised learning can efficiently utilise unlabelled data. As a result, unsupervised learning has been hypothesised to play a crucial role in neural computation (e.g. Barlow, 1989; Bell and Sejnowski, 1995; Hinton et al., 1995).

Formally, it has been shown that learning a feed-forward generative model leads to increased mutual information between the model representations and data (Barber and Agakov, 2004). I will extend this *infomax* argument to RNNs and sequence learning in section 6.1. The information gained in this process has been demonstrated to facilitate tasks such as pattern recognition (Hinton, 2007). Perhaps for the same reason, generation is

well-known to be an indispensable part of education. For instance, in the practice of *solfeggio* in music education, singing (generating correct sound) is an pathway to discriminating and recognising complicated musical sound, which provides the foundation for further instrumental training. Therefore, although this thesis only focuses on the computation of generation, the results are relevant for a wider range of tasks.

The examples I shall use throughout this thesis, prediction and recall of temporal sequences, are relevant in many aspects of life. Learning language, music, navigation and planning are all closely related to such sequence learning. In particular, several decades of experiments on sequence learning, across different species and brain regions, provide abundant high quality data waiting for more thorough theoretical investigation. Appendix A shows that sequence learning can be formulated as a generation task.

2.3 Understanding artificial neural networks as generative models

Pioneered by connectionists, there has been a long history of using artificial neural networks (ANNs) to understand neural systems in the brain (Mante et al., 2013; Marr, 1991; Olshausen and Field, 1996; Rumelhart et al., 1987). Most of these studies treat neural networks as black-boxes when optimising them for the performance on certain tasks (Sussillo and Barak, 2013). The resulting neural representations from these neural networks are then compared with recorded neural data. This approach has its advantages and disadvantages, ironically, for the same reason. On one hand, since the objective of certain task performance does not specify any criterion on representations *per se*, this approach justifies that the spontaneously emerged representations are computationally optimal for the task. On the other hand, this black-box approach poses the challenge of understanding why such representations are optimal. This thesis aims to go beyond such back-box method, starting from a normative understanding of ANNs.

One way is to understand ANNs is to interpret them as dynamical systems. As recent introduced in their paper with title “Opening the Black Box”, Sussillo and Barak (2013) presented a framework for understanding emerging representations in recurrent ANNs (or simply RNNs) from a dynamical systems perspective. This method is most helpful when the intrinsic dynamics of the system are mostly autonomous. I will review this approach in more detail in Section 4.4, where it is employed to analyse the recall dynamics of my sequence learning model. However, this approach does not generalise well to systems with persistent external input, and is less useful for understanding feed-forward neural networks.

An alternative and more general way of understanding neural networks comes from a statistical point of view (e.g., Barlow, 1989; Bell and Sejnowski, 1995; Field, 1994; Hinton, 1989; Lewicki and Sejnowski, 2000; Olshausen and Field, 1996). The majority of this thesis is devoted to understanding the emergent neural representations, by interpreting the ANNs as generative models. The most basic ANNs are simply deterministic functions parametrised by the weights and biases, which can be seen as analogous of neural networks in the brain (Rumelhart et al., 1987). However, when stochasticity is introduced either explicitly or implicitly, by for example added noise, ANNs can be interpreted as probabilistic models (Gal and Ghahramani, 2015; Hinton and Ghahramani, 1997; MacKay, 1992; Neal, 1996; Teh et al., 2004; Vincent et al., 2008a). This view has enabled interpretation of stochastic ANNs as generative models (Roweis and Ghahramani, 1999; Vincent, 2011), which is further explored in this thesis. Based on recent development in stochastic variational inference (Gal and Ghahramani, 2015; Kingma et al., 2015; Rezende and Mohamed, 2015), I interpret the neural activities as representing a posterior distribution given external stimuli. With this probabilistic interpretation, correspondences can be established between recorded neural activities and the latent variables in models.

2.4 Inference and learning

This section introduces the variational approach as a general framework for inference in generative models (Hoffman et al., 2013; Wainwright and Jordan, 2008). The computational cost of inference across different samples can be amortised via a parametrised inference model, allowing efficient inference for high dimensional data. This inference model is also called a recognition model when implemented by a neural network (Dayan et al., 1995; Hinton et al., 1995; Hinton and Zemel, 1994; Kingma and Welling, 2014; Rezende et al., 2014a). From the perspective of neural coding, we can interpret the inference model as an *encoder*, which transform input into a code. I will discuss the encoding/decoding perspective in more detail in the next section.

As the inverse of generation, inference computes the posterior distribution $p_\theta(\mathbf{r}|\mathbf{x})$ given an observation $\mathbf{x}$ (figure 2.1). In our previous examples, the posterior distribution describes the probability that the animal in certain location, given the noisy sensory input, or the probabilities of categories (the digits) given a hand-written digit in the form of an image. In principle, inference is straightforward using Bayes' rule

$$p_\theta(\mathbf{r}|\mathbf{x}) = \frac{p_\theta(\mathbf{x}|\mathbf{r})p_\theta(\mathbf{r})}{p_\theta(\mathbf{x})} \tag{2.1}$$

However, computing the marginal probability $p_\theta(\mathbf{x})$, as the normalising constant, requires integrating or summing over all possible configurations of the representation $\mathbf{r}'$

$$p_\theta(\mathbf{x}) = \int p_\theta\left(\mathbf{x}|\mathbf{r}'\right) p_\theta\left(\mathbf{r}'\right) \mathrm{d}\mathbf{r}' \tag{2.2}$$

This operation is generally intractable. If $\mathbf{r}'$ is continuous, the integral in equation 2.2 is not analytically tractable except in a few cases such as linear Gaussian models. If the of $\mathbf{r}'$ is discrete, equation 2.2 requires summation over states exponential in the dimensionality of $\mathbf{r}'$.

Despite its intractability, $p_\theta(\mathbf{x})$ is an important quantity. It is the likelihood function of the generative model when we consider $p_\theta(\mathbf{x})$ as a function of the parameters θ (fixing $\mathbf{x}$). A commonly used objective for training a generative model is the log-likelihood given all the data $\boldsymbol{X} = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$:

$$\theta \leftarrow \underset{\theta}{\operatorname{argmax}} \ln p_\theta(\mathbf{X}) \tag{2.3}$$

because a high likelihood indicates the observed data are likely to be generated from this model. Taking the assumption that data points are identically and independently distributed (iid) [1] , this objective is equivalent to the expectation:

$$\begin{aligned} \ln p_\theta(\boldsymbol{X}) &= \sum_{n=1}^{N} \ln p_\theta(\mathbf{x}_n) \\ &\propto \langle \ln p_\theta(\mathbf{x}) \rangle_{p^*(\mathbf{x})} \end{aligned} \tag{2.4}$$

where $p^*(\mathbf{x})$ is the (unknown) distribution of the data. For brevity, we may omit the averaging over the data in the following text. Since computing $p_\theta(\mathbf{x})$ is usually intractable, it is infeasible to directly use equation 2.3 for learning. We now turn to an alternative objective for maximising the log-likelihood $\ln p_\theta(\mathbf{X})$, which involves approximating the intractable posterior distribution $p_\theta(\mathbf{r}|\mathbf{x})$.

2.4.1 Variational inference and the EM algorithm

When exact inference is intractable, the alternative is approximate inference. The variational approach provides a general approximate inference method by formulating inference as an optimisation problem. For any distribution $q_\phi(\mathbf{r}|\mathbf{x})$ over the representation that is parameterised

[1] This assumption is not always true, in particular for sequential data. Derivation for sequential data and recurrent neural networks can be found in Appendix A.

by ϕ [2] , we have

$$\begin{aligned}\ln p_\theta(\mathbf{x}) &= \int q_\phi(\mathbf{r}|\mathbf{x}) \ln \frac{p_\theta(\mathbf{x},\mathbf{r})}{p_\theta(\mathbf{r}|\mathbf{x})} \mathrm{d}\mathbf{r} \\ &= \underbrace{\int q_\phi(\mathbf{r}|\mathbf{x}) \ln \frac{p_\theta(\mathbf{x},\mathbf{r})}{q_\phi(\mathbf{r}|\mathbf{x})} \mathrm{d}\mathbf{r}}_{\mathcal{L}} + \mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{r}|\mathbf{x}) \| p_\theta(\mathbf{r}|\mathbf{x})\right]\end{aligned} \tag{2.5}$$

where $\mathcal{L}$ is a variational lower-bound of the log-likelihood. This lower-bound is tight when the KL-divergence in equation 2.5 vanishes, that is, when $q_\phi(\mathbf{r}|\mathbf{x}) = p_\theta(\mathbf{r}|\mathbf{x})$. The equivalence is possible only in a few cases when the posterior $p_\theta(\mathbf{r}|\mathbf{x})$ can be computed exactly.

More generally, $q_\phi(\mathbf{r}|\mathbf{x})$ becomes closer to the posterior $p_\theta(\mathbf{r}|\mathbf{x})$ when the KL-divergence $\mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{r}|\mathbf{x}) \| p_\theta(\mathbf{r}|\mathbf{x})\right]$ is minimised. Although directly optimising this KL-divergence is difficult since it contains the intractable posterior, we can minimise it via maximising the lower-bound $\mathcal{L}$ with respect to ϕ — changing ϕ does not affect the log-likelihood $p_\theta(\mathbf{x})$, so $\mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{r}|\mathbf{x}) \| p_\theta(\mathbf{r}|\mathbf{x})\right]$ decreases as $\mathcal{L}$ increases.

On the other hand, this lower-bound also provides an alternative objective for training the model: maximising $\mathcal{L}$ with respect to θ increases the log-likelihood $\ln p_\theta(\mathbf{x})$, assuming the lower bound is already tight after updating ϕ. This alternating optimisation procedure can be summarised as the Expectation - Maximisation (EM) algorithm (Dempster et al., 1977), which performs coordinate descent over the lower-bound $\mathcal{L}$:

E-step Maximising $\mathcal{L}$ with respect to ϕ to improve the approximate posterior $q_\phi(\mathbf{r}|\mathbf{x})$:

$$\phi \leftarrow \underset{\phi}{\operatorname{argmax}}\, \mathcal{L} \tag{2.6}$$

M-step Maximising $\mathcal{L}$ with respect to θ to improve the log-likelihood $p_\theta(\mathbf{x})$:

$$\theta \leftarrow \underset{\theta}{\operatorname{argmax}}\, \mathcal{L} \tag{2.7}$$

As a brief summary, following the convention, I refer to $q_\phi(\mathbf{r}|\mathbf{x})$ the variational distribution, and the model parametrised by ϕ the inference model (Hinton et al., 1995), in contrast to the generative model parametrised by θ. In practice, it is unnecessary to have separated E- and M-steps, since θ and ϕ can be optimised jointly. However, the EM interpretation

[2] In its original formulation, mean-field variational inference uses variational calculus to optimise form-free the factorial distribution $q(\mathbf{z})$ (Wainwright and Jordan, 2008). Further, the conditioning on $\mathbf{x}$ is not necessary here. However, when $q_\phi(\mathbf{r}|\mathbf{x})$ is implemented in neural network as a inference model, this form of conditional distribution explicitly reflects the fact that the inference model takes $\mathbf{x}$ as input.

gives insights into the different roles played by the generative and inference model. As an alternative objective to the log-likelihood $\ln p_\theta(\mathbf{x})$, $\mathcal{L}$ is also called the evidence lower bound objective (ELBO, Wainwright and Jordan, 2008).

Training generative models using the lower-bound $\mathcal{L}$ is efficient. The inference model amortises the otherwise high computation cost from inference. Since once it is trained, sampling from $q_\phi(\mathbf{r}|\mathbf{x})$ simply involves running through the parametrised inference model. As a result, the variational lower-bound can be approximated efficiently using Monte-Carlo methods. [3] For a large family of continuous distributions $q_\phi(\mathbf{r}|\mathbf{x})$, stochastic backpropagation (Rezende et al., 2014a) or the equivalent re-parametrisation trick (Kingma and Welling, 2014) provides an unbiased and minimal variance estimator of the gradient of $\mathcal{L}$.

2.4.2 The variational lower-bound

Before we dive into the details of specific models, this section discusses a few important properties of the lower-bound $\mathcal{L}$, which provide high-level intuitions on the representation $\mathbf{r}$ that emerges from optimising this lower-bound. The lower-bound can be decomposed in two different ways, as noted in equation 2.8 and 2.9:

$$\mathcal{L} = \langle \ln p_\theta(\mathbf{x}|\mathbf{r}) \rangle_{q_\phi(\mathbf{r}|\mathbf{x})} - \mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{r}|\mathbf{x}) \| p_\theta(\mathbf{r})\right] \tag{2.8}$$

$$\mathcal{L} = \langle \ln p_\theta(\mathbf{x}, \mathbf{r}) \rangle_{q_\phi(\mathbf{r}|\mathbf{x})} + \mathrm{H}_\phi\left[\mathbf{r}|\mathbf{x}\right] \tag{2.9}$$

Maximum likelihood

We first look at equation 2.8 from the perspective of learning a generative model. The first term can be interpreted as a *reconstruction error* measured as the likelihood of an inferred $\mathbf{r}$ given the corresponding input $\mathbf{x}$, and the second term as a regulariser that penalises the digression of $q_\phi(\mathbf{r}|\mathbf{x})$ from the prior of the generative model $p_\theta(\mathbf{r})$. These two terms suggest a trade-off in optimising the inference model, which is shared by the underlying true posterior distribution: the approximate posterior $q_\phi(\mathbf{r}|\mathbf{x})$ must balance between both explaining the data via $\langle \ln p_\theta(\mathbf{x}|\mathbf{r}) \rangle_{q_\phi(\mathbf{r}|\mathbf{x})}$ and being consistent with the prior $p_\theta(\mathbf{r})$.

This perspective suggests learning the inference the prior $p_\theta(\mathbf{r})$ term is problematic, since the inference model may over-fit to the data without the regularisation from the prior. This problem presents when one trying to formulate "vanilla" autoencoder models as generative models (section 2.5). As I will show in section 2.6, this problem can be solved either explicitly by regularising (including directly using the KL-divergence as a regulariser as in

[3] In mean-field variational inference, the lower-bound may be analytically tractable if all the conditional distributions are in the exponential family.

equation 2.8) or implicitly via denoising training, where a prior is implicitly specified in the denoising process (section 3.3).

Minimum description length

The minimum description length perspective elegantly shows the connection between learning a generative model and data compression. The negative of $\mathcal{L}$ written in the decomposition of equation 2.8 can be interpreted as the description length (Graves, 2011; Hinton and Zemel, 1994; Hinton and van Camp, 1993; Rissanen, 1978):

$$-\mathcal{L} = \mathcal{D} = \left\langle \mathrm{D}_{\mathrm{KL}} \left[q_\phi(\mathbf{r}|\mathbf{x}) \| p_\theta(\mathbf{r}) \right] \right\rangle_{p_x(\mathbf{x})} - \left\langle \ln p_\theta(\mathbf{x}|\mathbf{r}) \right\rangle_{q_\phi(\mathbf{x}|\mathbf{r}) q_\phi(\mathbf{r})} \tag{2.10}$$

I add the averaging over data distribution $p^*(\mathbf{x})$ for clarity, and changed the factorisation of the joint distribution $q_\phi(\mathbf{x}, \mathbf{r})$ by defining

$$q_\phi(\mathbf{r}|\mathbf{x})\, p^*(\mathbf{x}) = q_\phi(\mathbf{x}, \mathbf{r}) = q_\phi(\mathbf{x}|\mathbf{r})\, q_\phi(\mathbf{r}) \tag{2.11}$$

We can set up the a communication scenario by assuming a message $\mathbf{x}$ is mapped to the code $\mathbf{r}$ by an *encoder* $q_\phi(\mathbf{r}|\mathbf{x})$ (the inference model), and the code $\mathbf{r}$ is then transmitted to the receiver through a noiseless channel. The receiver uses a *decoder* $p_\theta(\mathbf{x}|\mathbf{r})$ (the generative model) to reconstruct $\mathbf{x}$ from $\mathbf{r}$. We further assume that the receiver has its prior distribution of the code $p_\theta(\mathbf{r})$, possibly from its previous experience[4]. Now we consider the problem: how much information[5] is needed to transmit $\mathbf{x}$ *faithfully* to the receiver? Under our previous assumptions, the information to transmit has two parts, corresponding to the two terms in equation 2.10:

1. The nats required to specify the code given the receiver's knowledge of $p_\theta(\mathbf{r})$. This is the complexity cost. It is desirable to have a less complex code, but a code with zero complexity can not transmit any information unknown to the receiver.

2. The *extra* nats to specify $\mathbf{x}$, in addition to the code $\mathbf{r}$. This is the reconstruction cost. When this cost is zero, the code $\mathbf{r}$ conveys all the information about $\mathbf{x}$, so no extra nats are needed.

These two parts correspond exactly to the two terms in equation 2.10. According to Shannon's source coding theorem (Shannon, 1948), assuming that $\mathbf{x}$ comes from the encoder distribution

[4] Of course, this prior may not be the same as the marginal distribution from the encoder's view, $q_\phi(\mathbf{r})$.
[5] Information is measured in nats, since natural log is used throughout this thesis.

$q_\phi(\mathbf{x}|\mathbf{r})$, $-\langle \ln p_\theta(\mathbf{x}|\mathbf{r})\rangle_{q_\phi(\mathbf{x}|\mathbf{r})}$ is a lower bound of expected nats to transmit $\mathbf{x}$ under the decoder distribution $p_\theta(\mathbf{x}|\mathbf{r})$. This is shown by Gibbs' inequality:

$$\begin{aligned} -\langle \ln p_\theta(\mathbf{x}|\mathbf{r})\rangle_{q_\phi(\mathbf{x}|\mathbf{r})} &\leqslant -\left\langle \ln q_\phi(\mathbf{x}|\mathbf{r})\right\rangle_{q_\phi(\mathbf{x}|\mathbf{r})} \\ &= \mathrm{H}_\phi[\mathbf{x}|\mathbf{r}] \end{aligned} \tag{2.12}$$

where $\mathrm{H}_\phi[\mathbf{x}|\mathbf{r}]$ is the conditional entropy as the minimal nats needed to transmit $\mathbf{x}$ from the encoder distribution. The second term, $\mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{r}|\mathbf{x})\|p_\theta(\mathbf{r})\right]$, is better interpreted here as the *relative entropy* or *information gain*, which is the amount of information needed to describe $q_\phi(\mathbf{r}|\mathbf{x})$ when $p_\theta(\mathbf{r})$ is known.

Helmholtz free energy

Alternatively, the decomposition in equation 2.9 has the form of the (negative) Helmholtz free energy $\mathcal{F}$ as a measure of thermodynamic potential. This form is more often associated with the EM algorithm when the posterior can be computed analytically (Dempster et al., 1977; Roweis and Ghahramani, 1999).

$$\mathcal{F} = \langle E(\mathbf{x},\mathbf{r})\rangle_{q_\phi(\mathbf{r}|\mathbf{x})} + \mathrm{H}_\phi[\mathbf{r}|\mathbf{x}] \tag{2.13}$$

where $E(\mathbf{x},\mathbf{r}) = \ln p_\theta(\mathbf{x},\mathbf{r})$ is the energy of the configuration $(\mathbf{x},\mathbf{r})$. Since the entropy $\mathrm{H}_\phi[\mathbf{r}|\mathbf{x}]$ does not depend on θ, the M-step only needs to maximise $\langle \ln p_\theta(\mathbf{x},\mathbf{r})\rangle_{q_\phi(\mathbf{r}|\mathbf{x})}$. In addition, this physical interpretation gives important intuitions into the learning problem. For example, since the Helmholtz free energy is minimised at equilibrium, it is thus justified to minimise the Helmholtz free energy for target configurations.

2.5 Autoencoders

Now I introduce an important class of models, autoencoders, which I use to connects generative models with neural representations. In later chapters of this thesis, autoencoders will be employed to analyse neural systems. An autoencoder (also called auto-associative network in earlier literature) maps inputs into codes using an encoder, and reconstructs input data from these codes using a decoder. Considering these codes as latent variables, autoencoders are closely related to generative models, and in particular the variational method discussed above. This section reviews classical generative models from the perspective of autoencoders.

More formally, the encoder and decoders are functions parametrised by ϕ and θ respectively (equations 2.14 and 2.15). I use the same Greek letters for the inference and generative model as in the previous section to highlight their correspondences. The objective of training an autoencoder is to make the reconstruction $\hat{\mathbf{x}}$ as close to $\mathbf{x}$ as possible. To avoid trivial solutions, such as an identity map, a bottleneck is usually used to constraint the representation $\mathbf{r}$ (Bengio et al., 2012). This bottleneck may be implied in a low dimensional $\mathbf{r}$ or more explicitly as constraints on mutual information (Alemi et al., 2016; Tishby et al., 2000). As will be discussed later, a KL-divergence bottleneck appears when we explicitly consider an autoencoders as a generative model (i.e., a variational autoencoder, VAE).

$$\mathbf{r} = \mathbf{g}_\phi(\mathbf{x}) \tag{2.14}$$

$$\hat{\mathbf{x}} = \mathbf{f}_\theta(\mathbf{r}) \tag{2.15}$$

Here we only consider continuous data $\mathbf{x}$, so squared error is used [6] and the cost function for training is

$$\begin{aligned} \mathcal{C} &= \frac{1}{2}\left\|\mathbf{x}-\hat{\mathbf{x}}\right\|^2 \\ &= \frac{1}{2}\left\|\mathbf{x}-\mathbf{f}_\theta(\mathbf{g}_\phi(\mathbf{x}))\right\|^2 \end{aligned} \tag{2.16}$$

To capture the discrepancy between $\mathbf{x}$ and its reconstruction $\hat{\mathbf{x}}$, we introduce a noise model $\boldsymbol{\varepsilon} \sim \mathcal{N}\left(0, \sigma_f^2 I\right)$. Consequently, equation 2.14 and 2.15 can be formulated as distributions

$$q_\phi(\mathbf{r}|\mathbf{x}) = \delta(\mathbf{r}-\mathbf{g}_\phi(\mathbf{x})) \tag{2.17}$$

$$p_\theta(\mathbf{x}|\mathbf{r}) = \mathcal{N}\left(\mathbf{x}; \mathbf{f}_\theta(\mathbf{r}), \sigma_\mathbf{f}^2 \mathbf{I}\right) \tag{2.18}$$

A Dirac delta function is introduced in equation 2.17 as a result of the deterministic mapping g_ϕ. This choice constrains the approximate posterior distribution $q_\phi(\mathbf{r}|\mathbf{x})$ to be a degenerate distribution as a point-mass, which ignores any multi-modality and uncertainty. Due to the mode-seeking behaviour of variational inference (Bishop, 2006), $q_\phi(\mathbf{r}|\mathbf{x})$ is likely to end-up at a mode of the posterior distribution as a result of optimising the usually non-convex variational lower-bound. Since the delta distribution ignores uncertainty, training with it may result in over-fitting in training, exhibited as for example "running-away" weights. Despite this constrain, the delta distribution is widely used in combination with non-convex optimisation problems for its simplicity (MacKay, 1996; Olshausen and Field, 1996).

[6] For discrete data, cross-entropy error can be used instead (Bishop, 2006).

Fig. 2.2 The directed graphical model illustration of an autoencoder. The encoder (right) is the function $\mathbf{g}_\phi(\mathbf{x}) \rightarrow \mathbf{r}$ parametrised by ϕ (equation 2.14), which maps an input $\mathbf{x}$ to its representation $\mathbf{r}$. The decoder (right) is the function $\mathbf{f}_\theta(\mathbf{r}) \rightarrow \hat{\mathbf{x}}$ parametrised by θ (equation 2.15), which maps a representation $\mathbf{r}$ back to a reconstruction of the input $\hat{\mathbf{x}}$.

Autoencoders are closely related to generative models. The conditional distribution $p_\theta(\mathbf{x}|\mathbf{r})$ of the decoder, as specified in equation 2.18, is ready to be interpreted as the generative model distribution as in figure 2.1, which generates samples in the data space given a representation $\mathbf{r}$. The conditional distribution $q_\phi(\mathbf{r}|\mathbf{x})$ of the encoder (equation 2.17), on the other hand, corresponds to the inference model distribution. However, compared with a fully-specified generative model, the prior distribution $p_\theta(\mathbf{r})$ is missing in the formulation of "vanilla" autoencoders. Without this term, we cannot compute the variational lower-bound in equation 2.8. As mentioned in the previous section, training this model by merely minimising the likelihood term $\langle \ln p_\theta(\mathbf{x}|\mathbf{r}) \rangle_{q_\phi(\mathbf{r}|\mathbf{x})}$ (note that $\ln p_\theta(\mathbf{x}|\mathbf{r}) \propto -\|\mathbf{x} - \mathbf{f}_\theta(\mathbf{r})\|^2$) may lead to over fit, as the regulariser, the KL-divergence $\mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{r}|\mathbf{x}) \| p_\theta(\mathbf{r})\right]$ depending on the prior $p_\theta(\mathbf{r})$, is missing from the autoencoder's objective. In VAEs (Kingma and Welling, 2014; Rezende et al., 2014a), priors are explicitly introduced to constrain the re-parametrised posterior distributions. Section 2.6 will provide alternative solutions with implicit priors from regularisers.

Autoencoders are useful in many different areas. For example, they are used for data illustration and compression, as inputs for classifiers, or as input in another autoencoder as a way of pre-training deep neural networks (Bengio, 2009; Bengio et al., 2007; Hinton et al., 2006). They are also of particular interest for neuroscientists, mainly because of the representations emerging from unsupervised learning. In particular, the representations from autoencoders were observed to exhibit neuron-like firing patterns (e.g., Bergstra and Bengio, 2009; Vincent et al., 2008b).

To understand why such representations emerge, it is helpful to consider the connections between autoencoders and other simpler statistical models. For this purpose, I review a number of simple statistical models, including Gaussian mixture models (GMM), factor analysis (FA), and independent component analysis (ICA), in the light of their corresponding neural network implementations. As we shall see, these models can be seen as constrained

versions of more general autoencoders. All these models can be trained by optimising the variational lower bound $\mathcal{L}$ (equation 2.8), either by jointly optimising all parameters or in alternating steps using the EM algorithm (section 2.4.1). Therefore, I will skip the learning algorithm for individual models, and focus on their different statistical assumptions. The following review closely follows Roweis and Ghahramani (1999), with emphasis on the learned representations.

2.5.1 Gaussian mixture models

A Gaussian mixture model (GMM) can be written as

$$p_\theta(\mathbf{x}) = \sum_{i=1}^{K} \pi_i \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{2.19}$$

where K is the number of Gaussian distribution components in the mixture. Each Gaussian distribution has its own parameters, mean $\boldsymbol{\mu}_i$ and covariance $\boldsymbol{\Sigma}_i$. π_i is the mixing coefficients for the i'th component, with the constraint that $\sum_{i=1}^{K} \pi_i = 1$. We use $\theta = \left\{\boldsymbol{\pi}, \{\boldsymbol{\mu}_i\}_{i=1}^{K}, \{\boldsymbol{\Sigma}_i\}_{i=1}^{K}\right\}$ to summarise the parameters.

To interpret the GMM as a generative model in figure 2.1, we introduce a categorical (1-in-K) latent variable $\mathbf{r}$ to indicate the component being selected. Only one element of $\mathbf{r}$ can be 1, and all other $K-1$ elements are zeros.

$$p_\theta(\mathbf{r}) = \text{Categorical}(\mathbf{r}) = \prod_{i=1}^{K} \pi_i^{r_i} \tag{2.20}$$

$$p_\theta(\mathbf{x}|\mathbf{r}) = \prod_{i=1}^{K} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)^{r_i} \tag{2.21}$$

The above two equations give the same marginal distribution of $\mathbf{x}$ as in equation 2.19. To generate data, we first sample the component i with $r_i = 1$ from the categorical distribution $\prod_{i=1}^{K} \pi_i^{r_i}$, then sample $\mathbf{x}$ from the i'th Gaussian distribution $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$.

Given a data point, the posterior distribution $p_\theta(\mathbf{r}|\mathbf{x})$ indicates the probability that $\mathbf{x}$ comes from each of the components. Inference for GMM is tractable and can be obtained directly using Bayes' rule (equation 2.1):

$$p(r_i = 1|\mathbf{x}) = \frac{\pi_i \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(\mathbf{x}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)} \tag{2.22}$$

To implement a GMM as an autoencoder, binary hidden neurons are used to represent the 1-in-K code, so neuron r_k has the probability of π_k being turned 'on'. The generative weights from a hidden neuron convey the mean of a mixture. More specifically, $w_{ij} = \mu_i(j)$, where $\mu_i(j)$ is the j'th element of $\boldsymbol{\mu}_i$. The mean of the selected Gaussian is thus computed by the product of the generative weight matrix and the vector of hidden neuron activations, $\boldsymbol{\mu} = \boldsymbol{W}^\intercal \cdot \mathbf{r}$. However, the covariance structure of $\mathbf{x}$ (equation 2.21) cannot be easily computed by neural networks. For simplicity, we constrain all the covariance to have the same spherical shape, so $\boldsymbol{\Sigma}_j = \sigma_f^2 \boldsymbol{I}$, $\forall j$, which is the same as the noise model we assumed in equation 2.18. As discussed above, this assumption gives the log-likelihood function $\ln p_\theta(\mathbf{x}|\mathbf{r})$ in the form proportional to the negative of the reconstruction error (equation 2.16).

To derive the inference model that implements inference, we expand the posterior (equation 2.22)

$$p(r_i = 1|\mathbf{x}) = \frac{\pi_i \cdot |\boldsymbol{\Sigma_i}|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\mathbf{x}^\intercal \boldsymbol{\Sigma}_i^{-1}\mathbf{x} + \boldsymbol{\mu}_i^\intercal \boldsymbol{\Sigma}_i^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_i^\intercal \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i\right\}}{\sum_{j=1}^K \pi_j \cdot \left|\boldsymbol{\Sigma}_j\right|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\mathbf{x}^\intercal \boldsymbol{\Sigma}_j^{-1}\mathbf{x} + \boldsymbol{\mu}_j^\intercal \boldsymbol{\Sigma}_j^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_j^\intercal \boldsymbol{\Sigma}_j^{-1}\boldsymbol{\mu}_j\right\}} \tag{2.23}$$

The presence of 2nd order terms ($-\frac{1}{2}\mathbf{x}^\intercal \Sigma_j^{-1}\mathbf{x}$) suggests the inference model generally needs to be nonlinear, even when the generative model is linear. Under our spherical Gaussian assumption, the posterior can be simplified to

$$p(r_i = 1|\mathbf{x}) = \frac{\exp\left\{\mathbf{V}_i^\intercal \mathbf{x} + \mathbf{b}_i\right\}}{\sum_{j=1}^K \exp\left\{\mathbf{V}_j^\intercal \mathbf{x} + \mathbf{b}_i\right\}} \tag{2.24}$$

where $\mathbf{v}_i = \boldsymbol{\mu}_i^\intercal \boldsymbol{\Sigma}_i^{-1}$ the columns of the recognition weights and $\mathbf{b}_i = -\frac{1}{2}\boldsymbol{\mu}_i^\intercal \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i$ are elements of the bias. Here the non-linearity has the form of a softmax function.

In summary, under the assumption that all the Gaussian components have the same covariance matrix $\sigma_f^2 I$, the GMM is equivalent to an autoencoder with g_ϕ and f_θ with the following format:

$$\mathbf{r} = \mathrm{g}_\phi(\mathbf{x}) \sim \text{Categorical}\left(\text{softmax}(\mathbf{V}^\intercal \mathbf{x} + \mathbf{b})\right) \tag{2.25}$$

$$\hat{\mathbf{x}} = \mathrm{f}_\theta(\mathbf{r}) = \mathbf{W}^\intercal \mathbf{r} \tag{2.26}$$

where the decoder outputs the posterior mean of the full generative model. The GMM presents an example of extremely localist representations (Hinton and Ghahramani, 1997) — only one neuron in the hidden layer can be activated at a time via sampling the categorical distribution. This is a result of the nonlinear operation of sampling the categorical variable $\mathbf{r}$.

By contrast, as we shall see next, extremely distributed neural representations result from models without any nonlinear operation.

2.5.2 Factor analysis

Factor analysis is a class of linear Gaussian models. Principal component analysis (PCA) and probabilistic PCA can be seen as special cases of factor analysis with additional constraints (Roweis and Ghahramani, 1999). Assuming the data has zero mean[7], factor analysis can be described as a generative model (figure 2.1):

$$p_\theta(\mathbf{r}) = \mathcal{N}(\mathbf{r}; 0, \mathbf{I}) \tag{2.27}$$

$$p_\theta(\mathbf{x}|\mathbf{r}) = \mathcal{N}(\mathbf{x}; \mathbf{W}\mathbf{r}, \mathbf{\Sigma}) \tag{2.28}$$

where the covariance matrix $\mathbf{\Sigma}$ is constrained to be diagonal. We can use a spherical Gaussian as the prior $p_\theta(\mathbf{r})$ without loss of generality, since its covariance can be assimilated into the matrix $\mathbf{W}$. It is important to restrict the structure of the noise covariance matrix $\mathbf{\Sigma}$ (more restrictions are imposed on PCA and probabilistic PCA). Otherwise, one could obtain a trivial solution with $\mathbf{W} = 0$ and explain all the covariance from the data using the noise covariance $\mathbf{\Sigma}$. The matrix $\mathbf{W}$ is known as the factor loading matrix, and the diagonal elements of $\mathbf{\Sigma}$ (i.e., the variance along each coordinate) are called uniqueness, since these are the variances unique along each coordinate — all the covariances are captured by $\mathbf{W}$. Similar to GMMs, linear Gaussian models are analytically tractable. We have the marginal and posterior distributions:

$$p_\theta(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mathbf{0}, \mathbf{M}) \tag{2.29}$$

$$p_\theta(\mathbf{r}|\mathbf{x}) = \mathcal{N}\left(\mathbf{r}; \mathbf{W}^\intercal\mathbf{M}^{-1}\mathbf{x}, I - \mathbf{W}^\intercal\mathbf{M}^{-1}\boldsymbol{W}\right) \tag{2.30}$$

where $\mathbf{M} = \mathbf{W}\mathbf{W}^\intercal + \mathbf{\Sigma}$.

Linear Gaussian models have the problem of unidentifiability. To see this, notice that any rotation matrix $\mathbf{R}$ applied to $\mathbf{r}$, resulting in $\mathbf{r}' = \mathbf{R}\mathbf{r}$, would be cancelled in the marginal distribution of $\mathbf{x}$, since the covariance matrix of the marginal distribution $\mathbf{M}' = \mathbf{W}\mathbf{R}\mathbf{R}^\intercal\mathbf{W}^\intercal + \mathbf{\Sigma} = \mathbf{W}\mathbf{W}^\intercal + \mathbf{\Sigma} = \mathbf{M}$ does not change. Because of this rotational invariance, it is impossible to identify the exact coordinate of the latent variable. In fact, unidentifiability stems from the fact that Gaussian models only capture up to second order information. As a result, even though FA assumes independence in $\mathbf{r}$, when higher-order structures are present, FA can only *decorrelate* the representation, while higher order statistics may be unchanged (Field,

[7] I assume zero mean to simplify notations. Another parameter for the mean can be added without this assumption.

1994). Models using non-Gaussian priors, such as ICA (reviewed next), can be used to reduce dependencies in higher order statistics (section 2.5.3).

To implement FA as an autoencoder, $\mathbf{W}$ is readily the generative weight matrix. The posterior mean is a linear function of input $\mathbf{x}$ (equation 2.30), so we have the recognition weights $\mathbf{V} = \mathbf{W}^\top \mathbf{M}^{-1}$. As in GMMs, we assume the noise covariance matrix to be $\sigma_f^2 \mathbf{I}$ for simplicity. Again, this simplified covariance gives rise to a log-likelihood term $\ln p_\theta(\mathbf{x}|\mathbf{r})$ that is proportional to the negative squared reconstruction error. Such a spherical Gaussian noise distribution is actually the assumption used by probabilistic PCA (Roweis, 1998; Tipping and Bishop, 1999).

In general, the posterior covariance matrix is not sparse, as a result of conditional dependencies (explaining away). However, it is easy to show that the posterior covariance matrix $\mathbf{I} - \mathbf{W}^\top \boldsymbol{M}^{-1} \mathbf{w}$ becomes diagonal when the columns of $\mathbf{W}$ are orthogonal to each other. This orthogonality condition is generally not true, but it is always possible to orthogonalise a full rank matrix $\mathbf{W}$ by an invertible matrix $\mathbf{P}$ (e.g., from the Gram-Schmidt process). To maintain the same likelihood $p_\theta(\mathbf{x})$, we simply need to multiply the prior of $\mathbf{r}$ by $\mathbf{P}^{-1}$. Indeed, an orthogonal basis is imposed as a constraint in PCA, in which the columns of $\mathbf{w}$ are the principal components — eigenvectors of the data covariance matrix. PCA also takes the limit $\boldsymbol{\Sigma} \to 0$, where the posterior covariance matrix $\mathbf{I} - \mathbf{W}^\top \mathbf{M}^{-1} \mathbf{w} \to \mathbf{I} - \mathbf{W}^\top (\mathbf{W}\mathbf{W}^\top)^{-1} \mathbf{W}^\top = 0$ collapses to zero. Further, when $\mathbf{w}$ is an orthogonal matrix we can thus tie the generative and recognition weights $\mathbf{V} = \mathbf{W}^\top$ to reduce the number of parameters, as has been used in the pre-training of deep neural networks (Bengio et al., 2007).

In summary, FA, or more precisely probabilistic PCA under our assumption of spherical Gaussian noise model, is equivalent to an autoencoder with the following format:

$$\mathbf{r} = \mathbf{g}_\phi(\mathbf{x}) = \boldsymbol{V}\,\mathbf{x} \tag{2.31}$$

$$\hat{\mathbf{x}} = \mathbf{f}_\theta(\mathbf{r}) = \mathbf{w}\,\mathbf{r} \tag{2.32}$$

In this form, as in the autoencoder for GMM, the decoder only outputs the posterior mean of the full generative model. As discussed above, we can tie the weights using $\mathbf{V} = \boldsymbol{W}^\top$ to reduce the number of parameters and force enforce the PCA solution.

Compared with GMMs, there are only linear operations in both the generative model and recognition model. In the special case of PCA, learning is equivalent to finding eigenvectors of the data covariance matrix. In fact, regardless of the nonlinearity in the encoder function, an autoencoder with a linear decoder is similar to PCA (up to the rotation of basis $\mathbf{W}$) (Baldi and Hornik, 1989; Bourlard and Kamp, 1988). However, nonlinear auto-association is not exactly equivalent to PCA, since the non-linearity allows solutions with multiple local optima

(Japkowicz et al., 2000). Compared with the extremely localist representations from mixture of Gaussians, the representations from factor analysis are highly distributed — activities of hidden neurons form a Gaussian cloud.

2.5.3 Independent component analysis

Independent component analysis (ICA) solves the identifiability problem using non-Gaussian priors that are not invariant to rotation. In addition, it results in sparsely distributed representations that lie between the extremely localist representations from the mixtures of Gaussians and the highly distributed representations from factor analysis. Such representations have been used to explain sensory coding across different brain regions (Bell and Sejnowski, 1997; Field, 1994; Lewicki, 2002; Olshausen and Field, 1996). To continue with our theme of probabilistic generative models, I first introduce ICA through a maximum-likelihood approach (Roweis and Ghahramani, 1999). We then look at the *infomax* (information maximisation) approach which reveals information-theoretical aspects of the neural tuning curves we found in trained models (Bell and Sejnowski, 1995). For simplicity, here I only consider linear ICA.

Maximum-likelihood ICA

As in previous models, we have the generative model (figure 2.1):

$$p_\theta(\mathbf{r}) = \prod_{k=1}^{K} p_\theta(r_k) \tag{2.33}$$

$$p_\theta(\mathbf{x}|\mathbf{r}) = \mathcal{N}(\mathbf{x}; \boldsymbol{W}\mathbf{r}, \boldsymbol{\Sigma}) \tag{2.34}$$

where the conditional distribution (equation 2.34) has the same form as in factor analysis (equation 2.28). To simplify the following derivations, we assume the diagonal noise covariance matrix $\boldsymbol{\Sigma} = \sigma^2 \mathbf{I}$, as used by probabilistic PCA. It is straightforward to extend our derivation to more general diagonal covariance matrices. In principle, the factorial prior can be any non-Gaussian distribution. To obtain sparse activity patterns, we generally use super-Gaussian distributions (distributions with high kurtosis), such as the Laplace distribution, $p_\theta(r_k) = \frac{1}{2b}\exp\left(-\frac{|r_k|}{b}\right)$. However, after introducing the non-Gaussian prior, the posterior distribution $p_\theta(r|x)$ is no longer analytically tractable.

An alternative interpretation of ICA still assumes a Gaussian prior, which is then transform to a non-Gaussian following a non-linear mapping. This view explains why ICA yields representations between Gaussian mixture models and factor analysis — the nonlinearity

is usually not as extreme as the 1-in-K operation (sampling) in Gaussian mixture models, but is also different from the purely linear factor analysis models. ICA is equivalent to an autoencoder with similar a structure as that for FA, except a non-linearity is required in the encoder, as a result of the non-Gaussian prior $p_\theta(\mathbf{r})$. Since the posterior $p_\theta(\mathbf{r}|\mathbf{x})$ is intractable, it is not possible to have a closed-form inference model as for the GMM or FA.

Infomax ICA

The ICA was actually proposed with the alternative objective of maximising the entropy of the representation $\mathrm{H}_\phi[\mathbf{r}]$ (Bell and Sejnowski, 1995). This leads to independent representations, because

$$\mathrm{H}_\phi[\mathbf{r}] \leqslant \sum_{k=1}^{K} \mathrm{H}_\phi[\mathbf{r}_k] \tag{2.35}$$

where the equality holds when r_k are independent, and the bound tightens when the dependencies between r_k decrease. Maximising the entropy $\mathrm{H}_\phi[\mathbf{r}]$ can be achieved by maximising the mutual information $\mathrm{I}_\phi[\mathbf{r};\mathbf{x}]$, since

$$\mathrm{H}_\phi[\mathbf{r}] = \mathrm{I}_\phi[\mathbf{r};\mathbf{x}] + \mathrm{H}_\phi[\mathbf{r}|\mathbf{x}] \tag{2.36}$$

and the conditional entropy $\mathrm{H}_\phi[\mathbf{r}|\mathbf{x}]$ is a constant (though diverges to $-\infty$) when the approximate posterior $q_\phi(\mathbf{r}|\mathbf{x})$ is a delta distribution. This objective is desirable for neural networks, since Barber and Agakov (2004) showed that $\mathrm{I}_\phi[\mathbf{r};\mathbf{x}]$ is maximised from minimising reconstruction errors in an autoencoder setting (see also section 6.1).

As shown later in this thesis (section 4.5), infomax ICA reveals an intriguing feature of the representation in sequence learning. The infomax approach provides important links to neural data, as it provides a form of optimality that can be used to make experimentally testable predictions of tuning curves. Assuming y_k is the *direct* input into hidden neuron k, such that the tuning curve is the monotonic function mapping $f(y_k) \to r_k$ to firing rates. $f(\cdot)$ transforms the distribution of input $p(y_k)$ according to:

$$p(r_k) = \frac{p(y_k)}{|\partial f(y_k)/\partial y_k|} \tag{2.37}$$

With limited activity range r_k, the continuous distribution with maximum entropy is the uniform distribution. Therefore, the tuning curve $f(y_k)$ that results in the maximum entropy $\mathrm{H}_\phi[r_k]$ is the cumulative distribution function (CDF) of $p(y_k)$,

$$f(y_k) = \int_{-\infty}^{y_k} p(t)\mathrm{d}t \tag{2.38}$$

which transforms y_k into uniformly distributed outputs (Laughlin, 2001). A simple way to test the infomax hypothesis is thus to examine whether the tuning curves in a neural network match the shape of the CDF designated by equation 2.38. This is indeed the case as we will see in section 4.5.

2.6 The implicit prior

At the end of this chapter, we consider the prior $p_\theta(\mathbf{r})$ that is used in the bottleneck for autoencoders. Recall that the prior is a necessary component of a generative model; it is involved in computing the variational lower-bound (equation 2.8), as a tractable objective for training the generative model. To begin with, in the limit of $q_\phi(\mathbf{r}|\mathbf{x}) \to \delta(\mathbf{r} - \mathrm{g}_\phi(x))$ becomes a delta distribution, the lower-bound (equation 2.9) becomes

$$\begin{aligned}\mathcal{L} &= \langle \ln p_\theta(\mathbf{x}|\mathbf{r}) + \ln p_\theta(\mathbf{r}) \rangle_{q_\phi(\mathbf{r}|\mathbf{x})} + \mathrm{H}_\phi[\mathbf{r}|\mathbf{x}] \\ &= \langle \ln p_\theta(\mathbf{x}|\mathbf{r}) \rangle_{q_\phi(\mathbf{r}|\mathbf{x})} + \underbrace{\langle \ln p_\theta(\mathbf{r}) \rangle_{q_\phi(\mathbf{r}|\mathbf{x})}}_{J} + c\end{aligned} \tag{2.39}$$

From equation 2.8, the last two terms together form the (negative) KL-divergence $-\mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{r}|\mathbf{x}) \| p_\theta(\mathbf{r})\right]$, while the first term is the (negative) reconstruction error. $c = \mathrm{H}_\phi[\mathbf{r}|\mathbf{x}]$ is a constant that diverges to $-\infty$ as a result of the deterministic inference model. c vanishes when taking gradients with respect to parameters, so minimising the KL-divergence $\mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{r}|\mathbf{x}) \| p_\theta(\mathbf{r})\right]$ is simplified to maximising $J = \langle \ln p_\theta(\mathbf{r}) \rangle_{q_\phi(\mathbf{r}|\mathbf{x})}$, which functions as a regulariser added to the log-likelihood $\langle \ln p_\theta(\mathbf{x}|\mathbf{r}) \rangle_{q_\phi(\mathbf{r}|\mathbf{x})}$. Therefore, the optimal $\mathbf{r}$ that maximises the variational lower bound can be interpreted as the *maximum a priori* (MAP) solution.

For all the different priors I am going to consider next, I will focus on the term $J = \langle \ln p_\theta(\mathbf{r}) \rangle_{q_\phi(\mathbf{r}|\mathbf{x})\, p^*(x)}$, which is the only term in equation 2.8 that depends on the prior (here I reintroduced the averaging over $p^*(x)$ explicitly in our notation). From the perspective of hierarchical modelling, we can treat $\mathbf{r}$ (provided by $q_\phi(\mathbf{r}|\mathbf{x})$) as inputs into the prior model $p_\theta(\mathbf{r})$. Therefore, J can be seen as the log-likelihood of the prior model.

2.6.1 Uniform priors

When the generative prior is a uniform distribution, or $p_\theta(\mathbf{r}) = \mathbf{u}$ where $\mathbf{u}$ is a constant, the term J becomes a constant:

$$J = \langle \ln p_\theta(\mathbf{r}) \rangle_{q_\phi(\mathbf{r}|\mathbf{x})\, p^*(\mathbf{x})} = \ln u \tag{2.40}$$

Therefore, maximising the likelihood, or simply minimising the squared cost (equation 2.16), is equivalent to maximising the variational lower bound when the generative model has a uniform prior. However, the uniform prior can be unsuitable, since it assigns equal probabilities to all $\mathbf{r}$.

For example, when a population of 100 neurons are used to represent the location of an animal in a 2-dimensional space, the underlying dimensionality of $\mathbf{r}$ is usually much closer to 2 than 100. It is thus inappropriate to assume equal probability everywhere in the possibly very high dimensional space of representations. Even though, mathematically, uniformly distributed representations can be transformed by the generative model into arbitrary desired distributions, it is a waste of valuable (communication) resources to employ such a code.

2.6.2 Single Gaussian priors

We next consider a more informative but still very simple prior, a spherical Gaussian distribution $p_\theta(\mathbf{r}) = \mathcal{N}\left(\mathbf{r}; \mu_r, \sigma_r^2 I\right)$. Now J has the form:

$$\begin{aligned} J = \langle \ln p_\theta(\mathbf{r}) \rangle_{q_\phi(\mathbf{r}|\mathbf{x})\, p^*(\mathbf{x})} &= c + \left\langle \frac{1}{2\sigma_r^2} \int q_\phi(\mathbf{r}|\mathbf{x}) \left\|\mathbf{r} - \mu_r\right\|^2 \mathrm{d}\mathbf{r} \right\rangle_{p^*(\mathbf{x})} \\ &= c + \frac{1}{2\sigma_r^2} \left\langle \left\| g_\phi(\mathbf{x}) - \mu_r \right\|^2 \right\rangle_{p^*(\mathbf{x})} \end{aligned} \tag{2.41}$$

where c comes from the normalising constant. Linear Gaussian models, including Factor Analysis which I discussed in section 2.5.2, use this form of prior. In addition to the linear ICA view presented earlier, ICA can also be seen as a Gaussian priors model with non-linear transform. Although this formulation still has the non-identifiability problem that associated Gaussian distribution, it inspired recent advances including flow-based neural generative models (Dinh et al., 2014). Due to its convenient form, it is also the default choice of prior for more recent models such as VAEs (Kingma and Welling, 2014; Rezende et al., 2014b).

Using the fact that $q_\phi(\mathbf{r}|\mathbf{x})$ is a delta function, equation 2.41 reveals that this Gaussian prior is equivalent to an L2 regulariser on the activity $\mathbf{r} = g_\phi(\mathbf{x})$ around the mean μ_r, which restricts the magnitude of $\mathbf{r}$. Without loss of generality, we can fix the mean as $\mu_r = 0$, since the bias term in the recognition model $g_\phi(\mathbf{x})$ can cancel other values of μ_r. This objective is easy to optimise through backpropagation. However, the assumption that $\mathbf{r}$ is a Gaussian distribution may be too restrictive, as it cannot capture complex structures such as multi-modality. In practice, I found no improvement of generalisation by training with this regularisation term.

2.6.3 New View: Approximate Gaussian Mixture priors

This section presents a novel contribution of this thesis: I show that a simple contractive regulariser (Rifai et al., 2011) can approximate the (negative log-) likelihood of the generative model corresponding to a an autoencoder using a GMM prior (section 2.5.1). This connection suggests a practical and efficient way to use a GMM prior in the variational lower-bound $\mathcal{L}$ without explicit fitting the GMM. This approach combines the flexibility of GMM priors and the computational efficiency from a simple differentiable regulariser. This result will be further exploited to interpret denoising training as implicitly imposing a GMM prior (section 3.3).

Unlike the uniform prior and single Gaussian prior we have seen so far, the GMM as prior model also need to be optimised in maximising J, as a part of maximising $\mathcal{L}$. Here maximising J with respect to its parameters (which we include in θ) is mechanistically similar to maximising the likelihood of the generative model as in section 2.4, which can be done by maximising its own lower bound $\mathcal{L}_J$ (with subscript J) as in section 2.5.1. Therefore, we can similarly use the EM algorithm to maximise J. For the rest of this section, the terms E_J and M_J steps will refer to the steps of this inner EM procedure optimising J, which itself is embedded in an outer loop that optimises the lower bound of the full model (equation 2.39). Note we only use this "inner" EM algorithm to motivate the following derivation; in practice, this procedure will be simplified as we approximate the GMM prior.

More formally, the derivation in this section will show that J in equation 2.39 can be efficiently approximated by the regulariser:

$$J \approx \lambda \, \mathcal{R}_M = \lambda \left\langle \left\| \frac{\partial \mathbf{g}_\phi(\mathbf{x})}{\partial \mathbf{x}} \right\|_F^2 \right\rangle_{p^*(\mathbf{x})} \tag{2.42}$$

where λ is a constant controlling the strength of the regularisation, and the Frobenius norm of a matrix A is defined as:

$$\|\mathbf{A}\|_F = \sqrt{\sum_{ij} |A_{ij}|^2} \tag{2.43}$$

First, I introduce the constrained GMM as the prior model is defined as:

$$p_\theta(\mathbf{r}) = \frac{1}{K} \sum_{k=1}^{K} \mathcal{N}\left(\mathbf{r}; \boldsymbol{\mu}_k, \sigma^2 \mathbf{I}\right) \tag{2.44}$$

It has uniform mixing coefficients $\frac{1}{K}$ and tied variance σ^2 for all K components. As we will see, parameters of this GMM, including K and $\boldsymbol{\mu}_k$ will be cancelled or absorbed into the regularisation weight λ. Here I consider a sequential update process for the GMM's

parameters, and use subscript n to index the total N samples when necessary. Note that several approximations are used in the following derivation. The quality of these approximations is analysed at the end of this section.

Now I analyse fitting the GMM prior model from the perspective of the (inner-loop) EM algorithm. As a generic optimisation algorithm, the EM operation here can be seen as a part of end-to-end model optimisation (e.g., section 3.2), performing coordinate descent/ascent over parameters of this prior model. Recall that here we take as inputs the representations $\{\mathbf{r}_n\}_{n=1}^N$, which are projections of data $\{\mathbf{x}_n\}_{n=1}^N$ by $\mathbf{g}_\phi(\cdot)$. As in the derivation of the original variational lower-bound (Section 2.4.2), J can also be decomposed as:

$$J = \Big\langle \langle \ln p_\theta(\mathbf{r},\mathbf{v})\rangle_{q(\mathbf{v}|\mathbf{r})} - \mathrm{H}\left[q(\mathbf{v}|\mathbf{r})\right] + \mathrm{D}_{\mathrm{KL}}\left[q(\mathbf{v}|\mathbf{r})\|p_\theta(\mathbf{v}|\mathbf{r})\right]\Big\rangle_{q_\phi(\mathbf{r}|\mathbf{x})\,p^*(\mathbf{x})} \tag{2.45}$$

where I introduce a new hierarchical latent variable $\mathbf{v}$ indicating the assignment of mixture components. Its distribution is given by the categorical distribution $q(\mathbf{v}|\mathbf{r})$.

In the inner E_J-step, we minimise the KL-divergence to zero by setting $q(\mathbf{v}|\mathbf{x}) = p_\theta(\mathbf{v}|\mathbf{r})$. This is equivalent to computing the responsibility γ_{nk} for the n'th data point n and the k'th mixture component, as introduced in Section 2.5.1:

$$\gamma_{nk} = \frac{\mathcal{N}\left(\mathbf{r}_n;\boldsymbol{\mu}_k,\sigma^2\mathbf{I}\right)}{\sum_{k=1}^K \mathcal{N}\left(\mathbf{r}_n;\boldsymbol{\mu}_k;\sigma^2\mathbf{I}\right)} \tag{2.46}$$

This soft assignment of responsibility can be approximated by *hard assignment*:

$$\gamma_{nk} \approx \begin{cases} 1, \text{for } k = \operatorname{argmin} \|\boldsymbol{\mu}_k - \mathbf{r}_n\|_2^2 \\ 0, \text{otherwise} \end{cases} \tag{2.47}$$

This approximation is accurate when $\sigma^2 \to 0$ — which as we shall discuss more later, determines the quality of other parts of the approximation as well. However, the KL-divergence $\mathrm{D}_{\mathrm{KL}}\left[q(\mathbf{v}|\mathbf{r})\|p_\theta(\mathbf{v}|\mathbf{r})\right]$ is no longer 0 with this approximation, we therefore have

$$\begin{aligned} J \geqslant \mathcal{L}_J &= \Big\langle \langle \ln p_\theta(\mathbf{r},\mathbf{v})\rangle_{q(\mathbf{v}|\mathbf{r})} - \mathrm{H}\left[q(\mathbf{v}|\mathbf{r})\right]\Big\rangle_{q_\phi(\mathbf{r}|\mathbf{x})\,p^*(\mathbf{x})} \\ &= \Big\langle \langle \ln p_\theta(\mathbf{r},\mathbf{v})\rangle_{q(\mathbf{v}|\mathbf{r})}\Big\rangle_{q_\phi(\mathbf{r}|\mathbf{x})\,p^*(\mathbf{x})} + \text{constant} \end{aligned} \tag{2.48}$$

Using the index of component k obtained in the above E_J-step, the M_J-step then updates the parameters of the generative model θ — here the parameters of the GMM prior — by maximising the lower bound $\mathcal{L}_J$. Since the entropy term is unrelated to the prior, it is ignored as a constant in the M-step, so we only need to maximise $\langle \ln p_\theta(\mathbf{r},\mathbf{v})\rangle_{q_\phi(\mathbf{r}|\mathbf{x})\,p^*(\mathbf{x})\,q(\mathbf{v}|\mathbf{r})}$. As in

Data: N data points $\{\mathbf{r}_n\}_{n=1}^N$

1 Run the K-means algorithm with $\{\mathbf{r}_n\}_{n=1}^N$ to obtain the means $\{\boldsymbol{\mu}_k\}_{k=1}^K$ and assignments γ_{nk} equation 2.47;

2 Compute the variance $\sigma^2 = \frac{1}{ND}\sum_{n=1}^N \gamma_{nk} \sum_{d=1}^D \left(r_n(d) - \mu_k(d)\right)^2$;

3 Evaluate the average likelihood $\frac{1}{N}\sum_{n=1}^N \ln \frac{1}{K}\sum_{k=1}^K \mathcal{N}\left(\mathbf{r}_n; \boldsymbol{\mu}_k, \sigma^2\right)$;

Algorithm 1: Fitting a constrained Gaussian Mixture Model as the Prior model for $\mathbf{r}$.

Section 2.5.1, this involves computing the means:

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{n=1}^{N_k} \gamma_{nk}\,\mathbf{r}_n \tag{2.49}$$

where

$$N_k = \sum_{n=1}^{N} \gamma_{nk} \tag{2.50}$$

is the number of samples belonging to the k'th component. When using the hard assignment, this procedure is equivalent to the K-means clustering algorithm, which assigns a data point to its nearest centroids $\boldsymbol{\mu}_k$. For this reason, K-means is known as a special case of GMM fitted by maximum likelihood (Bishop, 2006). When converged, this algorithm produces the K centroids $\boldsymbol{\mu}_k$. We can further compute the maximum likelihood estimator of the scalar covariance as:

$$\sigma^2 = \frac{1}{ND}\sum_{n=1}^{N} \gamma_{nk} \sum_{d=1}^{D} \left(r_n(d) - \mu_k(d)\right)^2 \tag{2.51}$$

Recall that D is $\mathbf{r}$'s dimensionality. This procedure is summarised in Algorithm 1.

Once the fitting of the GMM prior model finished, we can optimise the lower-bound of the full model $\mathcal{L}$ in equation 2.10 using J's' lower-bound $\mathcal{L}_J$ in equation 2.48. We now analyse this term:

$$\left\langle \ln p_\theta\left(\mathbf{r}_n, \mathbf{v}\right)\right\rangle_{q(\mathbf{v}_n|\mathbf{r}_n)} = \ln \sum_{k=1}^{K} \gamma_{nk}\mathcal{N}\left(\mathbf{r}_n; \boldsymbol{\mu}_k, \sigma^2\right) = \ln \mathcal{N}\left(\mathbf{r}_{nk}; \boldsymbol{\mu}_k, \sigma^2\right) \tag{2.52}$$

The last equality dropped the summation by introducing the index k for $\mathbf{r}$, so that $\mathbf{r}_{nk} = \mathbf{g}_\phi\left(\mathbf{x}_{nk}\right)$ is assigned to component k in the E_J-step. Ignoring additive constant terms, which will

disappear when taking derivatives, we can expand equation 2.52 as:

$$\begin{aligned}
\ln \mathcal{N}\left(\mathbf{r}_{nk}; \boldsymbol{\mu}_k, \sigma^2\right) &= -\frac{\left\|\mathbf{r}_{nk} - \boldsymbol{\mu}_k\right\|^2}{\sigma^2} + \dots \\
&\approx -\frac{\left\|\mathbf{g}_\phi(\mathbf{x}_{nk}) - \mathbf{g}_\phi(\mathbf{x}_{\boldsymbol{\mu}_k})\right\|^2}{\sigma^2} + \dots \\
&\approx -\frac{1}{\sigma^2} \cdot \left[\mathbf{g}'_\phi(\mathbf{x}_{\boldsymbol{\mu}_k})\left(\mathbf{x}_{nk} - \mathbf{x}_{\boldsymbol{\mu}_k}\right)\right]^T \cdot \left[\mathbf{g}'_\phi(\mathbf{x}_{\boldsymbol{\mu}_k})\left(\mathbf{x}_{nk} - \mathbf{x}_{\boldsymbol{\mu}_k}\right)\right] + \dots \\
&= -\frac{1}{\sigma^2} \cdot \left(\mathbf{x}_{nk} - \mathbf{x}_{\boldsymbol{\mu}_k}\right)^T \left[\mathbf{g}'_\phi(\mathbf{x}_{\boldsymbol{\mu}_k})^T \cdot \mathbf{g}'_\phi(\mathbf{x}_{\boldsymbol{\mu}_k})\right] \left(\mathbf{x}_{nk} - \mathbf{x}_{\boldsymbol{\mu}_k}\right) + \dots
\end{aligned} \tag{2.53}$$

In the second line, we assumes that there exist the image $\mathbf{x}_{\boldsymbol{\mu}_k} = \mathbf{g}_\phi^{-1}(\boldsymbol{\mu}_k)$. This is a reasonable assumption for our model, since a decoder $\mathbf{f}_\theta$ is jointly trained with $\mathbf{g}_\phi$ to enforce the mapping for being invertible. Training an autoencoder to approximate bijection has also been used in models including generative moment matching networks (Li et al., 2015). The third line uses the second-order Taylor series approximation around $\mathbf{x}_{\boldsymbol{\mu}_k}$:

$$\mathbf{g}_\phi(\mathbf{x}_{nk}) \approx \mathbf{g}_\phi(\mathbf{x}_{\mu_k}) + \mathbf{g}'_\phi(\mathbf{x}_{\boldsymbol{\mu}_k})(\mathbf{x}_{nk} - \mathbf{x}_{\boldsymbol{\mu}_k}) \tag{2.54}$$

Since $\mathbf{r}_n$ is close to $\boldsymbol{\mu}_k$ as long as the GMM model is well-fitted (which we assume for now), under mild smoothness assumption of $\mathbf{g}_\phi$, $\mathbf{x}_{nk}$ is also close to $\mathbf{x}_{\boldsymbol{\mu}_k}$. Therefore, the error of this approximation is low when $\mathbf{g}_\phi$ is approximately linear in the vicinity of $\mathbf{x}_{\boldsymbol{\mu}_k}$. This error can be reduced when the number of components K is large, so that each component of the GMM prior only needs to cover a small neighbourhood region around $\boldsymbol{\mu}_k$.

In the outer-loop M-step, we maximise the expectation of equation 2.53 over samples $\mathbf{x}_n$. Note that once the E-step is done, $\mathbf{g}'_\phi(\mathbf{x}_{\boldsymbol{\mu}_k})$ is treated as a constant, since $\boldsymbol{\mu}_k$, and thus $\mathbf{x}_{\boldsymbol{\mu}_k}$, is fixed in the subsequent M-step. As a result, we regard the symmetric matrix $\left[\mathbf{g}'_\phi(\mathbf{x}_{\boldsymbol{\mu}_k})^T \cdot \mathbf{g}'_\phi(\mathbf{x}_{\boldsymbol{\mu}_k})\right]$ as a constant matrix, and $(\mathbf{x}_{nk} - \mathbf{x}_{\boldsymbol{\mu}_k})$ as a random variable. This expectation can then be simplified using the equality:

$$\left\langle \mathbf{x}^T \mathbf{A}\, \mathbf{x} \right\rangle = \mathrm{Tr}\left(\mathbf{A}\, \mathbf{C}\right) + \mathbf{m}^T \mathbf{A}\, \mathbf{m} \tag{2.55}$$

for symmetric matrix A, and random variable $\mathbf{x}$ with first moment $\mathbf{m}$ and second central moment $\mathbf{C}$. The second term can be dropped, since from equation 2.54:

$$\left\langle \left(\mathbf{x}_{nk} - \mathbf{x}_{\boldsymbol{\mu}_k}\right) \right\rangle \approx \frac{1}{\mathbf{g}'_\phi(\mathbf{x}_{\boldsymbol{\mu}_k})} \left\langle \mathbf{g}_\phi(\mathbf{x}_{nk}) - \mathbf{g}_\phi(\mathbf{x}_{\mu_k}) \right\rangle = \frac{1}{\mathbf{g}'_\phi(\mathbf{x}_{\boldsymbol{\mu}_k})} \left\langle \mathbf{r}_{nk} - \boldsymbol{\mu}_k \right\rangle = 0 \tag{2.56}$$

because $\boldsymbol{\mu}_k = \mathbf{g}_\phi(\mathbf{x}_{\mu_k})$ is the mean of $\mathbf{r}_{nk}$ by our definition. We further assume that $\mathbf{x}_{nk}$ corresponding to one cluster of $\mathbf{r}_{nk}$ are decorrelated (i.e., a mild conditional independence assumption given the component k), so that $(\mathbf{x}_{nk} - \mathbf{x}_{\boldsymbol{\mu}_k})$ has diagonal covariance matrix $C = \sigma_x^2 \mathbf{I}$. This assumption means that correlations in $\mathbf{x}_{nk}$ are all explained by the correlations between centroids $\mathbf{x}_{\mu_k}$. This scenario is more likely when the number of mixture component is large, so the centroids encodes most information including covariance. For non-delta $q_\phi(\mathbf{r}|\mathbf{x})$, the noise independent of $\mathbf{x}$ may additionally weakens the correlation. As a result, according to the previous two equations:

$$\begin{aligned}\left\langle \ln \mathcal{N}\left(\mathbf{r}_{nk}; \boldsymbol{\mu}_k, \sigma^2\right)\right\rangle_{p_k(\mathbf{x}_{nk})} &\approx -\frac{\sigma_x^2}{\sigma^2} \cdot \operatorname{Tr}\left(\mathbf{g}'_\phi(\mathbf{x}_{\boldsymbol{\mu}_k})^T \cdot \mathbf{g}'_\phi(\mathbf{x}_{\boldsymbol{\mu}_k})\right) \\ &= -\frac{\sigma_x^2}{\sigma^2} \cdot \left\| \mathbf{g}'_\phi(\mathbf{x}_{\boldsymbol{\mu}_k}) \right\|_F^2\end{aligned} \tag{2.57}$$

where p_k indicates the distribution of the neighbour images for the k'th mixture component. The last line uses the Frobenius norm identity $\|A\|_F^2 = \operatorname{Tr}\left(AA^T\right)$.

Following the same vicinity assumption from taking the Taylor expansion (equation 2.54), we can assume $\mathbf{g}'_\phi(\mathbf{x}_{nk}) \approx \mathbf{g}'_\phi(\mathbf{x}_{\boldsymbol{\mu}_k})$. This looks like a strong assumption; however, the regulariser itself directly encourages the smoothness of $\mathbf{g}_\phi(\cdot)$, so that $\mathbf{g}'_\phi(\mathbf{x}_{nk}) \approx \mathbf{g}'_\phi(\mathbf{x}_{\boldsymbol{\mu}_k})$ for $\mathbf{x}_{nk} \approx \mathbf{x}_{\boldsymbol{\mu}_k}$. In other words, training with the contractive regulariser justifies this assumption. Therefore,

$$\left\langle \ln \mathcal{N}\left(\mathbf{r}_{nk}; \boldsymbol{\mu}_k, \sigma^2\right)\right\rangle_{p_k(\mathbf{x}_{nk})} \approx -\frac{\sigma_x^2}{\sigma^2} \cdot \left\| \mathbf{g}'_\phi(\mathbf{x}_{nk}) \right\|_F^2 \tag{2.58}$$

Since no term in equation 2.58 depends on the particular cluster k, the same formula can be applied to other clusters, resulting in the regulariser as in equation 2.42, after removing the negative sign and taking expectation over all $\mathbf{x}_n$:

$$\frac{\sigma_x^2}{\sigma^2} \cdot \left\langle \left\| \frac{\partial \mathbf{g}_\phi(\mathbf{x})}{\partial \mathbf{x}} \right\|_F^2 \right\rangle_{p^*(\mathbf{x})} = \lambda \cdot \left\langle \left\| \frac{\partial \mathbf{g}_\phi(\mathbf{x})}{\partial \mathbf{x}} \right\|_F^2 \right\rangle_{p^*(\mathbf{x})} = \lambda \, \mathcal{R}_M \tag{2.59}$$

Therefore, the above derivation shows that the lower-bound of J under a GMM prior can be approximated by a contractive regulariser. The form of equation 2.59 suggests this regulariser penalises the magnitude of the Jacobian $\frac{\partial \mathbf{g}_\phi(\mathbf{x})}{\partial \mathbf{x}}$, thus encouraging $\mathbf{g}_\phi(\mathbf{x})$ to be smooth. From the perspective of probabilistic modelling, GMM is consistent with this interpretation. In a GMM, data close together are clustered together. For smoothly varying input $\mathbf{x}$, such clustering can be achieved by mapping nearby $\mathbf{x}$'s' to nearby output $\mathbf{g}_\phi(\mathbf{x})$'s, hence the smoothness of $\mathbf{g}_\phi$. This interpretation of smoothness regulariser is also consistent

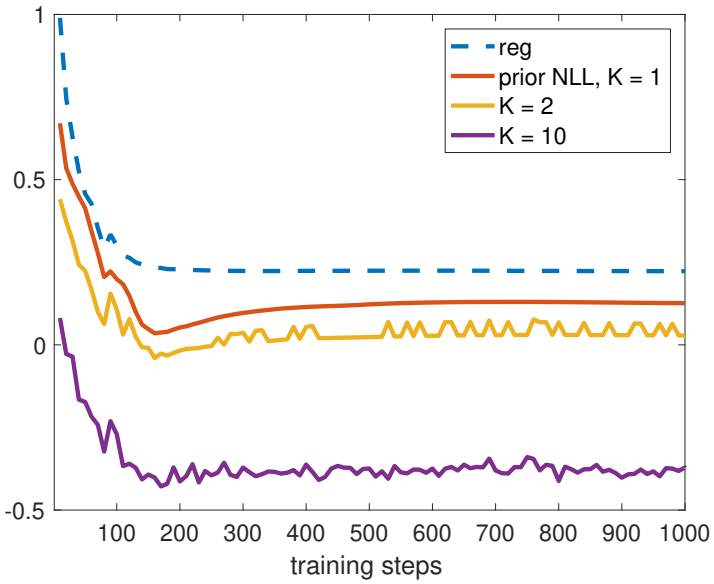


Fig. 2.3 The regularisation loss and the value of J (prior NNL) during training. A recurrent network was trained for sequence learning (Chapter 3) using the regulariser in equation 2.59. The regularisation loss is shown by the blue dashed line. During training, after every 10 steps, GMM prior models with different numbers of components K were fitted to the representation $\mathbf{r}$. The NLL is scaled by a factor of $\frac{1}{300}$ so the scales are comparable with the regularisation loss.

with the motivation from Rifai et al. (2011), who proposed the contractive regulariser to explicitly facilitate invariance of the representation.

Analysis of the Approximation

Two main approximations are used in deriving equation 2.54 and 2.58; both of them rely on local linear approximations of the encoding function $\mathbf{g}_\phi$ or its derivative. Therefore, the approximation is more accurate when the neighbouring region around each mixture mean is small, which can be realised by increasing the number of mixture component K, so the space of $\mathbf{r}$ is clustered into a larger number of smaller regions. Although K does not appear in equation 2.59, larger K generally results in smaller σ_x^2, since the variance within each component will be smaller for smaller regions. On the other hand, for a fixed σ_x^2 the strength of regularisation is inversely proportional to the variance of mixture components σ^2. In the limit of $\sigma^2 \to \infty$, the regulariser vanishes since the prior with infinite variance provides no constraint.

In practice, the two variance parameters σ_x^2 and σ^2 are merged into a single parameter λ that directly controls the strength of regularisation. In the extreme case of $K = N$, the same as the number of data points, $\sigma_x^2 \to 0$ and the regulariser again vanishes when every $\mathbf{r}$ has its own cluster. In another extreme of $K = 1$, $\mathbf{r}$ are strongly regularised to a single mode. This case is equivalent to the single Gaussian prior discussed in section 2.6.2. Instead of choosing K, the regulariser in equation 2.59 allows us to smoothly changing the regularisation weight λ.

To verify the analysis, I measured the value of J during training in figure 2.3. The RNN model used to test sequence learning (Chapter 3) is trained with the regulariser instead of adding noise to match the set-up in this section. After each 10 training steps, GMM with different number of mixture components K is fitted to $\mathbf{r}$ using Algorithm 1. As predicted by the analysis, when the number of mixture components K increase, the GMM achieved better likelihood and more closely tracked the regularisation loss.

Conclusion

This chapter reviews artificial neural networks from the perspective of probabilistic modelling. A particular focus is autoencoders and their connections with generative models. They provides statistical tools to analyse neural representations resembling those observed in the brain. A number of classical statistical models are reviewed, in connection with neural networks, to provide intuition into how the representations are affected by their underlying statistical assumptions. Appendix A further extends results from this chapter to recurrent neural networks.

Towards the end of this chapter, I presented a novel contribution of this thesis, showing that the contractive regulariser $\mathcal{R}_M$ (equation 2.42, Rifai et al., 2011) can approximate a constrained GMM prior in training neural generative models. This prior provides more flexibility compared with simpler priors, such as uniform and single Gaussian priors; this regulariser allows efficient training without explicitly fitting the GMM model. A potential problem of this implicit prior is that it is impossible to sample from it, as the exact parametric form (including parameters such as K, $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$) is not known. Despite this, it provides the necessary regularisation to train the inference model, which produces the representation (Dayan et al., 1995; Hinton et al., 1995). In the next chapter, I will show that this regulariser can be minimised implicitly when training with additive noise (section 3.3).

Chapter 3

Recurrent neural networks

This chapter discusses recurrent neural networks (RNNs), the main computational model used in this thesis for sequence learning. After an introduction of the basics, I will focus on practical issues including regularisation techniques and denoise training. Towards the end, I will come back to the topic generative models as introduced in the previous chapter; I will show that the contractive regulariser (section 2.6.3) connects denoising training with the generative model view of neural networks including RNNs.

3.1 RNNs: a brief introduction

A recurrent neural network (RNN) is a type of neural network characterised by its recurrent connections. A layer of nonlinear neurons, sitting between input and output, is usually required for expressive computation and rich representations (Minsky and Papert, 1969). Figure 3.1 illustrates an RNN with one such hidden layer; it is perhaps the simplest possible structure to support representations with rich temporal dynamics. Given the input $\mathbf{x}_t$ at each time step t, this RNN can be formally described by:

$$\mathbf{y}_t = \mathbf{W}_{\text{in}} \cdot \mathbf{x}_t + \mathbf{W}_{\text{rec}} \cdot \mathbf{r}_{t-1} \tag{3.1}$$

$$\mathbf{r}_t = \mathbf{s}(\mathbf{y}_t) \tag{3.2}$$

$$\mathbf{z}_t = \mathbf{W}_{\text{out}} \cdot \mathbf{r}_t \tag{3.3}$$

where $\mathbf{W}_{\text{in}}$, $\mathbf{W}_{\text{out}}$ and $\mathbf{W}_{\text{rec}}$ are input, output and recurrent weight matrices. $\mathbf{x}_t$, $\mathbf{y}_t$ and $\mathbf{r}_t$ are vectors representing the input, internal states and activities of the RNN's hidden neurons. $\mathbf{z}_t$ is the linear read-out from the RNN. $\mathbf{s}(\cdot)$ is a nonlinear activation function applied element-wise. Equation 3.1 to 3.3 can be interpreted as the first order Euler integration of a continuous dynamical system (section 4.4.1).

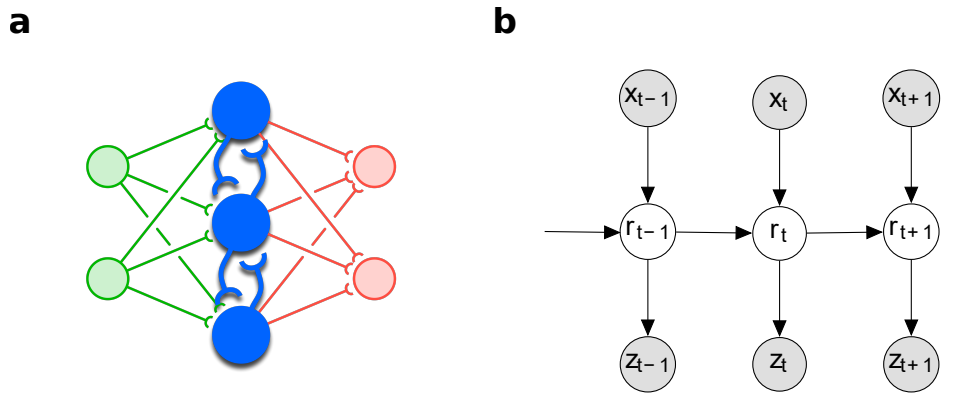


Fig. 3.1 A recurrent neural network (RNN, **a**) and the illustration of an RNN unrolled in time (**b**). (**a**) Green, blue and red show the input ($\mathbf{x}_t$), hidden layer ($\mathbf{y}_t$ and $\mathbf{r}_t$) and output layer ($\mathbf{z}_t$) respectively. Neurons in the hidden layer are interconnected by recurrent connections. (**b**) At each time step t, the activity of the hidden neurons (neural representation) $\mathbf{r}_t$ is a function of both the current input $\mathbf{x}_t$ and the previous activity $\mathbf{r}_{t-1}$. As a result, the activity $\mathbf{r}_t$ may preserve information about the history of inputs, which can represent the context. However, since information from input generally decays exponentially through time, only information relevant to the task tends to be preserved through training.

Although the exact form of the nonlinear activation function does not affect the following derivation, rectified linear units (ReLU) with the threshold activation function

$$\mathbf{s}(\mathbf{y}) = \begin{cases} \mathbf{y} & \text{if } \mathbf{y} \geqslant 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

are used throughout this thesis. Compared with the similarly widely used "tanh" nonlinearity, ReLU better approximates the firing rate function of biological neurons because of its non-negative output and non-saturation property (Priebe and Ferster, 2005). Moreover, neural networks using ReLUs have better performance in various tasks, because they suffer less from the problem of diminishing gradients (Glorot et al., 2011; Nair and Hinton, 2010). The effects of using ReLUs with denoising training are discussed in section 3.3.

Other more sophisticated forms of RNNs include the long-short term memory (LSTM, Hochreiter and Schmidhuber, 1997) and gated recurrent units (GRUs, Chung et al., 2014). These models have additional variables and gating mechanisms dedicated to represent and maintain the states of neurons, so that information from previous inputs can be more easily preserved compared with the simple RNN. As a result, these structurally more complex models are in fact much easier to train, especially for tasks requiring long temporal dependencies. However, the extent to which such an architecture is necessary is still unclear (Le et al.,

2015). Moreover, the extra structures introduced in these models complicate analysis, as the responsibilities of the additional variables are further blurred by interactions of various gates. In contrast, the vanilla RNN only have one set of hidden variables, which are responsible for both retaining temporal context and decoding outputs. Therefore, this thesis mainly focuses on the RNN as depicted in Figure 3.1 for its simplicity[1].

The ability of RNNs to learn complex temporal structure from data has first been systematically discussed in semantic learning by Elman (1990), who elegantly reasoned that contextual information embedded in a sequence can be learned from the sequence itself (as the title of his seminar paper suggested: *Finding Structure in Time*). Since then, RNNs have been exploited for more challenging temporal tasks, such as language and speech processing (Bahdanau et al., 2015; Graves et al., 2013; Maas and Le, 2012). This property of RNNs can be seen by unrolling an RNN through time (Figure 3.1): the activity $\mathbf{r}_t$, therefore the output $\mathbf{z}_t$, at each time step depends on inputs from all previous steps through activities $\mathbf{r}_{t-1}$, $\mathbf{r}_{t-2}$ ... $\mathbf{r}_1$. Therefore, in principle, the RNN is capable of capturing arbitrarily long ranges of temporal dependencies in data. More specifically, this capability places dual responsibility on the representation $\mathbf{r}_t$: in addition to providing the code from which the output is decoded, it functions as a memory of previous inputs. It is therefore interesting to analyse how such representation supports learning temporal structure (which we approach in later chapters via the mixed selectivity they exhibit).

In addition to their favourable computational properties, RNNs have been used as models of the cortex, due to the ubiquitous recurrent connections presented across various cortical regions, including the prefrontal cortex (PFC) and the hippocampus. For example, the recurrent neural networks in the PFC have been found to support crucial computation in context-dependent decision making (Mante et al., 2013; Sussillo et al., 2015). The dense recurrent connections in the hippocampus, in particular the CA3 region, may be crucial in memory and navigation, through their potential role in pattern separation, pattern completion, and temporal association (Bush et al., 2010; Hasselmo and Eichenbaum, 2005; Káli and Dayan, 2000; Lengyel and Dayan, 2007; Marr, 1971; Monaco and Levy, 2003). In particular, the role of RNNs in associative memory has been studied since the beginning of computational neuroscience (Festa et al., 2014; Hopfield, 1982; Lengyel and Dayan, 2005; Savin et al., 2013). More generally, recurrent neural networks may support rich and balanced neural dynamics that underline various functions (Ostojic, 2014; van Vreeswijk and Sompolinsky, 1996), such as sparse coding (Shriki et al., 2000), optimal control (Hennequin et al., 2014b) and probabilistic sampling (Hennequin et al., 2014a).

[1] Our basic results have been reproduced using LSTM, although details are not reported here.

In this thesis, I explore the role of RNNs in sequence learning. Compared with the other temporal domain tasks as reviewed above, the sequence learning task here requires learning relatively long temporal dependency (section 1.3.1). While this task can be easily solve with modern training techniques, I will analysed the neural representation from these RNNs and its implications for similar representations in the brain. Given the rich history of connecting RNNs with neural processes, I aim to uncover potential computational principles in neural systems.

3.2 Training RNNs

RNNs are powerful parametric models that are capable of modelling complicated temporal structures — however, their computational power relies on properly adjusting a possibly very large number of parameters. Since there is no closed-form solution for such complex nonlinear systems, iterative and gradient-based training procedures are usually used to adjust RNNs' parameters. Gradient descent in practice is challenging, due to the presence of long plateaus in the surface of the error function (Dauphin et al., 2014) and easily vanishing or exploding gradients (Pascanu et al., 2012). This section briefly introduces an efficient training procedure and essential additional techniques required for successful training. A probabilistic interpretation of this procedure for feed-forward neural networks has been presented in chapter 2, which is extended for RNNs in Appendix A.

3.2.1 Gradient descent with backpropagation

The purpose of training is to minimise a cost $\mathcal{C}$ as a function of the RNN's output $\mathbf{z}_t$ and the corresponding target $\mathbf{z}_t^*$ (for simplicity, we ignore the expectation over different training sequences):

$$\mathcal{C} = \frac{1}{T}\sum_{t=1}^{T} \mathcal{E}\left(\mathbf{z}_t, \mathbf{z}_t^*\right) \tag{3.5}$$

where $\mathcal{E}$ is an error function measuring the discrepancy between the model's output $\mathbf{z}_t$ and its target $\mathbf{z}_t^*$. The squared error

$$\mathcal{E}\left(\mathbf{z}_t, \mathbf{z}_t^*\right) = \frac{1}{2}\left\|\mathbf{z}_t - \mathbf{z}_t^*\right\|^2 \tag{3.6}$$

is often used for continuous outputs from a linear output layer, as in the simulations presented in this thesis. The derivative of the error function with respect to a parameter w_j has the form:

$$\frac{\partial \mathcal{E}\left(\mathbf{z}_t, \mathbf{z}_t^*\right)}{\partial w_j} = \left(\frac{\partial \mathbf{z}_t}{\partial w_j}\right)^{\mathsf{T}} \cdot \left(\mathbf{z}_t - \mathbf{z}_t^*\right) \tag{3.7}$$

The difference $(\mathbf{z}_t - \mathbf{z}_t^*)$ can be computed locally at the output layer. The derivative $\frac{\partial \mathbf{z}_t}{\partial w_j}$ can be computed by applying the chain-rule, which is implemented in neural networks as the backpropagation algorithm (Rumelhart et al., 1986; Werbos, 1990). The simplest implementation of gradient descent incrementally updates parameters $\mathbf{w}$ in each iteration by

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \frac{\partial \mathcal{C}}{\partial \mathbf{w}} \tag{3.8}$$

where α is the step size of updating. In theory, under some mild condition of diminishing step sizes through iterations, $\mathcal{C}$ will converge to its local minimum upon repeatedly applying the update rule in equation 3.8 (Robbins and Monro, 1951).

3.2.2 Techniques and tricks in training

The naïve implementation of equation 3.8 works poorly for problems at any practical scale, in part because the highly non-convex error surface is overloaded with saddle points and long plateaus (Dauphin et al., 2014). Moreover, the presentation of long-term dependencies makes RNNs prone to the problem of vanishing or exploding gradients (Pascanu et al., 2012). As a result, additional care must be taken in training RNNs. This section reviews several techniques and tricks that can be used to improve the training of RNNs.

Hessian-Free optimisation (HF) (Martens, 2010; Martens and Sutskever, 2011) is a second-order method that computes a positive semi-definite approximation of the Hessian via modified backpropagation through time (Pearlmutter, 1994; Schraudolph, 2002). HF solves the problem of vanishing and exploding gradient by normalising the gradients using second-order information, as in other quasi-Newton methods. In addition, it effectively avoids directly handling the tricky saddle points in error surfaces by forcing a positive semi-definite approximation of the Hessian, the Gauss-Newton matrix. However, HF is difficult to implement, and it often turns out to be less efficient than simple stochastic gradient descent (SGD) combined with careful initialisation and scheduling of training (Sutskever et al., 2013), which we shall introduce next.

Proper initialisation of parameters is the prerequisite of successfully training a neural network. Intuitively, appropriate initialisation brings the parameters close to good local optima, while avoiding regions of the error surface with malicious geometry, such as very long plateaus and narrow troughs. For RNNs in particular, good initialisation sets up the network in a favourable dynamical regime. A widely used method involves normalising the magnitude of input weights into each neuron to 1 (Sutskever et al., 2013). For square recurrent connection matrices with Gaussian random weights, this results in eigenvalues roughly uniformly distributed within the unit circle, which provides stable and rich dynamics

(Sompolinsky et al., 1988). Another technique I found particularly helpful is to initialise the diagonal of the recurrent weight matrix as the identity matrix multiplied by a number less than but close to 1, so that initially the hidden neurons are close to perfect integrators (Le et al., 2015). This technique alone reduced the number of iterations required by an order of magnitude in my experiments.

Given a properly initialised RNN, as discussed above, SGD worked well with a combination of large momentum, gradient clipping (Bengio, 2009) and denoising training (see below). SGD generally achieved the same performance as RNNs trained using HF and converged faster. Therefore, all the results reported in this thesis are from RNNs trained with SGD.

3.3 Regularisation and Denoising training

3.3.1 Denoising training: an overview

After initialisation, regularisation is often essential for training models with large numbers of parameters. This practice is usually interpreted as avoid over-fitting and improve generalisation. For over-parametrised and expressive models like neural networks, regularisation is usually necessary. It seems obvious that regularisation is crucial in the sequence learning tasks examined here, where the dataset size is relatively small (section 4.1). However, this issue is in fact more general since neural networks are typically over-parametrisation in practice.

For example, in our setting, the ratio of parameters vs. data size is $100 \times 100\,(\text{recurrent weights}) \times 0.1\,(\text{connectivity}) + 2 \times 2 \times 100\,(\text{input / output weights}) : 30\,(\text{steps}) \times 2\,(\text{trials}) \times 2\,(\text{dimensions}) = 1400 : 120 < 12 : 1$. Settings where this ratio is much higher (e.g., 1000 : 1) has been considered in recent papers on over-parametrised neural networks Allen-Zhu et al. (2018); Arora et al. (2019); Li and Liang (2018); Zhang et al. (2016). Most of these works focus on the *implicit* regularisation from optimising such models using stochastic gradient descent (SGD).

One technique I found particularly useful is implicit regularisation via denoising training (Bishop, 1995; Vincent et al., 2008b, 2010), which I later distil into the same contractive regulariser as equation 2.42 (section 3.3). Denoising training adds zero mean noise ξ_t to the input x_t

$$\tilde{\mathbf{x}}_t \leftarrow \mathbf{x}_t + \boldsymbol{\xi}_t \tag{3.9}$$

and then using the noisy copies $\tilde{\mathbf{x}}_t$ in training. This simple technique significantly improves the robustness of RNNs in my experiments. To avoid bias from few noise samples, many noise samples throughout training are necessary. Note that adding noise to targets $\mathbf{z}_t^*$ is unnecessary (Bishop, 1995). This can be seen by noticing that, when taking derivatives

of the cost function, the noise added to $\mathbf{z}_t^*$ is averaged out in equation 3.7. In a biological system, noise is ubiquitous (Faisal et al., 2008) — it may come from, for example, imperfect perception of the external world or imperfect communication between internal neural systems. I therefore added Gaussian noise to both input and output in my experiment to more faithfully reflect this fact. Other popular regularisation techniques, such as drop-out (Hinton, 2014a), are not clearly beneficial in my experimental setting, probably due to the relatively small model size.

It is important to understand why denoising training can improve training. Expanding the dataset with noisy copies incurs non-trivial computational cost in both the process of adding noise and the increased variance during training; from a practical perspective, it is worth investigating whether similar effects can be achieved more efficiently without such overheads. For theorists, understanding denoising training may provide insights into the robustness of neural systems, which face noise at almost all levels. Therefore, I propose an closed-form approximation of denoising training. This approximate form can be used as a regulariser in combination with the original loss function, whereby noisy copies of data are no longer required. In experiments, I found nearly identical results as denoising training were achieved with this regulariser. In addition, the analytical form of this regulariser allows us to more directly characterise the effects of de-noising training on neural representations. An interpretation from the perspective of probabilistic generative models is presented in section 2.6.3 and Appendix A.4.

3.3.2 Denoising as a Tikhonov regulariser

For feed-forward neural networks, Bishop (1995) showed that instead of corrupting input data with noise, denoising training can be approximated by adding a Tikhonov regulariser to the original cost function. Following the a similar procedure, I extend the Tikhonov regulariser to the representation $\mathbf{r}$ instead of the final output $\mathbf{z}$, which results in the contractive regulariser (Rifai et al., 2011) that I derived in section 2.6.3 from the perspective of approximating a GMM prior. Regularising the representation has the advantage that the same regulariser can be used for different output activations (e.g., linear, Bernoulli, or softmax functions). In addition, while Bishop (1995) derived the Tikhonov regulariser for feed-forward neural networks, I extend the derivation for RNNs.

Following equation 3.9, to account for the temporal dependencies of an RNN, the RNN's representation given noisy input is denoted as $\tilde{\mathbf{r}}_t\left(\{\mathbf{x}_k+\boldsymbol{\xi}_k\}_{k=1}^t\right)$ or $\tilde{\mathbf{r}}_t$ for short, a function of inputs and additive noise until time t, $\{\mathbf{x}_k\}_{k=1}^t$ and $\{\boldsymbol{\xi}_k\}_{k=1}^t$ respectively. Similarly, $\mathbf{r}_t$ is the shorthand for $\mathbf{r}_t\left(\{\mathbf{x}_k\}_{k=1}^t\right)$ without additive noise. The i'th element of $\tilde{\mathbf{r}}_t$ has the second

order Taylor expansion at $\mathbf{r}_t^i$,

$$\tilde{\mathbf{r}}_t^i = \mathbf{r}_t^i + \sum_{k=1}^{t} \boldsymbol{\xi}_k{}^\intercal \frac{\partial \mathbf{r}_t^i}{\partial \mathbf{x}_k} + \sum_{k=1}^{t} \frac{1}{2} \boldsymbol{\xi}_k{}^\intercal \frac{\partial^2 \mathbf{r}_t^i}{\partial \mathbf{x}_k^2} \boldsymbol{\xi}_k + \sum_{k=1}^{t} \mathcal{O}\left(\boldsymbol{\xi}_k^3\right) \tag{3.10}$$

where $\frac{\partial \mathbf{r}_t^i}{\partial \mathbf{x}_k}$ is the i'th column of the Jacobian matrix $\frac{\partial \mathbf{r}_t}{\partial \mathbf{x}_k}$, and $\frac{\partial^2 \mathbf{r}_t^i}{\partial \mathbf{x}_k^2}$ is the Hessian for $\mathbf{r}_t^i$.

The objective of denoising training is bringing $\tilde{\mathbf{r}}_t$ close to the noiseless representation $\mathbf{r}_t$, so that the representation becomes invariant to input noise. This objective can be achieved by minimising the following denoising cost $\mathcal{D}$, which quantifies the discrepancy between $\tilde{\mathbf{r}}_t$ and $\mathbf{r}_t$ as a result of the added noise:

$$\begin{aligned}
\mathcal{D} &= \frac{1}{2T} \sum_{t=1}^{T} \|\tilde{\mathbf{r}}_t - \mathbf{r}_t\|^2 \\
&= \frac{1}{2T} \sum_{t=1}^{T} \Bigg\{ (\mathbf{r}_t - \mathbf{r}_t)^2 + \sum_i \left(\sum_{k=1}^{t} \boldsymbol{\xi}_k{}^\intercal \frac{\partial \mathbf{r}_t^i}{\partial \mathbf{x}_k} + \sum_{k=1}^{t} \frac{1}{2} \boldsymbol{\xi}_k{}^\intercal \frac{\partial^2 \mathbf{r}_t^i}{\partial \mathbf{x}_k^2} \boldsymbol{\xi}_k \right)^2 \\
&\qquad + 2 \sum_{i=1}^{K} \left(\sum_{k=1}^{t} \boldsymbol{\xi}_k{}^\intercal \frac{\partial \mathbf{r}_t^i}{\partial \mathbf{x}_k} + \sum_{k=1}^{t} \frac{1}{2} \boldsymbol{\xi}_k{}^\intercal \frac{\partial^2 \mathbf{r}_t^i}{\partial \mathbf{x}_k^2} \boldsymbol{\xi}_k \right) \cdot \left(\mathbf{r}_t^i - \mathbf{r}_t{}^i \right) \Bigg\} \\
&= \frac{1}{2T} \sum_{t=1}^{T} \sum_{i=1}^{K} \left(\sum_{k=1}^{t} \boldsymbol{\xi}_k{}^\intercal \frac{\partial \mathbf{r}_t^i}{\partial \mathbf{x}_k} + \sum_{k=1}^{t} \frac{1}{2} \boldsymbol{\xi}_k{}^\intercal \frac{\partial^2 \mathbf{r}_t^i}{\partial \mathbf{x}_k^2} \boldsymbol{\xi}_k \right)^2
\end{aligned} \tag{3.11}$$

These terms can be further simplified by considering the expectation over the distribution of the noise ξ_k. Utilising the two moments of the added noise,

$$\langle \boldsymbol{\xi}_k \rangle_{p(\boldsymbol{\xi}_k)} = 0 \tag{3.12}$$

$$\left\langle \boldsymbol{\xi}_k \boldsymbol{\xi}_k{}^\intercal \right\rangle_{p(\boldsymbol{\xi}_k)} = \eta^2 I \tag{3.13}$$

The expectation then becomes

$$\left\langle \sum_{i=1}^{K} \left(\sum_{k=1}^{t} \boldsymbol{\xi}_k{}^\intercal \frac{\partial r_t^i}{\partial \mathbf{x}_k} + \sum_{k=1}^{t} \frac{1}{2} \boldsymbol{\xi}_k{}^\intercal \frac{\partial^2 r_t^i}{\partial \mathbf{x}_k^2} \boldsymbol{\xi}_k \right)^2 \right\rangle_{p(\{\boldsymbol{\xi}_k\}_{k=1}^t)} = \eta^2 \sum_{k=1}^{t} \left\| \frac{\partial \mathbf{r}_t}{\partial \mathbf{x}_k} \right\|_F^2 \tag{3.14}$$

where σ_ξ^2 is the variance of the additive noise.

Therefore, denoising can be approximated by adding the Tikhonov regulariser $\mathcal{T}$ to the original cost $\mathcal{C}$:

$$\mathcal{T} = \sum_{t=1}^{T} \sum_{k=1}^{t} \left\| \frac{\partial \mathbf{r}_t}{\partial \mathbf{x}_k} \right\|_F^2 \tag{3.15}$$

The noise variance σ_ξ^2 can be merged with the weighting of $\mathcal{D}$ which will be discussed later. This approximation is accurate when the magnitude of noise is small, so the higher order terms in the Taylor expansion (equation 3.10) can be ignored (Bishop, 1995).

At this point, we recovered the contractive regulariser derived in section 2.6.3 (equation 2.42) by ignoring the time index t for autoencoders, and substituting $\mathbf{r} = \mathbf{g}_\phi(\mathbf{x})$. Therefore, I have derived the regulariser from the two different perspectives:

1. Approximating a constrained GMM prior in generative models;

2. Approximating denoising training.

This correspondence will be discussed further. In this section, I follow Bishop (1995) to call $\mathcal{T}$ (equation 3.15) a "Tikhonov regulariser" to emphasise its derivation. In the following text, I shall return to the name "contractive regulariser" which more closely matches the mathematical form and better characterises its computation.

Applying the chain rule, we obtain the output at time t depending on the input at time k through

$$\frac{\partial \mathbf{r}_t}{\partial \mathbf{x}_k} = \boldsymbol{D}_t \circ \left(\prod_{s=k}^{t-1} \boldsymbol{W}_s \right) \cdot \mathbf{W}_{\text{in}} \tag{3.16}$$

where $\boldsymbol{D}_t$ is an $M \times M$ matrix such that all its columns are $\mathbf{s}'(\mathbf{y}_t)$, the derivatives of the activation functions of all hidden neurons at time t, and $\circ$ represents the Hadamard (element-wise) product. $\boldsymbol{W}_s = \mathbf{W}_{\text{rec}} \circ \boldsymbol{D}_s^\intercal$ represents the *effective* projection through the recurrent connections.

From equation 3.16, the penalty from $\mathcal{T}$ would be small if $\boldsymbol{D}_t$ were sparse (with many zero elements). For the ReLUs we use (equation 3.4), sparse $\boldsymbol{D}_t$ corresponds to sparse activity, as the zero activation and zero derivative of a ReLU coincide. A close examination of $\mathcal{T}$ gives us intuition into how this works. From equation 3.16, minimising the magnitude of $\mathcal{T}$ encourages large elements of $\left(\prod_{s=k}^{t-1} \mathbf{W}_s\right) \cdot \mathbf{W}_{\text{in}}$ to align with small (zero for ReLU) elements of $\mathbf{D}_t$. This means an arbitrary and unstructured input, which is likely to be noise, should diminish after passing through the input weights $\mathbf{W}_{\text{in}}$ and the effective recurrent weights $\mathbf{W}_s$ for $t-k$ times. Since both $\mathbf{D}_t$ and $\mathbf{W}_s$ may change in each step, this denoising procedure is dynamical. The flexibility of this dynamical process allows the RNN to find configurations that balance well the trade-off between the original objective (equation 4.1) and the regulariser.

3.3.3 An approximate Tikhonov regulariser for RNNs

Unlike the contractive regulariser for feed-forward neural networks (when $T = 1$), equation 3.16 and its derivatives are difficult to compute — due to the cumulative temporal dependencies, evaluating $\mathcal{T}$ requires $\mathcal{O}(T^2)$ computation. Therefore, $\mathcal{T}$ is not ideal to be used directly as a regulariser for training RNNs, and we need to seek a regulariser that is cheaper to compute but yields similar performance.

Based on the dynamical perspective developed above, I approximate the effect of equation 3.15 by removing the output matrix $\mathbf{W}_{\text{out}}$, the state-dependent derivatives $\boldsymbol{D}_s$, and higher-order dependencies in the product $\prod_{s=k}^{t-1} \boldsymbol{W}_s$. As discussed previously, removing the higher-order terms of $\boldsymbol{W}_s$ and $\boldsymbol{D}_s$ essentially encourage the RNN to filter out noise from inputs in no more than than $t-k$ steps. Although this sacrifices part of the flexibility, it significantly simplifies the computation from $\mathcal{O}(T^2)$ to $\mathcal{O}\left((t-k)T\right)$. Importantly, with $\boldsymbol{D}_t$ remains, the regulariser can still adapt to the dynamics. In my experiment, I used the second-order approximated Tikhonov regulariser by fixing $t-k=1$:

$$\mathcal{T}_2 = \sum_{t=1}^{T} \|\boldsymbol{B}\|_F^2 \tag{3.17}$$

$$B = \eta_1 \boldsymbol{D}_t \mathbf{W}_{\text{in}} + \eta_2 \boldsymbol{D}_t \mathbf{W}_{\text{rec}} \mathbf{W}_{\text{in}} \tag{3.18}$$

where two separate coefficients η_1 and η_2 are introduced for the first and second order terms respectively for extra flexibility. Note that at least two terms are required to include the recurrent connections, i.e. the temporal dynamics of the RNN. This second order regulariser encourages the noise to be filtered out in just *one* step.

Using grid search, I found the optimal values for these hyper-parameters to be $\eta_1 = 0.02$ and $\eta_2 = 0.01$, which are close to the initial guess based on the analytical forms — the larger value for η_1 possibly compensated for the higher order terms that were lost in the approximation. It is straightforward to include higher order terms, corresponding to increasing $t-k$ and allowing more steps before the input noise vanishes, which may be helpful in other tasks. Note that the derivation of the denoising regulariser did not assume any specific task, and so we expect it to generalise to a broad range of tasks of practical interests. This is a direction for future investigation.

Further, as the main purpose of deriving the regulariser $\mathcal{T}$ is to help us understand the denoising training procedure, I still used denoising training for all the results in the main thesis to simulate the noisy environment animals face [2] (Faisal et al., 2008). To verify its

[2] It is also the case that the efficiency gained from using the regulariser is negligible in the simple task we use.

effect in training, Appendix C presents main results from the thesis reproduced using the contractive regulariser (equation 3.18) instead of denoising training.

3.3.4 Probabilistic interpretation

A probabilistic view of the denoising process arrives by noticing that the regulariser $\mathcal{R}_M$ (equation 2.42) that approximates training with GMM prior (section 2.6.3) is identical to the Tikhonov regularizer $\mathcal{T}$ derived in this chapter for denoising (equation 3.15). This isometry suggests robustness to input noise can be interpreted as having a GMM prior in generative models. Intuitively, this is because GMM is insensitive to small perturbations that do not change the cluster (mixture component) assignment.

In addition, the same regulariser derived from different perspectives establishes a novel connection between denoising training and properly training generative models. It indicates that simply training with noise would provide the constraint on the approximating posterior, which is involved in computing the variational lower-bound in maximising the model's likelihood (section 2.6).

3.3.5 Related work

$\mathcal{R}_M$ (equation 2.42) has the exact form of the contractive regulariser (Rifai et al., 2011), which has been proposed from the motivation of explicitly controlling the variance of features. With the extra temporal dependent structure, $\mathcal{T}_2$ can be seen as a generalisation of the contractive regulariser for RNNs, which encourages input invariant representations through time in the hidden layer. Rifai et al. (2011) show that, with this regulariser, the resulting contractive autoencoders (CAE) are able to extract meaningful features that yield good performance in classification for a broad range of data. They explained the performance in terms of *local space contraction*, which is consistent with our probabilistic interpretation of Gaussian mixtures priors, since each of the mixture components can be seen as a local contraction around its (implicit) mean.

More recently, Sussillo et al. (2015) independently proposed a regulariser that is essential for reproducing experimentally observed naturalistic solutions and biologically plausible tunings in RNN models for motor control tasks. This regulariser is a special case of $\mathcal{T}_2$ by setting $\eta_1 = 0$, thus retaining only the second-order term.

Previous work have explored the connections between autoencoders and generative models. For example, Vincent (2011) discovered the connection between training denoising autoencoders and matching the score of energy based (undirected) models (Hyvärinen, 2006). More recently, Bengio et al. (2013) justified the generative model implicitly learned from

training a denoising autoencoder using arbitrarily corrupted data, and extended previous Markov Chain Monte-Carlo method for contrastive autoencoders (Rifai et al., 2012), proposing a general method to sample from these generative models. In this chapter, I further draw the connection between denoising autoencoders and directed graphical models, so the developed neural representation can be more easily characterised as a latent variable with statistical properties inherited from the prior.

Even though training with noise-corrupted data has been well-known to promote the performance of autoencoders as well as other neural network models, we explained the effects of noise in the context of variational inference in minimising the KL-divergence $\mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{r}|\mathbf{x})\|p_\theta(r)\right]$ in the lower bound $\mathcal{L}$. Compared with recently proposed variational autoencoders using stochastic backpropagation (Rezende et al., 2014a) or the equivalent re-parametrisation trick (Kingma and Welling, 2014), we show that a generative model can be trained without introducing any new model component (e.g., re-parametrised distributions, explicit priors). Given the ubiquity of noise in biological system, this simple method suggests a potential direction towards the statistical modelling employed by the brain. The next chapter shows further evidence supporting this hypothesis — a causal relationship between denoising /contractive autoencoders and the mixed-selectivity widely observed across different brain regions.

Conclusion

This chapter introduces recurrent neural networks, which I will use from the next chapter for the sequence learning task. From the practical problem of training neural networks, I dived into more theoretical aspects of commonly used tricks, such as regularisation denoising training. Somewhat surprisingly, by analytically approximating denoising training, I derived the same regulariser as the contrastive regulariser derived in the previous chapter, where it was used to approximate training generative models with a GMM prior. This generative model perspective justifies that denoising training is a principled way of capturing the statistical structure of data, which shall manifest later in this thesis in the form of mixed selectivity (Chapter 5). The technique used in this chapter mainly follows Bishop (1995). I extended his original Tikhonov regulariser on representations and for RNNs. The resulting regulariser generalises a few regularisers that previously have been shown to produce biologically plausible representations.

Chapter 4

RNNs performing sequence disambiguation

This chapter presents the RNN trained on the T-maze sequence learning task. As introduced in section 1.3.1, solving this task involves sequence disambiguation, and the vast amount of available neural data provides references for analysing representations developed in my simulation. I demonstrate that RNNs trained for sequence prediction exhibit place cell-like firing patterns. It was the first time such biologically plausible patterns were observed from RNNs trained only for performance (first reported in my COSYNE talk in 2015, in Salt Lake City). I further analyse statistical properties of such firing patterns, quantifying its similarity with neural representations observed in the brain, and show that such patterns support line-attractor dynamics that solve the task.

4.1 Training

The T-maze alternation task (section 1.3.1) can be formulated as a sequence prediction problem, which requires sequence disambiguation. Given the ability of RNNs to represent input history as context (section 3.1) and their structural similarity to cortical circuits, here an RNN (Figure 3.1) is employed solve this problem. At each time step, the position of a simulated rat, a two-dimensional coordinate, was fed as input into the RNN; the RNN then output a coordinate as the predicted position at the next time step.

Here I use the coordinate instead of the pixels from first person's view (e.g., Banino et al., 2018), and train the model for one-step prediction (as a form of maximum likelihood learning) instead of reinforcement learning (RL, Sutton and Barto, 1998) for the following reasons:

1. This thesis focuses on computational principles of neural representation, so I choose this simple setting to avoid unnecessary or confounding influence from other components. We can assume these coordinates come from other areas of the brain (e.g., visual cortex) that are less relevant to the main analysis here.

2. This minimal model is much easier to train compared with full-scale models that take raw-pixel inputs or those perform RL. Processing pixel inputs would require training powerful neural encoder and decoder, and RL generally has high variance. The model presented in this thesis can be trained locally on a CPU within a minute, allowing investigating the computational principles in sequence learning with much lower computational cost.

Importantly, the using coordinates as inputs still preserves the main computational problem this thesis investigates — the ambiguity from parts of the sequences. Therefore, although it is useful to study how the entire system interact with environment in an end-to-end fashion, here we can study the core computational principles with a more manageable computational budget.

More specifically, the input positions as coordinates were sampled uniformly along constant speed trajectories in left-turn or right-turn trials (figure 1.2). To simulate the effect of non-perfect perception of the spatial location, a significant amount of noise (with a standard deviation of 0.1, equivalent to 5 cm in the scale here, as shown in Figure 4.1) was added to the sampled position, which resulted in denoising training (section 3.3).

Besides added noise, the major challenge of this prediction problem was at the decision point, where prediction required integrating contextual information before the central arm — the input positions on the central arm were identically distributed for both types of trials, so they could not help disambiguation. In other words, external input alone was not sufficient to successfully disambiguate the two types of trials; the context, which exhibited temporal dependencies across about 10 time steps — the length of the central arm — needed to be represented by the RNN internally.

Formally, this task can be described using the notation of equations 3.1 - 3.3 and 3.5. At step t, the target $\mathbf{z}_t^*$ is the next (noisy) position $\tilde{\mathbf{x}}_{t+1}$. For clarity, I write the output $\tilde{\mathbf{z}}_t$ in equation 3.3 as $\hat{\mathbf{x}}_{t+1}$, to explicitly express it as a prediction of $\tilde{\mathbf{x}}_{t+1}$. The cost function (equation 3.5) then becomes the squared prediction error

$$\mathcal{C} = \frac{1}{2T} \sum_{t=1}^{T} \|\hat{\mathbf{x}}_{t+1} - \tilde{\mathbf{x}}_{t+1}\|^2 \tag{4.1}$$

From equation 3.1 - 3.3, $\hat{\mathbf{x}}_{t+1}$ is a function of the current position $\mathbf{x}_t$ and the hidden neurons' activity $\mathbf{r}_t$. Therefore, the context has to be represented by the activity $\mathbf{r}_t$. Unlike previous approaches that rely on heuristically designed context-sensitive neurons (Hasselmo and Eichenbaum, 2005; Howard and Kahana, 2002; Levy, 1996; Sohal and Hasselmo, 1998; Wallenstein and Hasselmo, 1997; Zilli and Hasselmo, 2008), I do not directly specify *how* the population activities $\mathbf{r}_t$ should encode contextual information. Instead, the representations $\mathbf{r}_t$ are optimised for task performance via minimising this squared prediction error.

Adapting the terminology of behavioural experiments (Wood et al., 2000), we call a sequence of noisy samples along the trajectory $\{\tilde{\mathbf{x}}_t\}_{t=1}^{T}$ a *trial*. Within the same type (left-turn or right-turn), trials differed only by different samples of the additive noise $\boldsymbol{\xi}_t$. The underlying noiseless positions $\mathbf{x}_t$, which were evenly sampled along the trajectories, were the same (Figure 4.1, **c**). Consequently, between different types of trials, the underlying positions along the central arm were identical. I train the input, output and recurrent connection weights to minimise the next-step prediction error. The exact parameters and other details of training are summarised in Appendix C.

4.2 Performance

Although the RNN was only trained for on one-step prediction (estimating the next location $\hat{\mathbf{x}}_{t+1}$ at each step t), I also tested it on a more challenging recall task. While (the noisy version of) the current position was fed into the RNN at each step for prediction, in recall, such external input was only supplied at the initial step; the outputs of the RNN were then *fed-back* as inputs in all subsequent steps, so that the RNN needed to recall the entire sequences given only the cue from initial step. This generalisation is ecologically relevant: one-step prediction is easy to learn, but prediction over multiple steps — to the extreme of recalling the entire experience — is often required in planning (Kaelbling et al., 1998; Pfeiffer and Foster, 2013; van Seijen and Sutton, 2015). However, this generalisation brings new challenges: it is reasonable to anticipate that prediction errors might accumulate in the recall process, which is essentially a chain of consecutive one-step predictions.

After training, the performance of an RNN in prediction and recall tasks is illustrated in Figure 4.1. During prediction (Figure 4.1, a), the variance of outputs (lines) along the maze was significantly smaller than the variance of the noisy inputs (dots), which shows that the RNN achieved both prediction and denoising. Figure 4.2 (a) shows that outputs had larger errors at the beginning (near the dots), which then quickly converged close to the true positions after only a few steps. Figure 4.2 (a) also shows that the step right before entering the central arm and the last step on the central arm had larger errors — since both of

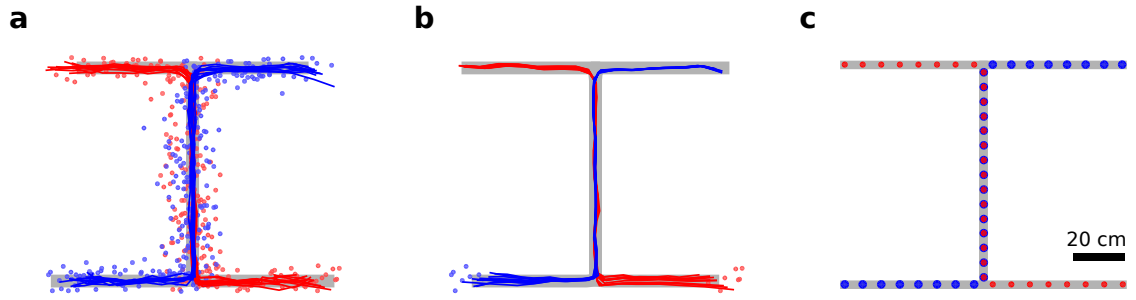


Fig. 4.1 Illustrative examples of a trained RNN performing prediction (**a**) and recall (**b**). Dots show the noisy coordinates of positions as inputs into the RNN; lines connect output from the RNN. Blue and red indicate left-turn and right-turn trials respectively. While external inputs were supplied all the way along the trajectories during prediction, only initial cues were given to the RNN for recall. Ten trials for each type are illustrated. The actual locations of the simulated rat traversed are plotted as dots in **c** (see also the task illustrated in Figure 1.2).

these steps were immediately before the turns, prediction errors from these steps were more difficult to correct using backpropagated information from the next steps. Nevertheless, the errors dropped after these steps.

To further exemplify performance, Figure 4.2 (b) shows the means and standard deviations of errors computed from 100 trials. As a control, we plotted the error computed from a trivial solution which simply copied inputs as predictions. This quantitative result shows that the RNN performed much better than the control in prediction (while no such trivial solution exists for recall). In the more challenging task of recall (Figure 4.1, b), only an initial (noisy) input (shown as a dot) at the beginning of each trial was supplied as a cue. Somewhat surprisingly, although the RNN was continuously performing one-step prediction by feeding back its output to its input at each step, the prediction errors did not accumulate along the trajectories. In fact, the errors during recall were slightly lower than in prediction. We hypothesize that this is because the outputs that fed-back into the RNN were less noisy than the inputs during prediction. In section 4.4, we will explain this result from the perspective of dynamical systems. A probabilistic explanation based on generative models is presented in Chapter 2 and Appendix A.

4.3 Neural representations

The task performance we described in the last section concerns only the input to output mapping of the RNN. To look "under the hood", we plot the activities of hidden neurons, $\mathbf{r}_t$, during the prediction task in figure 4.3 (right). These activities represented a topographical

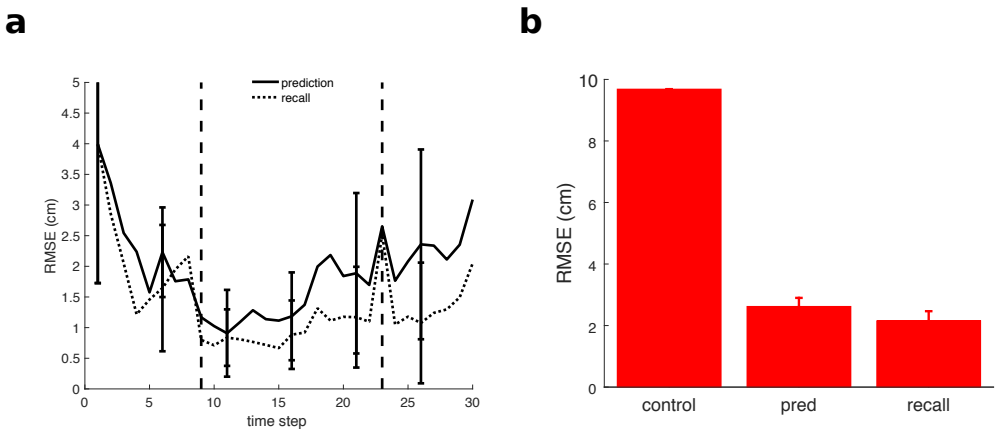


Fig. 4.2 (**a**) The root mean squared error (RMSE) at each step averaged over 100 trials and 10 RNNs for prediction (solid line) and recall (dotted line). During recall, the prediction errors did not accumulate through time. The two vertical dashed lines show the first and last step on the central arm. (**b**) Performance of the RNN in prediction and recall, compared with a control that simply copies input as prediction. Bars show averages from 100 trials each from 10 RNNs, error bars indicate standard errors. Consistent with panel **a**, the errors in recall were slightly lower than during prediction, possibly because the outputs that fed back into the RNN were less noisy than the noisy inputs during prediction.

code of the environment: neurons had developed uni-modal tuning coding for locations that tiled the entire T-maze. Moreover, the activities were sparse, as a neuron only fired on short segments along trajectories. Despite the topographical code being sparse, it was nevertheless distributed — a neuron never fired at only a single location. This representation of space lies in the category of *coarse codes*, as a trade-off between the extremely localist representations, where one neuron only represents a particular location, and the completely distributed representations, where a topographic mapping can not be established (Hinton, 1986). This trade-off has been examined in the light of probabilistic generative models in section 2.5. Such sparsely distributed codes have been argued to play a central role in neural computation, because they exhibit a balance between efficiency and robustness (Hinton, 1986, 2014b; Hinton and Ghahramani, 1997; Rumelhart et al., 1987). Similar sparse and distributed representations exhibiting mixed selectivity have been observed in various cortical areas (Baeg et al., 2003; Fujisawa et al., 2008; Huettel et al., 2002), including the hippocampus, when rats performed the same T-maze alternation task (Wood et al., 2000, see figure 1.3 to 1.6).

Rather than occurring incidentally, the sparsely distributed representations were a result of learning, which can be seen from comparing the firing patterns before and after training (Figure 4.3), as well as through their statistics (Figure 4.4). In contrast to the patterns in Figure 4.3 (**b**), the representation before training (Figure 4.3 **a**) were not sparse and there was

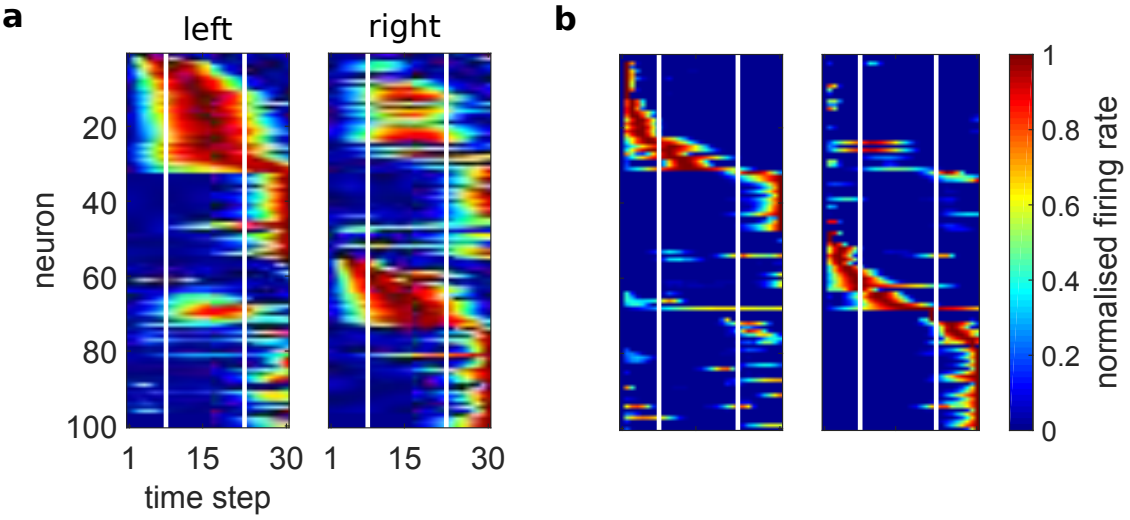


Fig. 4.3 Neural responses before (**a**) and after (**b**) training. Each row represents a neuron and each column represents a location. Two vertical white lines show the beginning and end of the shared central arm. Colour codes the firing rates of the 100 neurons, sorted by locations of maximum activity and normalised between 0 and 1 for each neuron (see also Figure 1.3 to 1.6).

hardly any topographical structure, since the firing of neurons did not reflect the structure of the space. To further confirm this observation, we measured sparseness[1] of the codes by the activity ratio (Olshausen and Field, 2004; Rolls and Tovee, 1995; Vinje and Gallant, 2000), which is defined as

$$a^i = \frac{\left(\frac{1}{T}\sum_{t=1}^{T} r_t^i\right)^2}{\frac{1}{T}\sum_{t=1}^{T}\left(r_t^i\right)^2} \tag{4.2}$$

As shown in figure 4.4 (a), the distribution of activity ratio shifted towards 0 after learning, indicating a significant increase of sparseness. To measure the stereotypy of the tuning curves, I used the measure of *bVar* proposed by Rajan et al. (2016), which quantifies the variability of a population's activities that can be explained by shifting a template. To compute bVar, the template is first obtained by averaging across all neurons their tuning curves $\bar{r}_t$ aligned according to their maximum firing rates:

$$r_t^{\text{template}} = \frac{1}{N}\sum_{i=1}^{N} \bar{r}_{t-t_i}^i \tag{4.3}$$

[1] More precisely, we measured the life time sparseness of individual neurons.

where t_i indicates the location (time) of maximum firing for neuron i averaged across trials. bVar for the population is then computed as the variability explained by shifting the template:

$$\text{bVar} = 1 - \frac{\sum_{i=1}^{N}\sum_{t=1}^{T}\left(\bar{r}_t^i - r_{t-t_i}^{\text{template}}\right)^2}{\sum_{i=1}^{N}\sum_{t=1}^{T}\left(\bar{r}_t^i - \frac{1}{N}\sum_{i=1}^{N}\bar{r}_t^i\right)^2} \tag{4.4}$$

In addition, to further quantify the typicality of *individual* neurons, I adapt the bVar from Rajan et al. (2016) to indicate the variability of individual neurons explained by the template as

$$\text{bVar}^i = 1 - \frac{\sum_{t=1}^{T}\left(\bar{r}_t^i - r_{t-t_i}^{\text{template}}\right)^2}{\sum_{t=1}^{T}\left(\bar{r}_t^i - \frac{1}{N}\sum_{j=1}^{N}\bar{r}_t^j\right)^2} \tag{4.5}$$

By definition, both bVar and bVar^i are normalised quantities between 0 and 1. Figure 4.4 (**b**) shows bVar increased at both individual neuron level and population level (from 0.37 to 0.47), confirming that the firing patterns of neurons became more stereotypical after training.

Since the challenge of the T-maze alternation task is mainly at the decision point at the end of the central arm, I analysed the neural activities more closely on the central arm. Figure 4.5 shows 6 exemplar neurons that fired significantly on the central arm. These neurons resembled the *splitter cells* reported by Wood et al. (2000) (Figure 1.3), as they fired differently on different types of trials, although the RNN received identical inputs at locations along the central arm. Therefore, the trial-type dependent firing of these splitter cells could be used to provide contextual information, disambiguating the left-turn or right-turn trials at the decision point. We thus conclude that the sparse and distributed neural representations developed during learning the one-step prediction task support sequence disambiguation, by encoding context as well as the location of the animal via mixed selectivity.

Why did this biologically plausible type of representations emerge in our model? A full answer will only be provided towards the end of this thesis. Nevertheless, in this section, I confirmed that several aspects of these representations were a result of learning, and such representations emerged when the RNN performed competently in the tasks, as shown in our simulations. This can be seen as an initial step towards linking such a sparse distributed representation with mixed selectivity and the task performance for which the RNN was optimised. More formal links will be established using probabilistic generative models (chapter 2) and information theory (chapter 6).

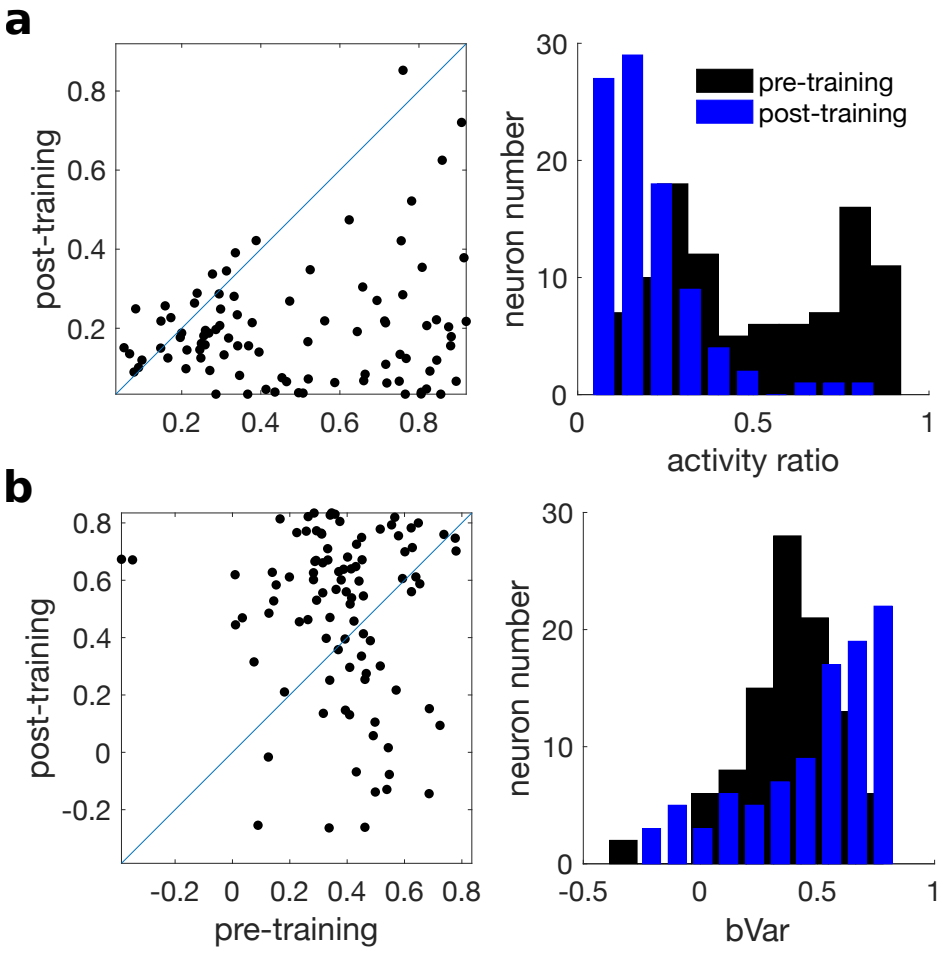


Fig. 4.4 Changes of activity ratio (**a**) and stereotypy (**b**) over training. The activity ratio measures lifetime sparseness for individual neurons, and bVar quantifies the variability of tuning curves explained by a template. At the population level, bVar increased from 0.37 to 0.47. Black and blue indicate the histograms before and after training respectively.

Fig. 4.5 Six example splitter cells. Spikes are shown as coloured dots at the locations of firing. Yellow and purple indicate left-turn and right-turn trials respectively, following the same convention as in Wood et al. (2000). The spikes were sampled with probabilities proportional to the activities (firing rates) $\mathbf{r}_t$. For visualisation purposes, Gaussian noise with standard deviation of 2.5 cm was added when plotting spikes to avoid overlap. All these 6 cells fired differently at the central arm depending on the type of the trial.

4.4 Neural dynamics

After examining the firing pattern of neurons, this section takes a more global view of how such representation functions together as a dynamical system. By treating the activities of hidden neurons in an RNN as the state-space of a dynamical system, I analyse the RNN using techniques based on local linearisation, where the behaviour of the system is characterised by its local Jacobian. Towards the end of this section, I will present an intriguing view of the developed tuning curves after taking into consideration the temporal dynamics.

4.4.1 The RNN as a dynamical system

To start our quest for understanding the RNN, especially the representations that developed with improved task performance, I analyse the RNN as a dynamical system. This is particularly convenient during the recall task — with the absence of external input (except the initial cue), the dynamics are self-sustained and the performance is completely determined by these dynamics.

The RNN described by equations 3.1 and 3.2 can be generalised by introducing a step size Δt and a temporal constant τ:

$$\tau \cdot \frac{\Delta \mathbf{y}_t}{\Delta t} = -\mathbf{y}_{t-1} + \mathbf{W}_{\text{in}} \cdot \mathbf{x}_t + \mathbf{W}_{\text{rec}} \cdot \mathbf{s}(\mathbf{y}_{t-1}) \tag{4.6}$$

$$\mathbf{y}_t = \mathbf{y}_{t-1} + \Delta \mathbf{y}_t \tag{4.7}$$

When $\frac{\tau}{\Delta t} = 1$, the above equations become equivalent to equations 3.1 and 3.2. Equation 4.6 and 4.7 are a discrete-time (first-order Euler) integration of the continuous system:

$$\tau \cdot \dot{\mathbf{y}} = -\mathbf{y} + \mathbf{W}_{\text{in}} \cdot \mathbf{x} + \mathbf{W}_{\text{rec}} \cdot s(\mathbf{y}) \tag{4.8}$$

This discrete approximation is accurate when $\mathbf{y}$ changes slowly in each step, which I will discuss more in later sections. When this RNN is used for recall, for which we plug-in $\mathbf{x} = \mathbf{W}_{\text{out}} \cdot \mathbf{s}(\mathbf{y})$, its state is purely determined by $\mathbf{y}$ as

$$\tau \cdot \dot{\mathbf{y}} = -\mathbf{y} + \boldsymbol{W} \cdot s(\mathbf{y}) \tag{4.9}$$

where we use the shorthand $\boldsymbol{W} = \mathbf{W}_{\text{in}}\mathbf{W}_{\text{out}} + \mathbf{W}_{\text{rec}}$.

Before the more detailed analysis, it is helpful to develop some intuition by viewing this dynamical system in a low-dimensional space. I used principal component analysis (PCA) to reduce the dimensionality of $\mathbf{y}$ collected over 20 trials (split for left- and right- turn trials). As an embedding method[2], the projection given by PCA tries to preserve the Euclidean distance between points in the space of $\mathbf{y}$. The projected trajectories of $\mathbf{y}$ from the same RNN shown in figure 4.1 is illustrated in figure 4.6 (viewed from 3 different angles).

From figure 4.6, we see that the trajectories lie on a highly restricted and structured manifold inside the space of all possible firing configurations. Trajectories of the same trial type (blue or red) are clustered together in the state space, while trajectories of different types (blue vs. red) are well separated. In particular, the trajectories along the shared central arm (solid lines) are well separated in the state space, despite the RNN receiving noisy samples of identical positions as inputs during training. Therefore, consistent with the evidence from splitter cells (Figure 4.5), the central arm was indeed represented differently by the hidden neurons, indicating disambiguation at the decision point.

[2] A probabilistic view of PCA is reviewed in section 2.5, under the context of autoencoders.

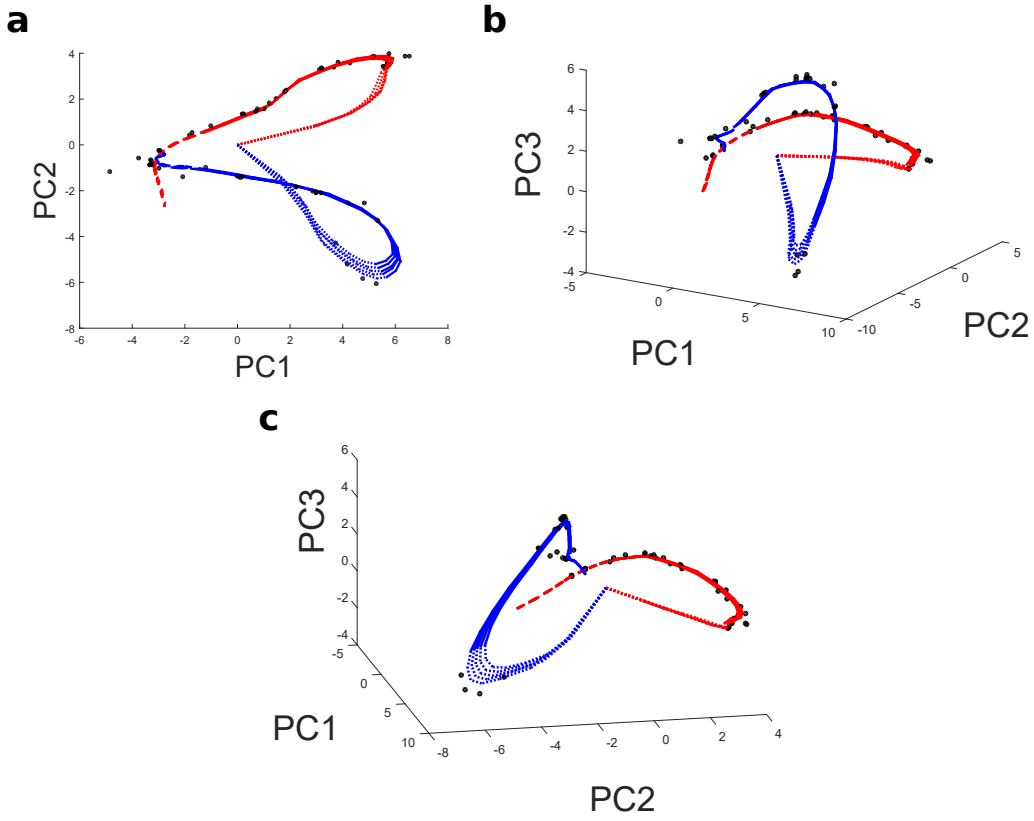


Fig. 4.6 Temporal evolution of hidden neuron states $\mathbf{y}$. The trajectories of $\mathbf{y}_t$ in 10 trials are projected along the first 3 principal components computed from neural activities. (For this, $\mathbf{y}_t$ at different time steps were treated as independent data points when computing the principal components. 77.3% of variance in the activities was preserved through this projection). The 3 plots are the same trajectories viewed from 3 different angles. Following the colour code used in figure 4.1, blue and red indicate left-turn and right-turn trials respectively. Dotted lines indicates the segment before entering the central arm, and dashed lines shows the segment after the central arm. Black dots are the numerically found slow points along the trajectories (see section 4.4.2 for explanation).

4.4.2 Slow point analysis

Analysis of non-linear dynamical system has traditionally been focusing on stationary points, where linearisation of non-linear dynamics captures the behaviour of the system locally. However, the requirement of stationary points may be over restrictive, limiting region of state-space on which linearisation can be applied. Here, following Sussillo and Barak (2013), I analyse the RNN through a less restricted class of points, called slow points, where techniques based on local linearisation provides useful insights in understanding the albeit non-linear RNN.

For a dynamical system defined as

$$\dot{\mathbf{y}} = F(\mathbf{y}) \tag{4.10}$$

it has the Taylor expansion at $\mathbf{y}_c$

$$\dot{\mathbf{y}} = F(\mathbf{y}_c + \delta\mathbf{y}) = F(\mathbf{y}_c) + \frac{\partial F(\mathbf{y}_c)}{\partial \mathbf{y}_c}\delta\mathbf{y} + \frac{1}{2}\delta\mathbf{y}^\intercal \frac{\partial^2 F(\mathbf{y}_c)}{\partial \mathbf{y}_c^2}\delta\mathbf{y} + \cdots \tag{4.11}$$

By linearisation, we ignore all except the linear term in the Taylor expansion. A fixed point $\mathbf{y}_0$, such that $F(\mathbf{y}_0) = \mathbf{0}$, is an obvious candidate for linearisation, since the constant term already vanishes. Sussillo and Barak (2013) demonstrate that other less restricted $\mathbf{y}^*$ may still be good candidates for linearisation, as long as

$$\left\| \frac{\partial F(\mathbf{y}^*)}{\partial \mathbf{y}^*}\delta\mathbf{y} \right\| > \|F(\mathbf{y}^*)\| \tag{4.12}$$

$$\left\| \frac{\partial F(\mathbf{y}^*)}{\partial \mathbf{y}^*}\delta\mathbf{y} \right\| > \left\| \frac{1}{2}\delta\mathbf{y}^\intercal \frac{\partial^2 F(\mathbf{y}^*)}{\partial \mathbf{y}^{*2}}\delta\mathbf{y} \right\| \tag{4.13}$$

i.e., when the linear term dominates. In practice, rather than testing the exact conditions described by equation 4.12 and 4.13, we simply look for the *slow points* $\mathbf{y}^*$ where $F(\mathbf{y}^*)$ is small (Sussillo and Barak, 2013).

These slow points can be found by minimising an auxiliary function

$$q(\mathbf{y}) = \frac{1}{2}\|F(\mathbf{y})\|^2 \tag{4.14}$$

with respect to $\mathbf{y}$. During minimisation, gradient-based optimisation methods are used to find stationary points (or locations close to them) of $q(\mathbf{y})$ that satisfy

$$\frac{\partial q(\mathbf{y}_0)}{\partial \mathbf{y}_0} = \frac{\partial F(\mathbf{y}_0)}{\partial \mathbf{y}_0}\dot{\mathbf{y}}_0 = \mathbf{0} \tag{4.15}$$

which implies one of the two possibilities:

1. $\dot{\mathbf{y}}_0 = \mathbf{0}$: $\mathbf{y}_0$ is a fixed point, thus a global minimum of $q(\mathbf{y})$.
2. $\dot{\mathbf{y}}_0 \neq \mathbf{0}$, but $\frac{\partial F(\mathbf{y}_0)}{\partial \mathbf{y}_0}\dot{\mathbf{y}}_0 = \mathbf{0}$: either $\frac{\partial F(\mathbf{y}_0)}{\partial \mathbf{y}_0} = \mathbf{0}$ or $\dot{\mathbf{y}}_0$ is a linear combination of the zero eigenvectors of $\frac{\partial F(\mathbf{y}_0)}{\partial \mathbf{y}_0}$. In this case, $\mathbf{y}_0$ is a local minimum.

Therefore, we need to check whether a stationary point $\mathbf{y}_0$ of $q(\mathbf{y}_0)$ qualifies as a slow point by comparing it with a threshold value (Sussillo and Barak, 2013). The exact value of this threshold value is problem and model dependent; in my experiment, the value of 2 resulted in the slow points densely concentrated near the recall trajectories. Initialised from 100 randomly chosen points in the state space near the recall trajectories, I found the 84 slow points shown by the black dots in Figure 4.6 which were close to the trajectories. To verify that points along the trajectories indeed had low speed, as the strings of slow points indicate, I tested how the speeds changed if these slow points were pushed away from the actual trajectories. This is simulated by perturbing the system at the slow points with Gaussian noise and computing speeds at these perturbed points using equation 4.14. The averaged speed as a function of the perturbing noise level is shown in figure 4.7. The upwards curve shows that the speed of the system increased as the states were perturbed away from the slow points.

We can thus linearise the (continuous) dynamical system at these slow points, and analyse the behaviours near them from their corresponding eigen-spectra. Figure 4.8 (red dots) shows the eigenvalues at four of the randomly chosen slow points plotted in the complex plane. In these graphs, each point represents the eigenvalue for an eigenvector. The directions whose eigenvalues have negative real parts are stable. We can see that most, but not all, of the directions along these eigenvectors were stable. As controls, the eigen-spectra for the same states before training (Figure 4.8, blue dots) and in a randomly initialised RNN with Gaussian random recurrent weights (random RNN) [3] (Figure 4.8, green dots) are also plotted in the same panels. The eigenvalues obtained from the random RNN are concentrated near (-1, 0) in the complex plane, resulting in a few *meta-stable* directions whose eigenvalues have real-parts close to 0. Note that, when the recurrent connection matrix was initialised close to identity (Le et al., 2015), a significant portion of directions at these state, either before or after training, were already meta-stable or near unstable (Figure 4.8).

Such features of individual slow points are confirmed in the histograms from all the 84 slow points shown in Figure 4.9 (**a**), where we can see a concentration of eigenvalues' real parts at both -1, corresponding to immediately vanishing of input, and around 0,

[3] In this random RNN, weights are scaled according to, for example, Martens and Sutskever (2011), instead of setting the connection matrix close identity as in Le et al. (2015).

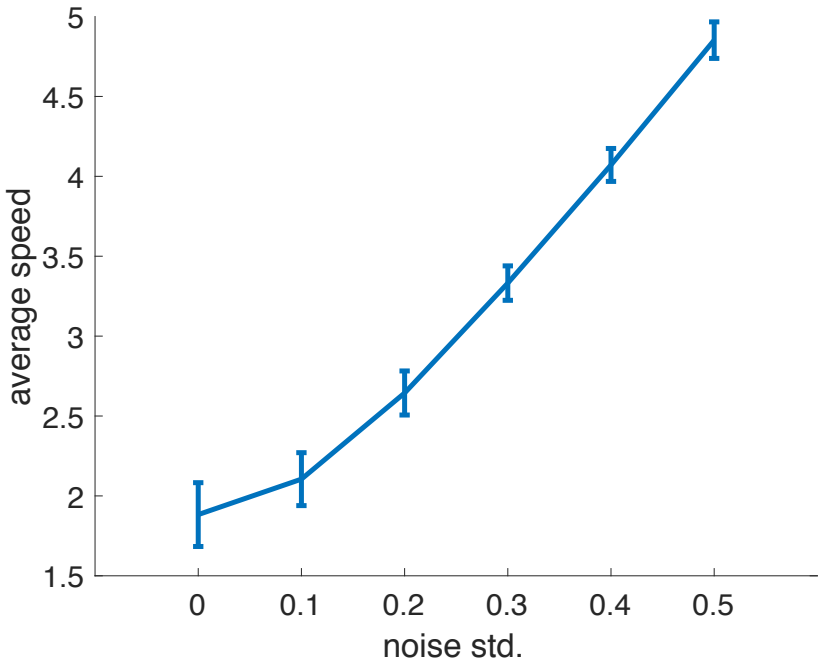


Fig. 4.7 Averaged speed of points near the recall trajectories as a function of perturbing noise level. Points away from the recall trajectories generally have higher speed. Lines and error bars shows the means and standard deviations computed from 30 RNNs.

indicating meta-stable states for the pre- (blue) and post-training (red) RNNs. However, the concentration of eigen values near 0 disappeared for the random RNN (Figure 4.9, **a**, green). Figure 4.9 (**b** and **c**) further show the histograms for the spectral abscissa and the proportions of unstable directions at all the 84 slow points. Therefore, the strings of these slow points formed *quasi-* line attractors — they were not exact attractors, as some of the directions at these points were not stable. Indeed, these few unstable directions guaranteed the system at these states did not stop, but proceeded along the trajectories specified by these line attractors, which the recall dynamics required (Ganguli et al., 2008; Sussillo, 2014).

Despite the popular hypothesis that neural systems use attractors for computations, it has been know in machine learning that stable attractors are unsuitable for learning systems since they necessarily exhibit the vanishing gradient problem (Bengio et al., 1994; Jozefowicz et al., 2015). Consistent with this argument against stable attractors, the recall trajectories faithfully followed quasi-line attractors. Moreover, slight digression caused by noise could be corrected by the stable attractor-like dynamics, which explains the RNN's robustness against input noise. In addition, my analysis justifies initialising the recurrent connection matrix near identity (Le et al., 2015). This technique is helpful because it yields a large number of meta-stable directions in the state-space, which is desirable for learning the quasi-line attractor dynamics.

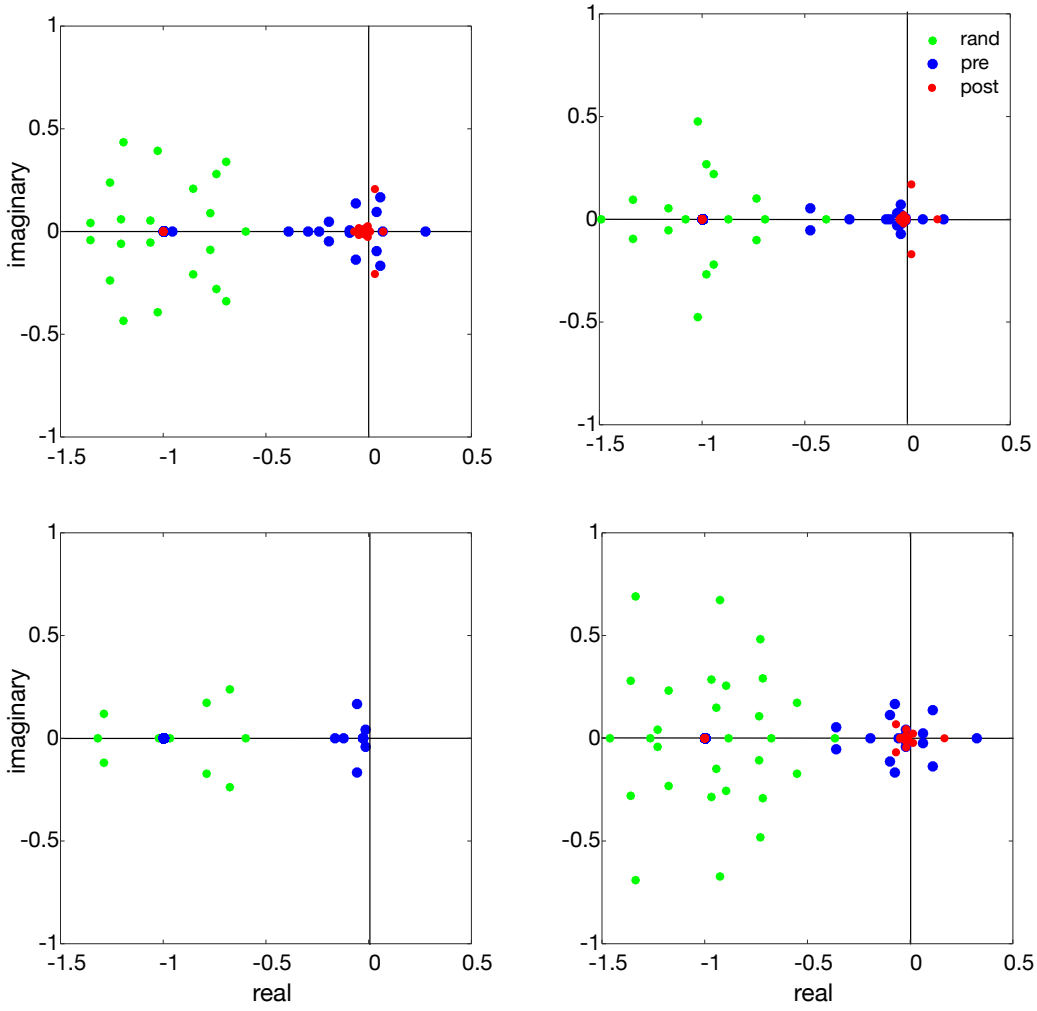


Fig. 4.8 Eigenvalues of 4 randomly chosen slow points. Blue and red represent the eigenvalues before and after training at the same states (slow points for the trained RNN). For comparison, green dots show the eigenvalues from an RNN with Gaussian random recurrent connections the same states. Note there are in total 100 dots in each plot for the 100 dimensions of the state space, but over half of these eigenvalues overlap at $(-1,0)$, corresponding to immediate vanishing of inputs along these directions. Therefore, most directions at these slow points were stable, as the real parts of most eigenvalues were negative.

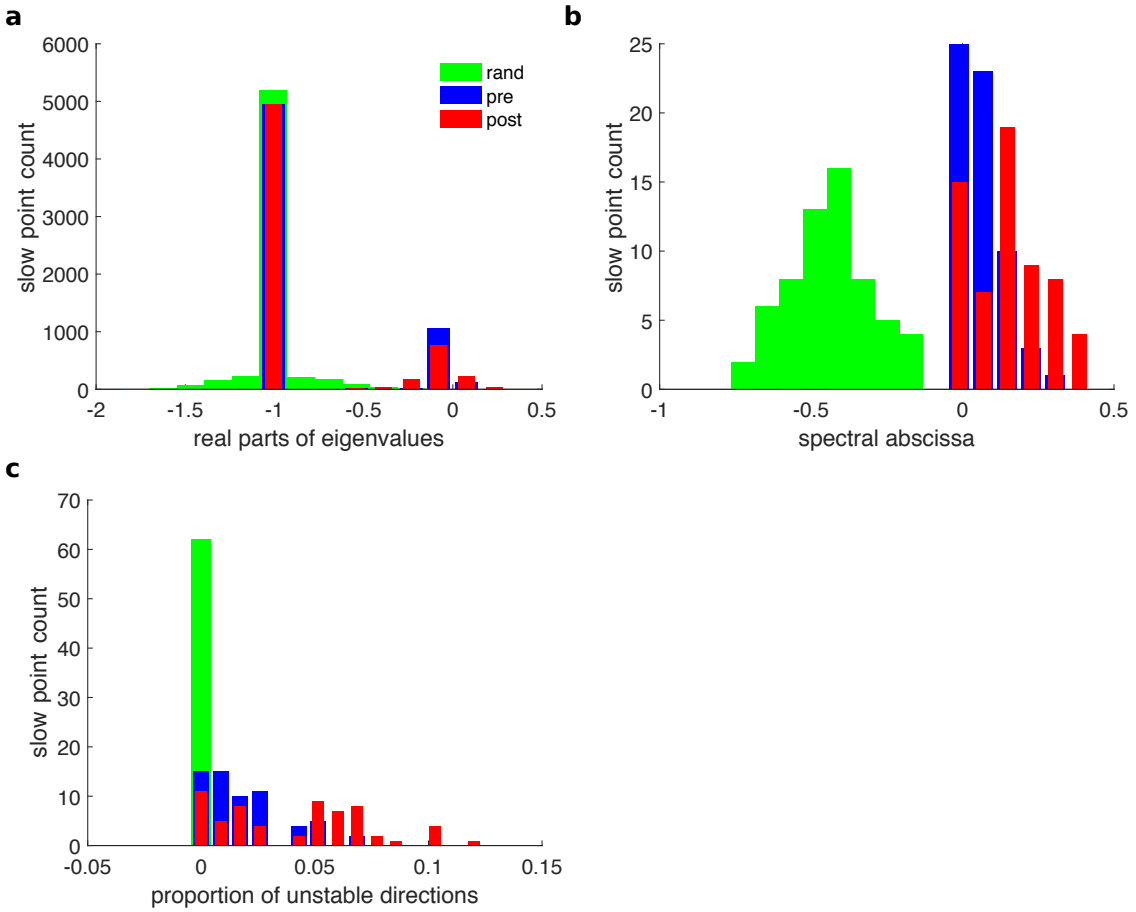


Fig. 4.9 Histograms for the real part of all eigenvalues around slow points (**a**), spectral abscissa (**b**), and the proportion of unstable directions (**c**) at all 84 slow points. Colour codes different RNNs as in Figure 4.8.

4.5 Adaptive tuning curves

To conclude this chapter, we present another, slightly unusual view of the dynamics of the RNN — we show how the tuning curves as functions of *feed-forward* inputs adapted during the task. Although the ReLU's threshold nonlinearity as a function of the *total input* is fixed (i.e., $\mathbf{r}_t$ as a function of $\mathbf{y}_t$, equation 3.2), one can consider the recurrent connections and inputs through these recurrent connections as a mechanism to modulate the tuning of the hidden layer neurons to their feed-forward inputs. To view these dynamically adapting tuning curves, we plot the activities of each hidden neuron ($\mathbf{r}_t$) as a function of its feed-forward input ($\mathbf{W}_{\text{in}} \cdot \mathbf{x}_t$) in Figure 4.10. I set a firing threshold of 0.1, and only analyse neurons whose firing crossed this threshold for over 20 times steps. In the end I obtained 85 neurons out of the population of 100 neurons that satisfy this criterion for plotting tuning curves.

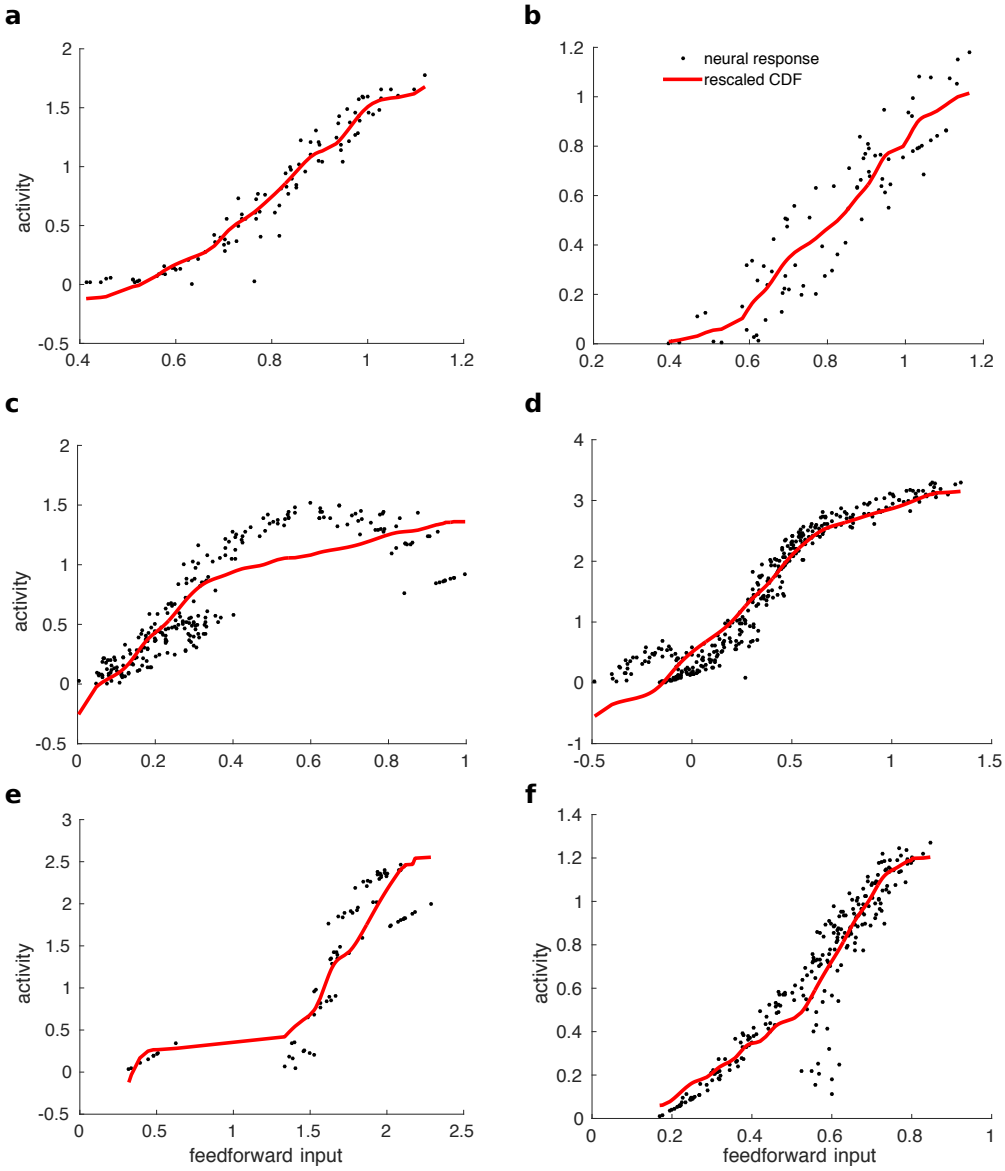


Fig. 4.10 After training, the activities of 6 representative hidden neurons are plotted against the feed forward inputs into these neurons (black dots). Superimposed are the cumulative distribution functions of the feed-forward inputs (red lines), scaled and shifted to fit the activities. Each black dot represents the activity of a neurons at a time step. Data are collected in 10 trials.

These adaptive tuning curves fit surprisingly well the CDF of the feed-forward inputs (Figure 4.10). It is known that a tuning curve with the shape of input CDF maximises the entropy of output neural responses with bounded activity range (see section 2.5.3). This is because the CDF transforms the input distribution into an uniform output distribution, which has the maximum entropy when the range of firing rates is bounded. Therefore, these adaptive tuning curves may maximise the entropy of individual neural responses.

To verify that the matching of tuning curves and their CDFs was not coincident, I shuffled the pairs of tuning curves and corresponding input CDFs, and compare the histogram of correlation coefficients before and after the shuffling, both before and after training (Figure 4.11). The clear decrease of correlation coefficients after shuffling suggests the matching illustrated in Figure 4.10 was not a trivial phenomenon. Figure 4.12 further illustrated the change of correlation coefficients for each neuron. Out of the the 85 neurons I analysed, 55 of them (64.7%) increased the correlation coefficient after training. This shift towards higher correlation coefficient after training suggests training at least partly accounted for this phenomenon.

In summary, analysis of neural responses show that the activities of neurons in RNNs dynamically adapted to input statistics. When the recurrent connections were considered as a part of the neurons adaptive mechanism, individual neurons' tuning curves seemed to match their input CDFs. The connection between this phenomenon and information maximisation is first explored in ICA literature. More recently, it has been noted by, for example Dinh et al. (2016), that denoising training of neural networks is related to ICA by encouraging the model to explain observations using independent additive noise. However, the exact contribution of independence in such models is obscured by their non-linearity, and better explanation of this phenomenon awaits future investigation.

Conclusion

This section presents the main experiments in this thesis, where RNNs were trained to solve the T-maze alternating task, formulated as sequence prediction. The RNN successfully solved the prediction task and could generalised to more challenging recall task. Moreover, analysis of the firing patterns of the trained RNN shows its similarity to representations observed in the brain both qualitatively and quantitatively. To understand the RNN's performance, I analysed its dynamics using a recently proposed technique based on a generalisation of fixed points, called slow points. Linearisation around these slow-points reveals the quasi-line attractor dynamics underlying the RNN's robust performance.

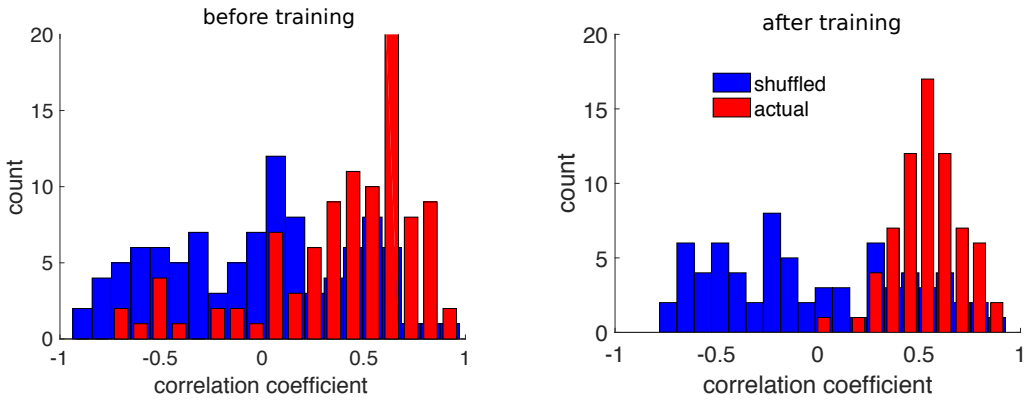


Fig. 4.11 Histograms of the correlation coefficients between the tuning curves and their corresponding input CDFs for actual (red) and shuffled (blue) tuning curves, before (left) and after (right) training.

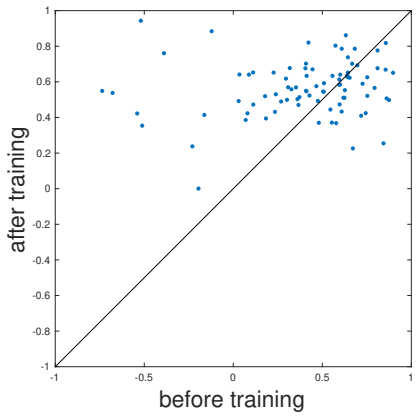


Fig. 4.12 The change of the correlation coefficients for each neuron after training. The correlation coefficients had increased for 55 out of the 85 neurons (64.7%) analysed.

In addition, I presented a novel perspective of inspecting tuning curves from neural networks. Tuning curves have been important features for studying neural computation (Dayan and Abbott, 2001). Although it is more straight forward to interpret the activation function, such as ReLU or sigmoid function as the analogues of tuning curves, here I show an alternative: when taking into consideration the influence from all neighbouring neurons and connections, the combined tuning curves may exhibit previously unseen connections with known computational principles. This is a direction for future investigation.

Chapter 5

Mixed selectivity

This chapter formally discusses the concept of mixed selectivity, which presents a new perspective for analysing both biological and synthetic neural data. As discussed in section 1.2, in practice mixed selectivity depends on well-defined features suitable for an experiment. In a biological experiment, these features are both relevant to the brain region under investigation and are well controlled. In a machine learning experiment, these features are either directly used as input or are indirectly presented but are known to affect the task performance. For example, in the T-maze experiment (e.g., section 1.3.1) examined in this thesis, the current as well as previous locations (context) are essential for solving the task, so they are considered as features.

Once the features are defined, this chapter starts with examining the existing measure of mixed selectivity from Rigotti et al. (2013). A new measure of mixture selectivity is then proposed based on information theory. The two measures of mixed selectivity are analysed both theoretically and experimentally. My analysis suggests that, although both measures reflect the degree neurons representing different features, the newly proposed *mixing index* has more favourable theoretical properties and better correlates with model behaviour.

5.1 Measuring Mixed Selectivity

Rigotti et al. (2013) proposed mixed-selectivity as an important property of cortical neurons. It is particularly relevant to neural computation, because of its strong correlation with the performance on behavioural tasks. Neurons that are tuned to multiple task-related features have been observed in multiple areas, such as the prefrontal cortex (Baeg et al., 2003; Fujisawa et al., 2008; Huettel et al., 2002; Mante et al., 2013; Rigotti et al., 2013), the hippocampus (Agster et al., 2002; Brown et al., 2010; Ginther et al., 2011; Markus et al., 1995; Pastalkova et al., 2008; Wood et al., 2000), and the posterior parietal cortex (PPC,

Harvey et al., 2012). Specifically, in sequence learning, mixed selectivity can be used to encode contextual information, and thus to disambiguate sequences with overlapping segments (Fujisawa et al., 2008; Ginther et al., 2011; Harvey et al., 2012; Pastalkova et al., 2008; Wood et al., 2000). Although the same information can be carried in pure-selectivity representations, this chapter argues that mixed selectivity is more robust to noise.

Despite the important computational role of mixed selectivity in various neural systems (Section 1.2), measuring mixed selectivity is non-trivial. In particular, the *mixed selectivity index* proposed by Rigotti et al. (2013) is limited by the requirement of model-fitting. As a result, it cannot be easily generalised to tasks with continuous stimuli, where enumerating the nonlinear combinations of stimuli is impossible and discretising to sufficient accuracy can be computationally infeasible. Moreover, the mixed selectivity index does not take into account variance around tuning curves given the same stimuli, i.e., noise variance, despite its significant effects on neural coding (Averbeck et al., 2006). In this chapter, I first analyse the mixed selectivity index (Rigotti et al., 2013), then propose a new measure of mixed selectivity, *mixing index*. It is motivated by information theory, and does not rely on model-fitting (i.e., model-free).

I compare these two different measures of mixed selectivity through simulation on both synthetic population coding examples and the RNNs trained on the T-maze alternation task used in section 4.1, showing that the new mixing index is superior to the mixed selectivity index proposed by Rigotti et al. (2013) in being more predictive of behavioural performance. Further, our mixing index is mathematically more convenient; I will show that training of a robust generative model (Chapter 2) simultaneously maximises a lower-bound of the mixing index.

5.2 Measure of mixed selectivity based on model fitting

I first review the definition of mixed selectivity index proposed by Rigotti et al. (2013), which I will refer to as ζ. To be consistent with the rest of the thesis, the notations and formulations are different from the original paper. We discretise continuous stimuli when necessary to employ the model-fitting procedure originally developed for discrete stimuli by Rigotti et al. (2013), despite the fact that, as we shall see, this would result in exponentially many parameters to fit.

A neuron's response $\mathbf{r}$ is assumed to be a function of M stimuli $\{\mathbf{x}_i\}_{i=1}^{M}$ [1]. Stimulus $\mathbf{x}_i$ may take one of K_i discrete values, which is encoded in a column vector using an 1-in-K_i (one-hot) coding, so that $\mathbf{x}_i(j) = 1$ means that the j'th value at stimulus i is presented (all

[1] A stimulus can be any task relevant variable, such as trial type, context or cue.

other $K_i - 1$ elements of $\mathbf{x}_i$ are zeros). We decompose the neural response $\mathbf{r}$ into two parts, a linear component and a non-linear component [2]:

$$\mathbf{r} = \mathbf{r}_{\text{linear}} + \mathbf{r}_{\text{non-linear}} \tag{5.1}$$

The linear component is a linear combination of all the M stimuli:

$$\mathbf{r}_{\text{linear}} = \boldsymbol{\beta}_0 + \sum_{i=1}^{M} {\boldsymbol{\beta}^i_{\text{linear}}}^{\mathsf{T}} \cdot \mathbf{x}_i \tag{5.2}$$

where $\boldsymbol{\beta}_0$ is a bias, $\boldsymbol{\beta}^i_{\text{linear}}$ is a column vector with K_i elements as the linear coefficients for stimulus i. Conversely, the non-linear component is defined as

$$\mathbf{r}_{\text{non-linear}} = \sum_{j_1=1}^{K_1} \sum_{j_2=1}^{K_2} \cdots \sum_{j_M=1}^{K_M} \boldsymbol{\beta}^{j_1,j_2,\ldots,j_M}_{\text{non-linear}} \cdot \prod_{i=1}^{M} \mathbf{x}_i(j_i) \tag{5.3}$$

where all possible combinations of the M stimuli are enumerated. Therefore, $\prod_{i=1}^{M} K_i$ additional parameters are required as the coefficients for all the combinations of stimuli.

Both equation 5.2 and 5.3 are linear in the parameters, so finding the parameters is simply a linear regression problem, which can be solved analytically using, for example, the normal equation. However, since there as many parameters $\beta_{\text{non-linear}}$ as the number of trial types (all possible trial types are enumerated in equation 5.3), we can always over-fit $\beta_{\text{non-linear}}$ with trial-averaged firing rates. As a result, the linear component (equation 5.2) has to be fitted first, and equation 5.3 is subsequently used to fit the residuals (Rigotti et al., 2013). The order of fitting is important, and the two components cannot be fitted jointly. This process can be problematic, as the linear component may over fit the data. Unlike coordinate descent where individually fitted coefficients are iteratively improved, the possibly over-fitted linear coefficients are never revisited, so they cannot be corrected.

Based on the fitted models, the *mixed selectivity index* [3] measures the difference between variances explained by the linear and nonlinear components, σ^2_{linear} and $\sigma^2_{\text{nonlinear}}$ respectively:

$$\zeta = \frac{\sigma^2_{\text{nonlinear}} - \sigma^2_{\text{linear}}}{\sigma^2_{\text{nonlinear}} + \sigma^2_{\text{linear}}} \tag{5.4}$$

[2] In the original paper of Rigotti et al. (2013), these two components are somewhat confusingly called linear and non-linear mixed selectivity components, as if r always had mixed selectivity.

[3] We use this term in accordance with Rigotti et al. (2013). However, it is perhaps more appropriately termed as *nonlinear* mixed selectivity index, to emphasise the removal of the linear term.

Here $\mathbf{r}$ are *trial-averaged* firing rates, the non-linear model can perfectly fit the residual, so $\sigma^2_{\text{nonlinear}} + \sigma^2_{\text{linear}}$ would be the total variance of $\mathbf{r}$.

However, this measure of mixed selectivity is insensitive to noise variance across different trials. To see this, assume $\mathbf{r}$ were neural activities from *individual trials*, instead of those average across trials that are actually used in computing mixed selectivity index. In this case, the same linear model can still fit the data. As a result, the same $\sigma^2_{\text{nonlinear}}$ and σ^2_{linear}, thus the same ζ are obtained despite using data from individual trials. In other words, the variance from noise across trials is always ignored in this formula, even if additional activity data from individual trials were provided. As a result, we would obtain the same ζ regardless of the amount of noise presented across trials. This behaviour is at odds with the fact that noise variance may significantly affect decoding (Averbeck et al., 2006). To solve this problem, we need a different way to measure mixed selectivity.

5.3 Mixing index: an information theoretic measure

To address the problem of the mixed selectivity index proposed by Rigotti et al. (2013), I propose a different measure, which I call *mixing index* and denote it by κ. It is free from model-fitting and takes noise variability into consideration. We start with a representation of two *independent* scalar variables x_1 and x_2. The independence assumption is based on the fact that mixed selectivity is only relevant when the inputs are independent. Otherwise, the representation $\mathbf{r}$ will always be selective to both x_1 and x_2 because of the correlation between the inputs themselves. Before presenting the formal definition of our mixing index κ, we review the motivations behind the proposal of the mixed selectivity index ζ.

Mixed selectivity is motivated from the information encoded by neurons (Fusi et al., 2016; Rigotti et al., 2013). As noted by Fusi et al. (2016), which we quote below, mixed selectivity generally means the tuning of a neuron is context dependent:

> The neurons behave the same way in the same context, but their selectivity is highly context-dependent. As a consequence, the activity of any individual mixed selectivity neuron doesn't mean anything by itself. Only in the context of other neurons it is possible to disambiguate the information encoded by mixed selectivity neurons.

Consequently, the information that is encoded in a context-dependent way can only be decoded when the context as well as the firing of a neuron is given. This idea can be formally expressed using conditional mutual information: while the mutual information $\mathrm{I}[\mathbf{r}; x_1]$ expresses the amount of information about x_1 that can be decoded from $\mathbf{r}$ alone,

the conditional mutual information $\mathrm{I}[\mathbf{r};x_1|c]$ expresses the information about x_1 that can be decoded from $\mathbf{r}$ when the context c is available. Note that the context may come from other neurons, and utilising context information from other neurons is consistent with the view that ensembles of neurons, instead of individual neurons, are the functional units in neural computation (Yuste, 2015).

From this view, it is natural to use the the difference $\mathrm{I}[\mathbf{r};x_1|c]-\mathrm{I}[\mathbf{r};x_1]$ as a measure of mixed selectivity of $\mathbf{r}$. Since the context c can be interpreted as another independent stimulus, which we denote by x_2, the mixed selectivity of $\mathbf{r}$ as a representation of $\mathbf{x}=\{x_1,x_2\}$ can be defined as

$$\kappa(\mathbf{r},\mathbf{x})=\mathrm{I}[\mathbf{r};x_1|x_2]-\mathrm{I}[\mathbf{r};x_1] \tag{5.5}$$

Here κ is also equivalent to the negative of *interaction information* (McGill, 1954), which is defined as $\mathrm{I}[x;y;z]=\mathrm{I}[x;y]-\mathrm{I}[x;y|z]$ to quantify the interaction between random variables x, y and z. Generally speaking, the interaction information $\mathrm{I}[x;y;z]$ can be positive, negative or zero. Intuitively, positive interaction information indicates that the third variable z facilitates the correlation between x and y, while negative interaction information indicates that z inhibits the correlation between x and y. It can thus be zero when the correlation between x and y is unaffected by z. For example, when the context (z) shifts neural activity (x) in an *orthogonal* direction to that shifted by location (y), despite that the neuron population is selective to the context, the difference between the conditional mutual information $\mathrm{I}[x;y|z]$ and the marginal $\mathrm{I}[x;y]$ is zero. However, as will be discussed in the rest of this chapter, under the assumed graphical model in Figure 5.1, κ is always non-negative. In particular, the situation of $\kappa=0$ is generally not true given *conditionally dependency* (equation 5.13), which is violated in the above example.

To understand κ, first note that equation 5.5 is symmetric, which can be shown using the conditional mutual information equality

$$\mathrm{I}[\mathbf{r};x_1|x_2]=\mathrm{I}[\mathbf{r};x_1,x_2]-\mathrm{I}[\mathbf{r};x_2] \tag{5.6}$$

so that

$$\kappa(\mathbf{r},\mathbf{x})=\mathrm{I}[\mathbf{r};x_1,x_2]-\mathrm{I}[\mathbf{r};x_1]-\mathrm{I}[\mathbf{r};x_2] \tag{5.7}$$

It is straightforward to prove that the symmetry also applies in the general case of N stimuli $\mathbf{x}=\{x_1,x_2,\ldots,x_N\}$ as

$$\begin{aligned}\kappa(\mathbf{r},\mathbf{x})&=\mathrm{I}\left[\mathbf{r};x_n|x_{\setminus n}\right]-\mathrm{I}[\mathbf{r};x_n]\\&=\mathrm{I}[\mathbf{r};x_1,x_2,\ldots,x_N]-\sum_{n=1}^{N}\mathrm{I}[\mathbf{r};x_n]\end{aligned} \tag{5.8}$$

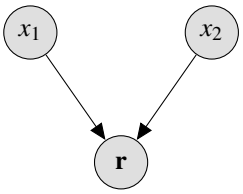


Fig. 5.1 The directed graphical model illustration of the dependencies between the stimuli $\mathbf{x} = \{x_1, x_2\}$ and the neural response $\mathbf{r}$. The arrows indicate that the response is affected by the two external stimuli. Although x_1 and x_2 are marginally independent from each other, they generally become *conditionally dependent* (a.k.a. *explaining away*) given the response $\mathbf{r}$, such that $p(x_1|\mathbf{r})p(x_2|\mathbf{r}) \neq p(x_1, x_2|\mathbf{r})$, unless one of them is independent of $\mathbf{r}$ (i.e., when one of the arrows is absent).

where $x_{\setminus n}$ indicates all inputs except x_n, which can be interpreted as the context for decoding x_1, as in the case of two stimuli. From the symmetry, it follows that the specific choice of x_n does not matter for the value of κ. Using the symmetric form of κ, we now have a formal definition of this measure of mixed selectivity:

Definition 1. *The mixed selectivity of a representation* $\mathbf{r}$ *of independent input variables* $\mathbf{x} = \{x_1, x_2\}$ *is measured by the* mixing index

$$\kappa(\mathbf{r}, \mathbf{x}) = \mathrm{I}[\mathbf{r}; x_1, x_2] - \mathrm{I}[\mathbf{r}; x_1] - \mathrm{I}[\mathbf{r}; x_2] \tag{5.9}$$

Equation 5.9 has an intuitive interpretation: the mixed selectivity is the *additional* amount of information $\mathbf{r}$ conveys about x_1 and x_2 *together*, compared with what $\mathbf{r}$ conveys about x_1 and x_2 *alone*.

This measure has several favourable properties. First, $\kappa(\mathbf{r}, \mathbf{x})$ is symmetric as we have already discussed. It is also non-negative. When x_1 and x_2 are independent, conditional mutual information obeys the inequality $\mathrm{I}[\mathbf{r}; x_2|x_1] - \mathrm{I}[\mathbf{r}; x_2] \geqslant 0$. Therefore, knowing the equality of equation 5.6,

$$\kappa(\mathbf{r}, \mathbf{x}) \geqslant \mathrm{I}[\mathbf{r}; x_1, x_2] - \mathrm{I}[\mathbf{r}; x_1] - \mathrm{I}[\mathbf{r}; x_2|x_1] = 0 \tag{5.10}$$

As a proper measure, $\kappa(\mathbf{r}, \mathbf{x}) = 0$ correctly reflects the lack of mixed selectivity. First, $\kappa(\mathbf{r}, \mathbf{x}) = 0$ *if* $\mathbf{r}$ is independent of x_1 or x_2. Assuming $\mathbf{r}$ is independent of x_2,

$$\begin{aligned} \mathrm{I}[\mathbf{r}; x_1, x_2] &= \mathrm{H}[\mathbf{r}] - \mathrm{H}[\mathbf{r}|x_1, x_2] \\ &= \mathrm{H}[\mathbf{r}] - \mathrm{H}[\mathbf{r}|x_1] \\ &= \mathrm{I}[\mathbf{r}; x_1] \end{aligned} \tag{5.11}$$

Using this equality and the mutual information $\mathrm{I}[\mathbf{r};x_2] = 0$,

$$\begin{aligned}\kappa(\mathbf{r},\mathbf{x}) &= \mathrm{I}[\mathbf{r};x_1,x_2] - \mathrm{I}[\mathbf{r};x_1] - 0 \\ &= 0\end{aligned} \tag{5.12}$$

Further, $\kappa(\mathbf{r},\mathbf{x}) = 0$ *only if* $\mathbf{r}$ does not depend on either x_1 or x_2. This can be seen from an alternative form

$$\kappa(\mathbf{r},\mathbf{x}) = \mathrm{H}[x_1|\mathbf{r}] + \mathrm{H}[x_2|\mathbf{r}] - \mathrm{H}[x_1,x_2|\mathbf{r}] \tag{5.13}$$

Thus, $\kappa(\mathbf{r},\mathbf{x}) = 0$ indicates $\mathrm{H}[x_1|\mathbf{r}] + \mathrm{H}[x_2|\mathbf{r}] - \mathrm{H}[x_1,x_2|\mathbf{r}] = 0$, or $p(x_1|\mathbf{r})p(x_2|\mathbf{r}) = p(x_1,x_2|\mathbf{r})$, the conditional independence of x_1 and x_2 given $\mathbf{r}$. Because of the statistical structure between the stimuli and response, as illustrated in Figure 5.1, conditional independence is possible only when $\mathbf{r}$ is independent of at least one of x_1 and x_2. Otherwise, in the language of **?**, one of the stimuli always "explain away" another in this case. Therefore, the $\kappa(\mathbf{r},\mathbf{x}) = 0$ *if and only if* $\mathbf{r}$ has no mixed selectivity.

The definition and properties of κ in definition 1 can be extended to the general multi-variable case:

Definition 2. *The mixed selectivity of a representation* $\mathbf{r}$ *of independent input variables* $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$ *is measured with the mixing index*

$$\kappa(\mathbf{r},\mathbf{x}) = \mathrm{I}[\mathbf{r};x_1,x_2,\ldots,x_N] - \sum_{n=1}^{N} \mathrm{I}[\mathbf{r};x_n] \tag{5.14}$$

The above properties for two-variable mixed selectivity can be proved for the general case using induction.

Compared with the mixed selectivity index ζ, it is clear that κ does not involve model fitting. While ζ relies on model-fitting to assess how the neural response $\mathbf{r}$ is influenced by input stimuli $\mathbf{x}$, κ captures the dependencies using mutual information. In addition, it is easy to show that mutual information decreases with increased noise variance. Therefore, our definition of κ automatically accounts for the noise variance of $\mathbf{r}$. However, like other methods requiring estimation of entropy or mutual information, estimating κ in practice, especially for high-dimensional neural data, can be difficult (Golomb et al., 1997; Paninski, 2003; Strong et al., 1998). At the end, note that κ is formally similar to *synergy* , which is defined in a "reverse" direction and used to quantify the dependency between different neural responses in decoding the same external stimulus (Latham, 2005).

5.4 Examples of extreme representations

To illustrate the difference between the measures ζ and κ, this section introduces 4 extreme representations (figure 5.2). In contrast to the biologically plausible neural representation we reported in section 4.3, these 4 representations are extreme cases in terms of their mixed selectivity and the regularity of firing patterns.

To be consistent with the T-maze task I use throughout this thesis, the external stimuli that a population of neurons need to encode are location (a continuous variable) and context (a binary variable). The spectrum of possible representations have different degrees of mixed selectivity and regularity (Figure 5.2). At one end of the selectivity axis, neurons without mixed selectivity are tuned to either context or location, but never to both (pure selectivity neurons). As illustrated in figure 5.2 **a** and **c**, these neurons can be categorised as "context neurons" or "location neurons" according to their selectivity. In contrast, neurons with highly mixed selectivity are tuned to both context and position, which results in neurons in figure 5.2 **b** and **d**. Importantly, the mixed selectivity for neurons in **b** and **d** are nonlinear (Rigotti et al., 2013), as switching context does not simply change the average level of activities.

Along the axis of tuning curve regularity, there are extremely regular and extremely irregular neurons. The neural tuning curves in figure 5.2 **a** and **b** are shifted versions of one another; the shifting is required to tile the entire space of relevant stimuli (locations). In contrast, tuning curves in figure 5.2 **c** and **d** are almost entirely random and are very different from each other (Note the context-invariant in **c**).

None of the four examples in Figure 5.2 resembles the biologically plausible neural representation in our RNN (Figure 4.3) or in experimental data (Figures 1.3 to 1.5) — they are in a middle ground, as a blend of these extreme cases. Along the axis of mixing degrees, few neurons distinguish the context and locations as clearly as Figure 5.2 **a**. For example, the splitter cells (Figure 4.5) fired differently in different contexts, but they still fired in both contexts. Some neurons (those at the top of Figure 4.3) responded little to both of the stimuli. Along the axis of regularity, the neurons in the RNN were mostly heterogeneous but were visibly stereotypical. They remained silent during a large portion of the trials (thus the high life-time sparsity).

How are these two measures of mixed selectivity related across the 4 extreme representations? By their definitions, ζ ranges from -1 to 1, while κ ranges from 0 to 1. We plot ζ against κ in a scatter plot for these 4 special neural representations [4] in Figure 5.3, along with the tuning curves from trained RNNs. Dots are widely scattered in the plot, which indicates that there is no simple one-to-one mapping between these two measures. Nevertheless, a

[4] Here we consider tuning curves only, without considering the variability around them, which will be treated later.

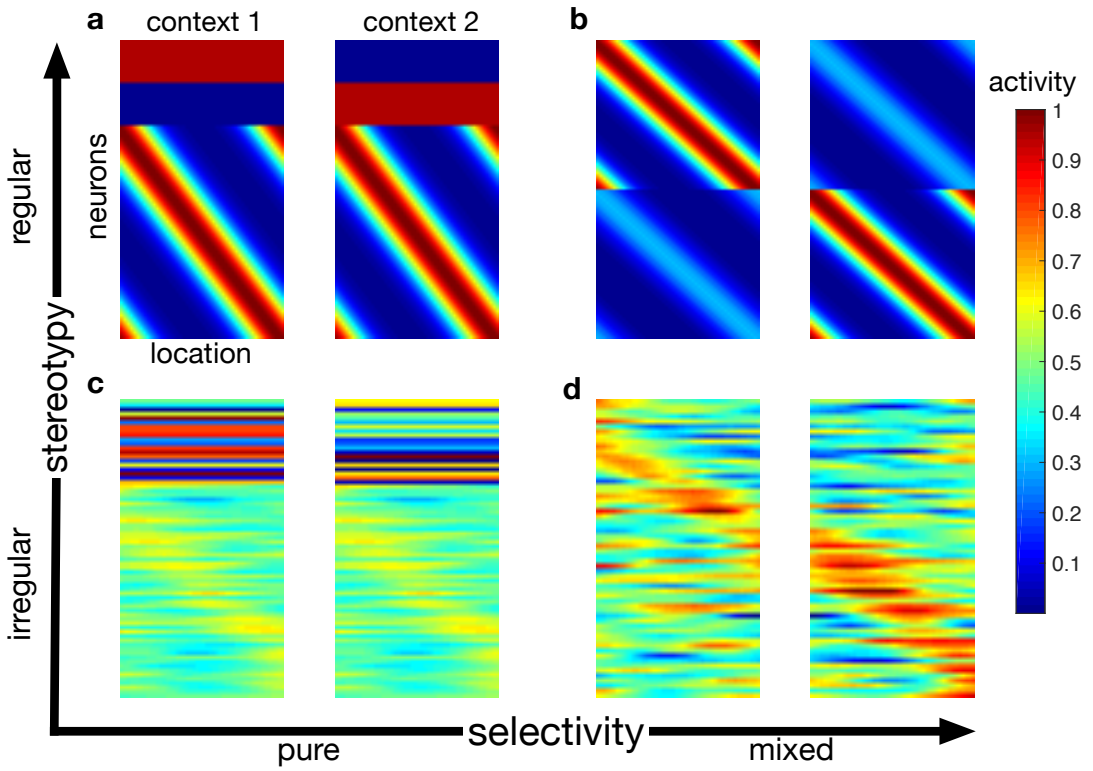


Fig. 5.2 Four illustrative neural representations that are extreme cases in terms of their degrees of mixed selectivity and the regularities of tuning. In these four plots, the left and right panel indicate two different contexts respectively, and the horizontal axis within each panel indicates location. Each row in these panels identifies a neuron, and the colour codes activity. Therefore, the pattern on each row shows the tuning curve of a neuron. Along the axis of regularity, neurons in **a** and **b** have highly regular tunings, while neurons in **c** and **d** have highly irregular tunings. Along the axis of degrees of mixing, neurons in **a** and **c** have pure selectivity, so these neurons can be categorised as "context neurons" or "location neurons". In contrast, neurons in **b** and **d** have highly (nonlinear) mixed selectivity, which are tuned to both context and location.

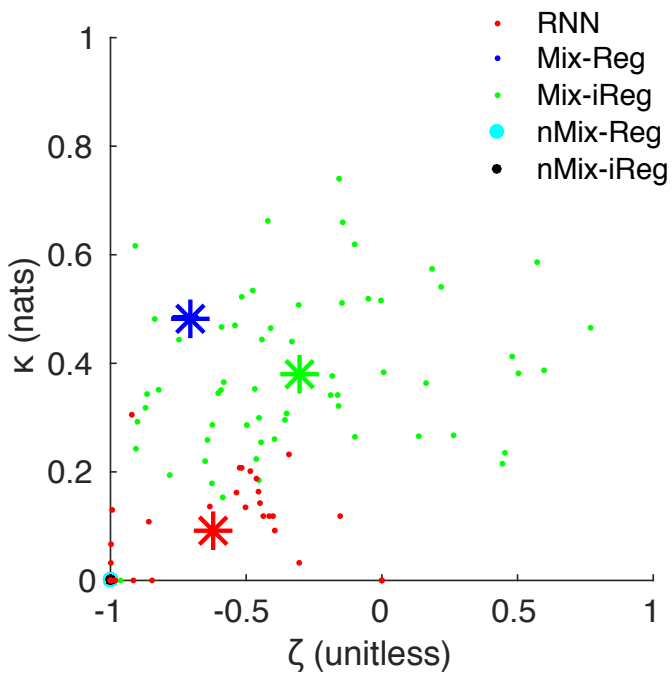


Fig. 5.3 Comparison of mixed selectivity measures. Each dot represents the value of ζ and κ for one neuron. Dots with different colours correspond to different representations: cyan for the pure and regular representation (figure 5.2, **a**); blue for the mixed and regular representation (figure 5.2, **b**); black for the pure and irregular representation (figure 5.2, **c**); green for the mixed and irregular representation (figure 5.2, **d**); finally, red for the representation of the trained RNN. Both measures coincide at their lowest value (lower-left corner), indicating no mixed selectivity, and they are roughly positively correlated. The asterisks illustrate the means of the corresponding colour coded representation. All points within the blue, cyan, and black populations overlaps completely, showing a consensus for the absence of mixed selectivity.

rough positive correlation can be observed from the scatter plot. In particular, the lowest values for both measures (-1 for ζ, 0 for κ) mostly coincide, indicating no mixed selectivity. Overall, dots are more widely distributed along ζ.

5.5 Mixed selectivity under noise

So far, the analysis of the extreme representations is based on tuning curves averaged over multiple trials, ignoring the variance of tuning curves. Next, I empirically compared how noise around tuning curves (i.e. noise variance) affects different mixed selectivity measures. Since the representations with pure selectivity (Figure 5.2, **a** and **c**) always have their mixed selectivity measured at $\kappa = 0$ or $\zeta = -1$, this section focuses only on the representations

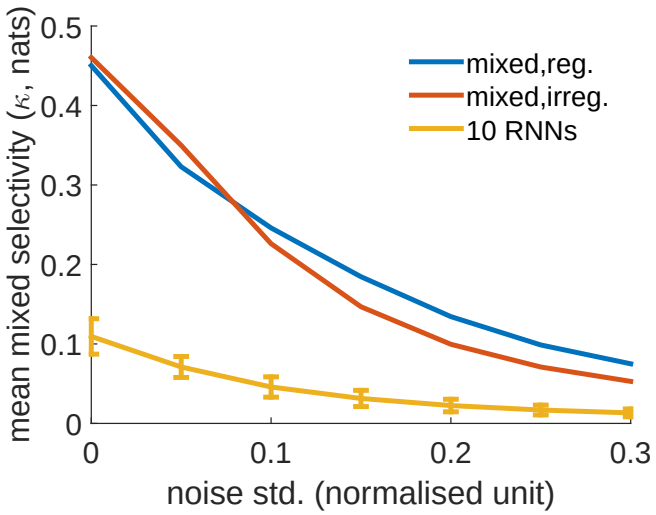


Fig. 5.4 Mixed selectivity measured by κ decreased as the noise variance increased. The blue line shows κ averaged over all neurons with the mixed and regular representation (Figure 5.2, **b**). The red line shows the averaged κ for mixed and irregular representation (Figure 5.2, **d**). The yellow line and error bars show the means and standard deviations of the averaged κ computed from 10 trained RNNs. All the tuning curves were normalised to the range between 0 and 1, and additive noise around the tuning curves were simulated as independent Gaussian noise across different locations, contexts, and trials.

with mixed selectivity (Figure 5.2, **b** and **d**, as well as representations from trained RNNs). As discussed in section 5.2, ζ is defined on tuning curves, which are invariant to the noise considered here. Therefore, this section only presents how κ for representations with mixed selectivity changes with noise.

Noise around tuning curves was simulated by adding independent Gaussian noise at each location and context of the tuning curves that are shown in Figure 5.2 (**b** and **d**); different samples of noise were used for simulating different trials. Although this independence assumption is unrealistic for the noise in RNNs, for a consistent comparison, I also added the same independent Gaussian noise to tuning curves obtained from RNNs (e.g., Figure 4.3). Only neurons that fired on the central arm were considered, since location and context inputs were correlated elsewhere. For example, locations on the upper-left arm always correspond to left-turn trials; *vice versa* for the lower arms. On the other hand, locations on the central arm may correspond to both left-turn and right-turn trials. All the tuning curves were normalised to the range between 0 and 1.

Figure 5.4 shows how κ for the two illustrative representations (blue and red lines) and the representations from 10 trained RNNs (yellow lines) change when the standard deviations of the added Gaussian noise increased from 0 to 0.3. Recall that ζ is computed

using trial-averaged firing, thus is insensitive to the noise across trials. However, the mutual information-based κ, for all the representations, were decreased as the noise level increased. This is consistent with our intuition — mutual information between the sender and receiver generally decreases as noise in the communication channel increases. Although all the mutual information terms in κ decrease with added noise, the joint mutual information is usually more sensitive to noise, due to possibly more delicate covariance structure. As a result, κ generally decreases with noise. It is also clear that the asymptote of infinite noise would lead to $\kappa \to 0$ since there is no mixed selectivity (or selectivity of any kind) when the system is overwhelmed by noise. The differences between the exact effects of noise on different representations need further investigation.

5.6 Mixed selectivity and performance

Despite the differences between these two measures we have shown so far, the significance of mixed selectivity mostly lies in its strong correlation with behavioural performance. Therefore, the final evaluation of ζ and κ should be based on how strongly these measures are correlated with performance. For this purpose, I re-visit the T-maze alternation task, and examine this correlation by analysing: 1. how the two measures of mixed selectivity change after training; 2. in a trained model, how selectively removing units based on their different mixed selectivity measures affect the performance.

In both analyses, I focus on neurons that fired on the central arm after training, which contains less than half of the total neurons (48 out of 100). I focused on the central arm because, as the most challenging part of the task, contextual information that came from mixed selectivity at the decision point was crucial for task performance. I first compute the mixed selectivity of all these neurons before and after training using both ζ and κ. Figure 5.5 illustrates this in both scatter plots and histograms. Despite their different motivation and different value ranges, both ζ and κ consistently reflected the increase of mixed selectivity after training.

To better differentiate the two measures, I then ranked these neurons according to their mixed selectivity assessed by both measures, and evaluated them through the following two "virtual lesion" experiments:

1. We lesion the neurons with the most mixed selectivity, measured by ζ or κ respectively. If these neurons were important, the performance should deteriorate significantly.

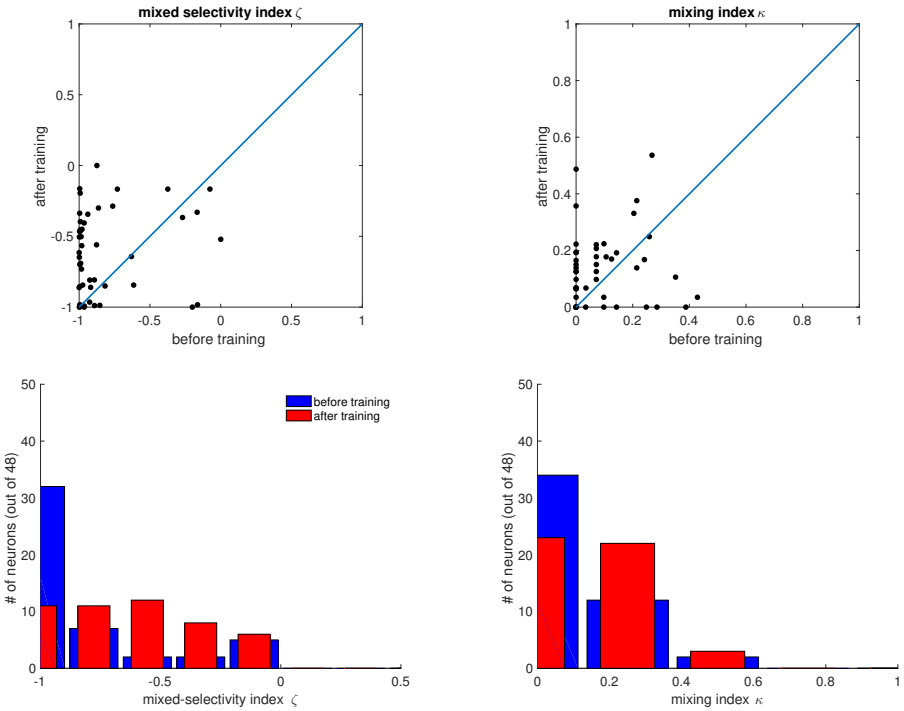


Fig. 5.5 Mixed selectivity measured by both ζ and κ, for neurons fired on the central arm. The scatter plot (top) and histograms shows that overall the mixed selectivity of neurons increased after training in both measures.

2. Conversely, we lesion the neurons with the least mixed selectivity, measured by ζ or κ respectively. If these neurons were unimportant, performance should remain at a similar level.

In each case, the lesion was implemented by setting the read-out weights from the corresponding neurons to 0, and re-training all other output weights. Such lesion might be tested experimentally using optogenetics (Deisseroth, 2011). Since we used linear decoding for the read-out, the optimal output weights were simply obtained analytically from the normal equation. The results of lesioning different fractions of neurons are reported in figure 5.6. Generally, prediction performance deteriorated (RMSE increased) with the increasing fraction of lesioned neurons. However, κ was better correlated with decoding performance than ζ — the red solid line being above the red dash line suggests that neurons with higher κ contributed more to performance. In contrast, ζ seemed to be anti-correlated with decoding performance. In fact, judging from the prediction performance, lesioning neurons with high ζ (blue solid line) could be hardly distinguished from random lesion (black solid line). Therefore, we conclude that the mixing index κ better correlates with performance than the mixed selectivity index ζ.

Conclusion

This chapter proposes a new measure of mixed selectivity, the mixing index κ. κ is a non-negative quantity that uses tools from information theory to quantify the interaction between input features and their neural representation. It is positive *if and only if* the representation depends on multiple input features. In addition to its favourable theoretical properties, experiments showed that κ better correlated with the performance in one-step prediction in terms of RMSE. Of course, since κ is a task-independent quantity, one should not expect it to be the quantity *most* correlated with task performance. To find the neurons contribute most to the prediction task, it is necessary directly measure the correlation between individual neuron's activities and task-dependent variables, such as the RMSE. The next chapter will dive deeper into the components of κ, showing its connection with generative models and denoising training. This leads to the conclusion that κ measures the information from $\mathbf{x}$ being *robustly* represented in $\mathbf{r}$.

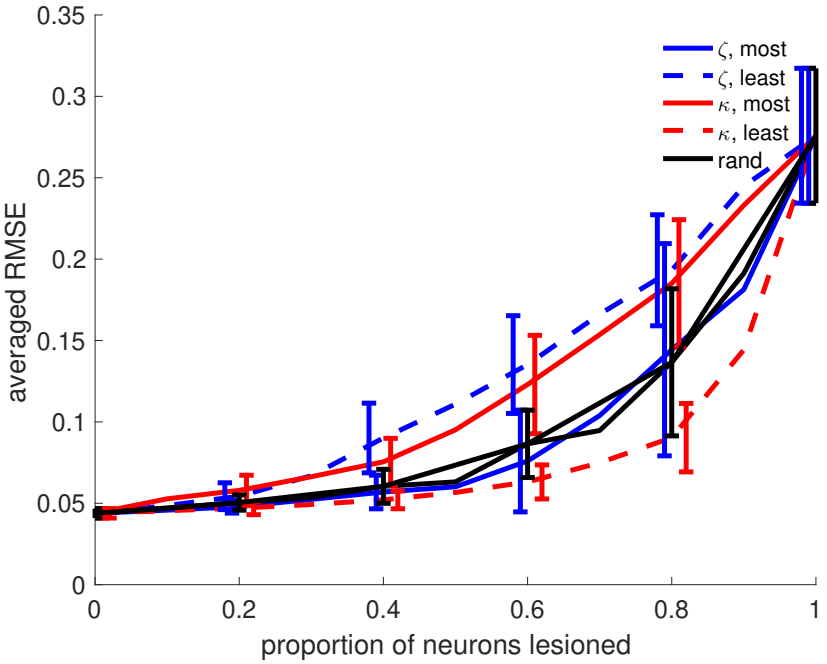


Fig. 5.6 Prediction performance as a function of the fraction of neurons lesioned according to different mixed selectivity measures. Blue and red indicate results obtained using measures ζ and κ respectively. Solid and dashed lines indicate results obtained by removing neurons with the most and least mixed selectivity respectively. Data were obtained from 30 RNNs.

Chapter 6

Information theory

This chapter unifies topics discussed in previous chapters. I will show that generative models, denoising training, and mixed selectivity are closely coupled in training. A study of these couplings suggests that mixed selectivity is a signature of robust representation of input features. To learn such representation requires constraining the representation for robustness, in addition to reconstructing inputs, which parallels the balance of prior and likelihood terms in training generative models. Despite this dual-requirement, a simple and principled way to obtain mixed selectivity is denoising training, as in training a denoising autoencoder.

6.1 Information maximisation

Barber and Agakov (2004) shows that minimising the reconstruction error 2.16 in an autoencoder maximises the mutual information $\mathrm{I}\phi\left[\mathbf{r};\mathbf{x}\right]$ via maximising a variational lower-bound. Here we reproduce their result and prove it with our notation for completeness.

As a reminder from previous chapters, we use $\mathbf{x}$ for external input, $\mathbf{r}$ for neural representation as the latent variable. As in Chapter 2, $p_\theta(\mathbf{x}|\mathbf{r})$ and $q_\phi(\mathbf{r}|\mathbf{x})$ are the parametrised distributions for the generative and inference models respectively. $p^*(\mathbf{x})$ is the data distribution.

Theorem 1. *The expectation of log-likelihood* $\left\langle \ln p_\theta(\mathbf{x}|\mathbf{r})\right\rangle_{q_\phi(\mathbf{r}|\mathbf{x})p^*(\mathbf{x})}$ *is related to the mutual information via*

$$\mathrm{I}_\phi\left[\mathbf{r};\mathbf{x}\right] \geqslant \mathrm{H}\left[\mathbf{x}\right] + \left\langle \ln p_\theta(\mathbf{x}|\mathbf{r})\right\rangle_{q_\phi(\mathbf{r}|\mathbf{x})p^*(\mathbf{x})} \tag{6.1}$$

so that maximising this term, or equivalently minimising the negative reconstruction error (equation 2.16) maximises the mutual information $\mathrm{I}_\phi\left[\mathbf{r};\mathbf{x}\right]$.

Proof. Note that the mutual information $\mathrm{I}_\phi\left[\mathbf{r};\mathbf{x}\right]$ is a property of the recognition model (parametrised by ϕ, or encoder), rather than the generative model (parametrised by θ, the

decoder). The mutual information can be expressed as

$$\begin{aligned} \mathrm{I}_{\phi}\left[\mathbf{r};\mathbf{x}\right] &= \mathrm{H}\left[\mathbf{x}\right] - \mathrm{H}_{\phi}\left[\mathbf{x}|\mathbf{r}\right] \\ &= \mathrm{H}\left[\mathbf{x}\right] + \left\langle \ln q_{\phi}(\mathbf{x}|\mathbf{r}) \right\rangle_{q_{\phi}(\mathbf{r}|\mathbf{x})p^{*}(\mathbf{x})} \\ &\geqslant \mathrm{H}\left[\mathbf{x}\right] + \left\langle \ln p_{\theta}(\mathbf{x}|\mathbf{r}) \right\rangle_{q_{\phi}(\mathbf{r}|\mathbf{x})p^{*}(\mathbf{x})} \end{aligned} \tag{6.2}$$

the joint distribution $q_{\phi}\left(\mathbf{r},\mathbf{x}\right)$ is factorised of as in equation 2.11. $q_{\phi}(\mathbf{x}|\mathbf{r})$ is the likelihood of the recognition model. The difference between the last two lines is a KL-divergence describing the discrepancy between the intractable likelihood of the recognition model $q_{\phi}(\mathbf{x}|\mathbf{r})$ and the generative distribution

$$\mathrm{I}_{\phi}\left[\mathbf{r};\mathbf{x}\right] - \mathrm{H}\left[\mathbf{x}\right] - \left\langle \ln p_{\theta}(\mathbf{x}|\mathbf{r}) \right\rangle_{q_{\phi}(\mathbf{r}|\mathbf{x})p^{*}(\mathbf{x})} = \left\langle \mathrm{D}_{\mathrm{KL}}\left[q_{\phi}(\mathbf{x}|\mathbf{r}) \| p_{\theta}(\mathbf{x}|\mathbf{r})\right] \right\rangle_{p^{*}(\mathbf{x})} \tag{6.3}$$

which shall decrease as the recognition model better matches the generative model. □

This theorem shows that maximising $\left\langle \ln p_{\theta}(\mathbf{x}|\mathbf{r}) \right\rangle_{q_{\phi}(\mathbf{r}|\mathbf{x})p^{*}(\mathbf{x})}$, the first term in the variational lower-bound $\mathcal{L}$ (equation 2.8) maximises a lower-bound of the joint mutual information $\mathrm{I}_{\phi}\left[\mathbf{r};\mathbf{x}\right]$. This mutual information is the first term in the definition of the mixing index $\kappa\left(\mathbf{r},\mathbf{x}\right)$. The other term $\sum_{n=1}^{N}\mathrm{I}_{\phi}\left[\mathbf{r};x_{n}\right]$, as I will show later, comes from the robustness. The Intuitively, training an autoencoder only requires the representation $\mathbf{r}$ to be sufficiently informative to represent data $\mathbf{x}$. Exactly how the information of $\mathbf{x}$ is represented, robust or not, is less a concern. It is training a *robust* autoencoder that demands mixed-selectivity — a way to represent faithfully even with the presentation of significant input or perceptual noise.

6.2 Training robust autoencoders

As a reminder, constraining the representation from an autoencoder, in addition to reconstructing input $\mathbf{x}$, can be motivated from two different perspectives:

1. From a generative model view, the KL-divergence $\mathrm{D}_{\mathrm{KL}}\left[p_{\theta}(\mathbf{x}|\mathbf{r}) \| p_{\theta}\left(\mathbf{r}\right)\right]$ in a part of the the variational lower bound $\mathcal{L}$ (equation 2.8), which is required for maximising the likelihood of the model.

2. From the perspective of generalisation, we require the representation to be invariant to noise from the inputs, for which purely empirical error minimisation (of the reconstruction error) is not sufficient.

We have explored the connection between denoising training and generative models in section 2.6.3 and section 3.3. This section instead focuses on mixed-selectivity. Together,

these results suggests that properly constraining the representation $\mathbf{r}$ also encourages high mixed-selectivity.

At a high level, why does mixed selectivity support robust performance? Generally speaking, data are structured but noise are not (or at least less) structured. The structure of data is usually captured in the joint distribution across multiple dimensions, such as the long range dependencies found in natural images (Field, 1994). Such statistical structure is represented in $\mathbf{r}$ via maximising the mutual information $\mathrm{I}_\phi[\mathbf{r};\mathbf{x}]$ by minimising reconstruction error (Theorem 1).

In contrast, perceptual noise is usually local, which may come from individual sensors or transmitters. Therefore, an intuitive direction to learn robust representations is to make such representations depend more on the joint density $p^*(\mathbf{x})$, but less on the marginal densities $\{p^*(x_n)\}_{n=1}^N$. To formalise this intuition in information theory, one seeks to increase the joint mutual information $\mathrm{I}_\phi[\mathbf{r};\mathbf{x}]$ while decrease the marginal mutual information $\sum_{n=1}^N \mathrm{I}_\phi[\mathbf{r};x_n]$. These two objectives exactly corresponds to the two terms in our definition of the mixing index (equation 5.14). Consistent with this intuition, the following theorem 2 shows denoising training minimises an upper-bound of $\sum_{n=1}^N \mathrm{I}_\phi[\mathbf{r};x_n]$.

Theorem 2. *Under the assumption that an efficient decoder of* $\mathbf{x}$ *from* $\mathbf{r}$ *is available, the contractive regulariser* $\left\|\frac{\partial \mathbf{r}}{\partial \mathbf{x}}\right\|_F^2$ *(as in equation 2.42 and equation 3.15) approximates an upper bound of the mutual information* $\sum_{n=1}^N \mathrm{I}_\phi[\mathbf{r};x_n]$.

Proof. We relax the delta distribution $q_\phi(\mathbf{r}|\mathbf{x})$ (from the deterministic recognition model) using a Gaussian with very small variance σ_r^2 and mean $\mathbf{r}(\mathbf{x}) = \mathbf{g}_\phi(\mathbf{x})$:

$$q_\phi(\hat{\mathbf{r}}|\mathbf{x}) = c \cdot \exp\left(-\frac{\|\hat{\mathbf{r}} - \mathbf{r}(\mathbf{x})\|^2}{2\sigma_r^2}\right) \tag{6.4}$$

where I use $\hat{\mathbf{r}}$ as the random variable to distinguish it from the mean of this random variable $\mathbf{r}$. c is a normalising constant. This relaxation is exact in the limit of $\sigma_r \to 0$. I use the short-hand $\boldsymbol{\xi}_k = \hat{\mathbf{r}}_k - \mathbf{r}_k$. In the following derivation, I use the result from Brunel and Nadal (1998) to connect mutual information $\mathrm{I}_\phi[\hat{\mathbf{r}};x_n]$ with the Fisher information

$$\mathcal{J}(x_n) = \left\langle \left(\frac{\mathrm{d}}{\mathrm{d}x_n} \ln p_\phi(\hat{\mathbf{r}}|x_n)\right)^2 \right\rangle_{p(\boldsymbol{\xi})} \tag{6.5}$$

which involves a derivative form that will be useful later. Here I outline the derivation from Brunel and Nadal (1998).

In their original derivation, Brunel and Nadal (1998) takes the assumption that when the neural population $\hat{\mathbf{r}}$ is sufficiently large, an *efficient* estimator of x_n exists and can be approximated by a maximum-likelihood decoder $d(\hat{\mathbf{r}})$. Here we assume that the generative model $\mathbf{f}_\theta(\mathbf{r})$ provides such a decoder. Under the assumption that $\mathbf{f}_\theta(\mathbf{r})$ is an efficient estimator of x_n, the variance of the estimator achieves the Cramér-Rao bound, which is the inverse Fisher information $\frac{1}{\mathcal{J}(x_n)}$. Applying central limit theorem, we can characterise the estimator as a Gaussian distribution

$$p(x_n|\hat{\mathbf{r}}) = \mathcal{N}\left(x_n; x_\mu, \frac{1}{\mathcal{J}(x_n)}\right) \tag{6.6}$$

whose mean x_μ is unimportant here. This Gaussian distribution gives the conditional entropy

$$\mathrm{H}[x_n|\hat{\mathbf{r}}] = \int p(x_n)\frac{1}{2}\ln\left(\frac{2\pi e}{\mathcal{J}(x_n)}\right)\mathrm{d}x_n \tag{6.7}$$

which further gives the mutual information via $\mathrm{I}_\phi[\hat{\mathbf{r}}; x_n] = \mathrm{H}[x_n] - \mathrm{H}[x_n|\hat{\mathbf{r}}]$. Summing-up all the N elements of $\mathbf{x}$, we therefore have

$$\begin{aligned}
\sum_{n=1}^{N}\mathrm{I}_\phi[\mathbf{r}; x_n] &\approx \sum_{n=1}^{N}\mathrm{I}_\phi[\hat{\mathbf{r}}; x_n] \\
&= \sum_{n=1}^{N}\left\{\mathrm{H}[x_n] - \int p(x_n)\frac{1}{2}\ln\left(\frac{2\pi e}{\mathcal{J}(x_n)}\right)\mathrm{d}x_n\right\} \\
&= \sum_{n=1}^{N}\mathrm{H}[x_n] - \frac{N}{2}\ln(2\pi e) + \frac{1}{2}\sum_{n=1}^{N}\int p^*(\mathbf{x})\ln\mathcal{J}(x_n)\,\mathrm{d}\mathbf{x} \\
&\leqslant \sum_{n=1}^{N}\mathrm{H}[x_n] - \frac{N}{2}\ln(2\pi e) + \frac{1}{2}\int p^*(\mathbf{x})\ln\sum_{n=1}^{N}\mathcal{J}(x_n)\,\mathrm{d}\mathbf{x}
\end{aligned} \tag{6.8}$$

The last line comes from Jensen's inequality. Since $\ln(\cdot)$ is a monotonic function, minimising this upper bound of $\sum_{n=1}^{N}\mathcal{J}(x_n)$ minimises $\sum_{n=1}^{N}\mathrm{I}_\phi[\mathbf{r}; x_n]$.

Using the Gaussian assumption (equation 6.4), the Fisher information becomes

$$\begin{aligned}
\mathcal{J}(x_n) &= \left\langle\left(\frac{\mathrm{d}}{\mathrm{d}x_n}\ln p_\phi(\hat{\mathbf{r}}|x_n)\right)^2\right\rangle_{p(\boldsymbol{\xi})} \\
&= \left\langle\left(\sum_{k=1}^{K}\frac{\xi_k}{\sigma_r}\cdot\frac{\partial r_k}{\partial x_n}\right)^2\right\rangle_{p(\boldsymbol{\xi})} \\
&= \sum_{k=1}^{K}\left(\frac{\partial r_k}{\partial x_n}\right)^2
\end{aligned} \tag{6.9}$$

Change	Generative models	Autoencoders	Information theory
$\Uparrow$	$\mathcal{L}$	$-\left\|\tilde{\mathbf{x}} - f_\theta(g_\phi(\tilde{\mathbf{x}}))\right\|^2$	$\kappa(\mathbf{r},\mathbf{x})$
$\Uparrow$	$\langle \ln p_\theta(\mathbf{x}\|\mathbf{r})\rangle_{q_\phi(\mathbf{r}\|\mathbf{x})}$	$-\left\|\mathbf{x} - f_\theta(g_\phi(\mathbf{x}))\right\|^2$	$\mathrm{I}_\phi[\mathbf{r};\mathbf{x}]$
$\Downarrow$	$\mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{r}\|\mathbf{x})\|p_\theta(\mathbf{r})\right]$	$\mathcal{R}_M$ or $\mathcal{T}$	$\sum_{n=1}^N \mathrm{I}_\phi[\mathbf{r};x_n]$

Table 6.1 The effects of training robust autoencoders. "Change" shows the expected direction of change after training. $\tilde{\mathbf{x}}$ are noise-corrupted data $\mathbf{x}$. Notice that using $\tilde{\mathbf{x}}$ and $\mathbf{x}$ as targets are equivalent when the added noise has zero mean. Within each column, the item in the first row can be decomposed as the item in the second row minus the item in the third row. This decomposition is exact for generative models and for the information theoretic quantities, and is an approximation for autoencoders. The autoencoder loss is used in training and Figure 5.5 confirms the increase of κ after training.

	before training	after training
reconstruction loss	479.043	13.961
regularisation loss	1.947	0.519

Table 6.2 Change of losses after training.

where we used the expectations $\langle \boldsymbol{\xi}_k \rangle_{p(\boldsymbol{\xi})} = \mathbf{0}$ and $\left\langle \boldsymbol{\xi}_k^2 \right\rangle_{p(\boldsymbol{\xi})} = \sigma_r^2 \mathbf{I}$. Again, from equation 6.9 I recovered exactly the contractive regulariser of Rifai et al. (2011) (see also section 3.3). □

From section 2.6.3 and section 3.3, we know that denoising training at the same time minimising the KL-divergence $\mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{r}|\mathbf{x})\|p_\theta(\mathbf{r})\right]$ between the approximate posterior and a Gaussian mixture model prior $p_\theta(\mathbf{r})$. Note that that simply optimising the KL-divergence term $\mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{r}|\mathbf{x})\|p_\theta(\mathbf{r})\right]$ alone actually minimises an upper bound of $\mathrm{I}_\phi[\mathbf{r};\mathbf{x}]$, thus having the opposite effect in information maximisation. This is because minimising this KL-divergence moves $\mathbf{r}$ closer to the prior, which has no information about the input $\mathbf{x}$. This is shown more formally in Appendix B. Therefore, denoising training provides a robustness constraint in minimising an upper bound of $\sum_{n=1}^N \mathrm{I}_\phi[\mathbf{r};x_n]$, *in addition to* approximating training with the constrained GMM prior, which minimises an upper bound of $\mathrm{I}_\phi[\mathbf{r};\mathbf{x}]$.

Overall, denoising training provides a computationally simple, yet theoretically grounded way to train autoencoders as generative models. In the end, we summarise the results from different perspectives in table 6.1. Similar results can be extended to an RNN by unrolling the RNN and compute the corresponding $\mathbf{x}_t$ and $\mathbf{r}_t$ in the order specified by their time indexes. See Appendix A for more detailed derivations.

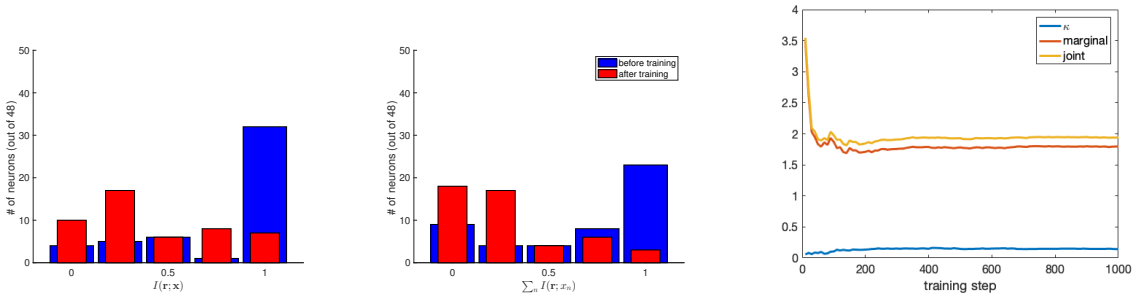


Fig. 6.1 Mixing index decomposed as the joint mutual information (left) and the sum of marginal mutual information (middle). The right panel shows these quantities together with κ (averaged across neurons) across training steps. Both mutual information quantities decreased after training. However, their difference, which defines the mixing index κ, increased as suggested by the theory (figure 5.5 and the right panel here).

6.3 Experiments

Recall that figure 5.5 (right) has shown the change of κ after training. To further illustrate the components in κ during and after training, the changes of terms in the "Information theory" column are plotted as histograms in figure 6.1, together with the all these components averaged over neurons during training. It might be surprise at first that the mutual information (both $\mathrm{I}[\mathbf{r};\mathbf{x}]$ and $\sum_{n=1}^{N}\mathrm{I}[\mathbf{r};x_n]$) actually decreased after training, since one may expect training to increase this mutual information. However, recall we discussed in the previous section that denoising training, as well as minimising the KL-divergence $\mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{r}|\mathbf{x})\|p_\theta(\mathbf{r})\right]$, may reduce the mutual information (see also Appendix B). Intuitively, a large amount of information in $\mathbf{x}$, which were sequences of locations (including the context distinguishing left-turn or right-turn trials) in the simulation, are actually not useful for the task of one-step prediction. On the other hand, denoising training requires the neurons to only robustly represent the location and context useful for predicting the *next* location. As a compromise of these two objectives, we observed overall decreased mutual information, for losing the less useful information, but increase in κ, for retaining the robustly represented information essential for the task.

In addition, table 6.2 shows the changes of reconstruction error and regularisation loss from the same RNN, corresponding to the (negative of the) second row and third row under the "Autoencoders" column. Notably, the regularisation loss ($\mathcal{T}$) decreased even when this model was obtained from denoising training, without explicitly using the regulariser. These empirical results thus verified our theory connecting mixed selectivity with mutual information, generative model, and denoising training. Similar results for RNNs trained

directly using the denoising regulariser (section 3.3) are shown in figure C.2 and C.3 and table C.2.

Conclusion

This chapter uses information theory to reveal the deep connections between mixed selectivity, generative models, and denoising training. Denoising training efficiently provides a robustness constraint on the representation, in addition to approximating training with a GMM prior in supporting training a generative model. While the derivation in this chapter relies on different bounds, the empirical results verified that denoising training indeed led to increased mixed selectivity of neural representation. In this process, the joint mutual information $\mathrm{I}_{\phi}\left[\mathbf{r};\mathbf{x}\right]$ was *maximised* to preserve information about the structure (joint distribution) of data, while the marginal mutual information $\sum_{n=1}^{N}\mathrm{I}_{\phi}\left[\mathbf{r};x_n\right]$ was *minimised* for the robustness of the model against unstructured noise at individual stimuli.

Chapter 7

Discussion

This thesis establishes a concrete theoretical link between the widely observed neural representations with mixed selectivity and denoising training of neural networks. This connection gives such neural representations a statistical explanation. The generative model perspective and denoising training approach are unified under information theory to formally justify mixed selectivity as a signature of robust representations. In support of recent experiments and analyses showing a strong correlation between mixed selectivity and performance (Rigotti et al., 2013; Saez et al., 2015), the results in this thesis explain this correlation as a manifestation of robust encoding of input features implemented by neural networks.

7.1 Mixed selectivity is ubiquitous

Neurons with mixed selectivity can be found in different areas, exhibiting various forms (Baeg et al., 2003; Ginther et al., 2011; Harvey et al., 2012; Hubel and Wiesel, 1962; Rigotti et al., 2013). These include the context dependent hippocampal neurons (Wood et al., 2000), which we analysed in detail. Despite their vastly different tuning curves, these neural responses exhibited non-linear combinations of multiple stimuli that were relevant to task performance. Since their unique tuning curves were visibly regular, these cells are relatively easy to identify among the large number of cortical neurons that often present more chaotic activities (Sompolinsky et al., 1988). Moreover, some of these mixed selectivity neurons are regular and stereotypical enough that they were bestowed with names such as “place cells” and “splitter cells”.

Neurons with mixed selectivity are also found in computational models, either as an emergent property or by construction. Following the convention of the pioneering connectionists, I call the characteristic neural responses are termed “neural representations”, and the input patterns to which such neurons respond most strongly are called “features” (Bengio et al.,

2012; Hinton, 2014b; Rumelhart et al., 1987, 1986). Such input patterns are in the form of combinations of inputs of lower level features. The presentation of such combined features can be interpreted as mixed selectivity. Therefore, in most neural network models based on learning, including the RNNs presented in this thesis, mixed selectivity is an emergent property arising spontaneously from training to solve certain tasks (Bengio et al., 2012; Hinton, 1984; Olshausen and Field, 1996; Rumelhart et al., 1987; Sussillo, 2014).

In parallel to the recently discovered strong correlation between mixed selectivity and performance, it is well known in the machine learning community that the emergence of "interesting" representations is correlated with good performance. Feature learning or representation learning have being increasingly popularity in machine learning, because learning good features has been regarded as an intermediate step for solving other challenging tasks (Bengio et al., 2012; Hinton, 2014b). Higher level features, corresponding to neurons with more complex mixed selectivity, are hypothesised to capture more abstract aspects of data, which can be used for high level cognitive tasks for both the brain and computers. In support of this hypothesis, neurons tuned to increasingly abstract features have been found in both hierarchical neural network models and the hierarchically organised cortical visual stream (Yamins and Dicarlo, 2016; Yamins et al., 2014).

Alternatively, instead of relying on learning, mixed selectivity may be explicitly incorporated into models. This approach is not explored in this thesis. It is perhaps better known in cognitive science and computational linguistics as "variable binding" (Anderson, 2013; Eliasmith and Anderson, 2004; Smolensky, 1990), by which multiple variables, possibly representing different stimuli, are explicitly combined via operations such as inner-product, convolution or more complicated nonlinear mappings. Such manually created features are especially helpful when they are known to benefit tasks *a priori*, and have been used as input into some of the state-of-the-art neural network models (Eliasmith et al., 2012; Silver et al., 2014).

7.2 Why mixed selectivity is useful

The computational benefit of mixed selectivity is explained using the expansion of dimensionality by Rigotti et al. (2013). Nonlinear mixed selectivity has the same effect as projecting input into a high dimensional *feature space*, where nonlinear combinations of input can be read-out from a linear decoder, in a way similar to the kernel methods that have been popularised by support vector machines (Burges, 1998) and Gaussian Processes (Rasmussen and Williams, 2006). As has been explored in the area of random features (Rahimi and Recht, 2008), random projections combined with nonlinear neurons is able to generate nonlinear

mixed selectivity and expanded dimensionality (Barak et al., 2013; Fusi et al., 2016). However, dimensionality expansion needs to trade-off generalisation, especially in the presence of noise. Extra dimensions may be introduced as an artefact of noise, when the true signal usually lies in a low dimensional sub-space or manifold. For example, principal component analysis (PCA, see also section 2.5.2) finds the subspace that preserves the most variance of data (Hotelling, 1933). In this case, dimensions with small variances, which may be a result of noise, can be discarded. In more complex settings, nonlinear dimensionality reduction techniques based on non-Euclidean metrics can be used to learn low-dimensional manifold (Roweis and Saul, 2008; Seung, 1997; Tenenbaum et al., 2000; Vincent et al., 2008b). To sum up, the trade-off is that the dimensionality needs to be high enough to allow linear read-out of specific input configuration, and low enough to allow generalisation over similar inputs.

In section 4.4.1, we explained the benefit of mixed selectivity from its contributions to the dynamics of the RNN. In the sequence learning example (section 4.1), the neural representation with mixed selectivity emerged as a solution for the T-maze alternation task which requires both denoising and sequence disambiguation. The biologically plausible representation in our model emerged from optimising the RNN for task performance, without any objectives regarding the representation itself. This is different from recent work of Rajan et al. (2016), which trained an RNN explicitly to fit the experimentally recorded neural representations. The analysis in section 4.4.2 further showed mixed selectivity neurons ("splitter cells") contributed to the formation of (quasi-)line attractors, in which the overlapping parts of sequences were actually separated in the RNN's state-space. We thus provided a normative explanation of context-dependent neural representations observed in various cortical areas, which support optimal performance for sequence learning.

A large part of this thesis has been dedicated to justify mixed selectivity from a statistical perspective, which applies to a wider range of problems compared with those amenable to dynamical systems analysis. Viewing a neural system as a probabilistic generative model, its neural representation functions as the latent variable which describes unobserved factors that affect the observed data. Neural representations, when regarded as latent variables, reflect statistical assumptions about the data. These are ultimately constrained by the neural system's biological properties, which may be experimentally testable. Importantly, mixed selectivity can be investigated using an information-theoretic measure (section 5.3). This measure is consistent with the idea of context-based coding (Rigotti et al., 2013). Moreover, this measure enabled us to establish a theoretical link between neural generative models and mixed selectivity based on mutual information between stimuli and neural representations. In this front, the main contributions of this thesis are:

1. Deriving the implicit GMM priors approximated by denoising training (section 2.6)

2. showing that a lower bound for mixed selectivity (measured by the mixing index) is maximised in denoising training (section 6.2).

7.3 Regularisers for training neural networks

Another contribution of this thesis is in developing the denoising regulariser that achieved nearly the same performance in our sequence learning task as denoising training, without using any noise (section 3.3, see also the results from regularised training in Appendix C). It has the potential as an efficient and low-variance alternative of the popular denoising training. However, at this moment, the advantage of this regulariser in more general setting is unclear, given the limited empirical evaluation in this thesis. Overall, regularisers play an important role in this thesis. They bridge the gaps between mixed selectivity, generative models, and the hitherto largely heuristic denoising training.

More generally, regularisers provides a pathway for understanding existing heuristics. Training complex parametric models such as neural networks usually requires various tricks. As seen in this thesis as well as other related work, regularisers can help us understand *ad-hoc* tricks by providing a potential link to more principled methods. Regularisers bridge them by formalising or approximating algorithmically described heuristics as a mathematical expression of a cost term. Such a distilled form is more amenable to further theoretical interpretation. Several existing heuristics, such as "drop-out" Hinton (2014a), have been better understood (and often consequently improved) in this way. As new heuristics and tricks almost always come with novel and innovative architectures such as Generative Adversarial Nets (Goodfellow et al., 2014), novel regularisers are always likely to be useful.

References

Agster, K. L., Fortin, N. J., and Eichenbaum, H. (2002). The hippocampus and disambiguation of overlapping sequences. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 22(13):5760–8.

Aldous, D. J. (1985). *Exchangeability and related topics*. Springer.

Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K. (2016). Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*.

Allen-Zhu, Z., Li, Y., and Liang, Y. (2018). Learning and generalization in overparameterized neural networks, going beyond two layers. *arXiv preprint arXiv:1811.04918*.

Anderson, J. R. (2013). *The architecture of cognition*. Psychology Press.

Arora, S., Du, S. S., Hu, W., Li, Z., and Wang, R. (2019). Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. *arXiv preprint arXiv:1901.08584*.

Averbeck, B. B., Latham, P. E., and Pouget, A. (2006). Neural correlations, population coding and computation. *Nature reviews. Neuroscience*, 7(5):358–366.

Baeg, E. H., Kim, Y. B., Huh, K., Mook-Jung, I., Kim, H. T., and Jung, M. W. (2003). Dynamics of population code for working memory in the prefrontal cortex. *Neuron*, 40:177–188.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. *International Conference on Learning Representations*, pages 1–15.

Baldi, P. and Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58.

Banino, A., Barry, C., Uria, B., Blundell, C., Lillicrap, T., Mirowski, P., Pritzel, A., Chadwick, M. J., Degris, T., Modayil, J., et al. (2018). Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557(7705):429.

Barak, O., Rigotti, M., and Fusi, S. (2013). The Sparseness of Mixed Selectivity Neurons Controls the Generalization-Discrimination Trade-Off. *Journal of Neuroscience*, 33(9):3844–3856.

Barber, D. and Agakov, F. (2004). The IM Algorithm: A Variational Approach to Information Maximization. *Advances in Neural Information Processing Systems 16*, (2).

Barlow, H. (1989). Unsupervised Learning. *Neural Computation*, 1(3):295–311.

Bayer, J. and Osendorfer, C. (2014). Learning Stochastic Recurrent Networks. pages 1–9.

Bell, a. J. and Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 7(6):1129–1159.

Bell, A. J. and Sejnowski, T. J. (1997). The 'independent components' of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338.

Bengio, Y. (2009). *Learning Deep Architectures for AI*, volume 2.

Bengio, Y., Courville, A., and Vincent, P. (2012). Representation Learning: A Review and New Perspectives. (1993):1–34.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy Layer-Wise Training of Deep Networks. *Advances in neural information processing systems*, 19(1):153.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–66.

Bengio, Y., Yao, L., Alain, G., and Vincent, P. (2013). Generalized Denoising Auto-Encoders as Generative Models. *Advances in Neural Information Processing Systems (NIPS)*, pages 1–9.

Bergstra, J. and Bengio, Y. (2009). Slow, Decorrelated Features for Pretraining Complex Cell-like Networks. *NIPS*, pages 1–9.

Berkes, P., Orbán, G., Lengyel, M., and Fiser, J. (2011). Spontaneous Cortical Activity Reveals Hallmarks of an Optimal Internal Model of the Environment. *Science*, 331(January):83–88.

Bishop, C. M. (1995). Training with Noise is Equivalent to Tikhonov Regularization. *Neural Computation*, 7(1):108–116.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*, volume 4 of *Information science and statistics*. Springer.

Blundell, C., Uria, B., Pritzel, A., Li, Y., Ruderman, A., Leibo, J. Z., Rae, J., Wierstra, D., and Hassabis, D. (2016). Model-free episodic control. *arXiv preprint arXiv:1606.04460*.

Bourlard, H. and Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59(4-5):291–294.

Brown, T. I., Ross, R. S., Keller, J. B., Hasselmo, M. E., and Stern, C. E. (2010). Which way was I going? Contextual retrieval supports the disambiguation of well learned overlapping navigational routes. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 30(21):7414–22.

Brunel, N. and Nadal, J.-P. (1998). Mutual Information, Fisher Information, and Population Coding. *Neural Computation*.

Burges, C. J. C. (1998). A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167.

Bush, D., Philippides, A., Husbands, P., and O'Shea, M. (2010). Dual coding with STDP in a spiking recurrent neural network model of the hippocampus. *PLoS computational biology*, 6(7):e1000839.

Cadena, S. A., Denfield, G. H., Walker, E. Y., Gatys, L. A., Tolias, A. S., Bethge, M., and Ecker, A. S. (2019). Deep convolutional models improve predictions of macaque v1 responses to natural images. *PLoS computational biology*, 15(4):e1006897.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv*, pages 1–9.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A., and Bengio, Y. (2015). A Recurrent Latent Variable Model for Sequential Data. In *Advances in neural information processing systems (NIPS)*, page 8.

Dauphin, Y., Pascanu, R., and Gulcehre, C. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in Neural . . .*, pages 1–9.

Dayan, P. and Abbott, L. (2001). Theoretical neuroscience: Computational and mathematical modeling of neural systems.

Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. (1995). The Helmholtz machine. *Neural computation*, 7(5):889–904.

Deisseroth, K. (2011). Optogenetics. *Nature methods*, 8(1):26–29.

Dempster, A. P., Laird, N., and D.B. Rubin (1977). *Maximum Likelihood from Incomplete Data via the EM Algorithm*, volume 39.

Dinh, L., Krueger, D., and Bengio, Y. (2014). Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using Real NVP. *arXiv preprint arXiv:1605.08803*.

Eichenbaum, H. and Cohen, N. J. (2001). *From Conditioning to Conscious Recollection: Memory Systems of the Brain*, volume 4.

Eliasmith, C. and Anderson, C. H. (2004). *Neural Engineering*. The MIT Press.

Eliasmith, C., Stewart, T. C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., Tang, C., and Rasmussen, D. (2012). A large-scale model of the functioning brain. *Science (New York, N.Y.)*, 338(6111):1202–5.

Elman, J. (1990). Finding Structure in Time. *Cognitive Science*, 211:179–211.

Euston, D. R., Tatsuno, M., and McNaughton, B. L. (2007). Fast-forward playback of recent memory sequences in prefrontal cortex during sleep. *Science (New York, N.Y.)*, 318(5853):1147–50.

Faisal, A. A., Selen, L. P. J., and Wolpert, D. M. (2008). Noise in the nervous system. *Nature Reviews Neuroscience*, 9(april):292–303.

Festa, D., Hennequin, G., and Lengyel, M. (2014). Analog Memories in a Balanced Rate-Based Network of E-I Neurons. *Adv. Neural Inf. Process. Syst. 27*, pages 1–9.

Field, D. J. (1994). What Is the Goal of Sensory Coding? *Neural Computation*, 6(4):559–601.

Fiser, J., Berkes, P., Orbán, G., and Lengyel, M. (2010). Statistically optimal perception and learning: from behavior to neural representations. *Trends in cognitive sciences*, 14(3):119–30.

Fujisawa, S., Amarasingham, A., Harrison, M. T., and Buzsáki, G. (2008). Behavior-dependent short-term assembly dynamics in the medial prefrontal cortex. *Nature neuroscience*, 11(7):823–833.

Fusi, S., Miller, E. K., and Rigotti, M. (2016). Why neurons mix: high dimensionality for higher cognition. *Current Opinion in Neurobiology*, 37:66–74.

Fuster, J. M. (2001). The prefrontal cortex-An update-Time is of the essence. *Neuron*, 30(C):319–333.

Gal, Y. and Ghahramani, Z. (2015). Dropout as a Bayesian Approximation : Representing Model Uncertainty in Deep Learning. *Icml*, pages 1–10.

Ganguli, S., Bisley, J. W., Roitman, J. D., Shadlen, M. N., Goldberg, M. E., and Miller, K. D. (2008). One-Dimensional Dynamics of Attention and Decision Making in LIP. *Neuron*, 58:15–25.

Ginther, M. R., Walsh, D. F., and Ramus, S. J. (2011). Hippocampal neurons encode different episodes in an overlapping sequence of odors task. *The Journal of neuroscience*, 31(7):2706–11.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier networks. In *AISTATS*, volume 15, pages 315–323.

Golomb, D., Hertz, J., Panzeri, S., Treves, A., and Richmond, B. (1997). How well can we estimate the information carried in neuronal responses from limited samples? *Neural Comput*, 9(3):649–665.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Graves, A. (2011). Practical variational inference for neural networks. *Advances in Neural Information Processing Systems*, pages 1–9.

Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech Recognition with Deep Recurrent Neural Networks. *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on. IEEE*, (3):6645–6649.

Griffiths, T. L., Chater, N., Kemp, C., Perfors, A., and Tenenbaum, J. B. (2010). Probabilistic models of cognition: exploring representations and inductive biases. *Trends in Cognitive Sciences*, 14(8):357–364.

Harvey, C. D., Coen, P., and Tank, D. W. (2012). Choice-specific sequences in parietal cortex during a virtual-navigation decision task. *Nature*, 484(7392):62–68.

Harvey, C. D., Collman, F., Dombeck, D. a., and Tank, D. W. (2009). Intracellular dynamics of hippocampal place cells during virtual navigation. *Nature*, 461(7266):941–6.

Hasselmo, M. E. and Eichenbaum, H. (2005). Hippocampal mechanisms for the context-dependent retrieval of episodes. *Neural Networks*, 18:1172–1190.

Hennequin, G., Aitchison, L., and Lengyel, M. (2014a). Fast Sampling-Based Inference in Balanced Neuronal Networks. *Advances in Neural Information Processing Systems*, pages 1–9.

Hennequin, G., Vogels, T. P., and Gerstner, W. (2014b). Optimal control of transient dynamics in balanced networks supports generation of complex movements. *Neuron*, 82(6):1394–1406.

Hinton, G. (1984). Distributed representations.

Hinton, G. (1986). Learning distributed representations of concepts. *Proceedings of the eighth annual conference of the ....*

Hinton, G. (2014a). Dropout : A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research (JMLR)*, 15:1929–1958.

Hinton, G. (2014b). Where do features come from? *Cognitive Science*, 38(6):1078–1101.

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. and Zemel, R. S. (1994). Autoencoders, minimum description length and Helmoltz free energy. *Advances in neural information ...*, pages 3–10.

Hinton, G. E. (1989). Learning distributed representations of concepts. In Morris, R. G. M., editor, *Parallel distributed processing: Implications for psychology and neurobiology*, pages 46–61. Clarendon Press, Oxford, England.

Hinton, G. E. (2007). To recognize shapes, first learn to generate images. *Progress in Brain Research*, 165:535–547.

Hinton, G. E. and Ghahramani, Z. (1997). Generative models for discovering sparse distributed representations. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 352(1358):1177–90.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Hinton, G. E. and van Camp, D. (1993). Keeping Neural Networks Simple by Minimizing the Description Length of the Weights. In *Conference on Learning Theory*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–80.

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. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79(8):2554–2558.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417.

Howard, M. W. and Kahana, M. J. (2002). A Distributed Representation of Temporal Context. *Journal of Mathematical Psychology*, 46(3):269–299.

Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154.

Huettel, S. A., Mack, P. B., and McCarthy, G. (2002). Perceiving patterns in random series: dynamic processing of sequence in prefrontal cortex. *Nature neuroscience*, 5(5):485–90.

Hyvärinen, A. (2006). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6:695–708.

Japkowicz, N., Hanson, S. J., and Gluck, M. A. (2000). Nonlinear autoassociation is not equivalent to pca. *Neural computation*, 12(3):531–545.

Jozefowicz, R., Zaremba, W., and Sutskever, I. (2015). An Empirical Exploration of Recurrent Network Architectures. *Proceedings of the ...*, 6(3):171–180.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134.

Káli, S. and Dayan, P. (2000). The involvement of recurrent connections in area CA3 in establishing the properties of place fields: a model. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 20(19):7463–77.

Kandel, E. R., Schwartz, J. H., Jessell, T. M., and Others (2000). *Principles of neural science*, volume 4. McGraw-hill New York.

Karpathy, A., Johnson, J., and Fei-Fei, L. (2015). Visualizing and Understanding Recurrent Networks. *arXiv*, pages 1–13.

Kingma, D. and Welling, M. (2014). Efficient Gradient-Based Inference through Transformations between Bayes Nets and Neural Nets. *Proceedings of the 31th International Conference on Machine Learning (ICML)*, 32.

Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational Dropout and the Local Reparameterization Trick. In *Advances in neural information processing systems (NIPS)*, pages 1–13.

Latham, P. E. (2005). Synergy, Redundancy, and Independence in Population Codes, Revisited. *Journal of Neuroscience*, 25(21):5195–5206.

Laughlin, S. B. (2001). Energy as a constraint on the coding and processing of sensory information. *Current opinion in neurobiology*, pages 475–480.

Le, Q. V., Jaitly, N., and Hinton, G. E. (2015). A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. *arXiv preprint arXiv:1504.00941*, pages 1–9.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Lengyel, M. and Dayan, P. (2005). Rate- and Phase-coded Autoassociative Memory. In *Advances In Neural Information Processing Systems*, number section 3.

Lengyel, M. and Dayan, P. (2007). Uncertainty, phase and oscillatory hippocampal recall. *Advances in Neural Information Processing Systems*, 19:833–840.

Levy, W. B. (1996). A sequence predicting CA3 is a flexible associator that learns and uses context to solve hippocampal-like tasks. *Hippocampus*, 6:579–590.

Lewicki, M. S. (2002). Efficient coding of natural sounds. *Nature neuroscience*, 5(4):356–63.

Lewicki, M. S. and Sejnowski, T. J. (2000). Learning overcomplete representations. *Neural computation*, 12(2):337–365.

Li, Y. and Liang, Y. (2018). Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*, pages 8157–8166.

Li, Y., Swersky, K., and Zemel, R. (2015). Generative moment matching networks. In *International Conference on Machine Learning*, pages 1718–1727.

Maas, A. and Le, Q. (2012). Recurrent Neural Networks for Noise Reduction in Robust ASR. *Interspeech*, pages 3–6.

MacKay, D. J. (1996). Maximum likelihood and covariant algorithms for independent component analysis. Technical report, Citeseer.

MacKay, D. J. C. (1992). A practical Bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472.

MacKay, D. J. C. (2003). *Information theory, inference and learning algorithms*. Cambridge university press.

Mante, V., Sussillo, D., Shenoy, K. V., and Newsome, W. T. (2013). Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503(7474):78–84.

Markus, E. J., Qin, Y. L., Leonard, B., Skaggs, W. E., McNaughton, B. L., and Barnes, C. a. (1995). Interactions between location and task affect the spatial and directional firing of hippocampal neurons. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 15(11):7079–94.

Marr, D. (1971). Simple Memory : a Theory F O R Archicortex. *Philosophical transactions of the Royal Society B*, 262(July):23–81.

Marr, D. (1982). Vision: A computational investigation into the human representation and processing of visual information.

Marr, D. (1991). From the Retina to the Neocortex. *Citeseer*.

Martens, J. (2010). Deep learning via Hessian-free optimization. *27th International Conference on Machine Learning*, 951:735–742.

Martens, J. and Sutskever, I. (2011). Learning Recurrent Neural Networks with Hessian-Free Optimization. In *Proceedings of the 28th International Conference on Machine Learning*.

McGill, W. (1954). Multivariate information transmission. *Transactions of the IRE Professional Group on Information Theory*, 4(4):93–111.

Mikolov, T., Karafiát, M., Burget, L., Cernock\'y, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Interspeech*, volume 2, page 3.

Minsky, M. and Papert, S. (1969). Perceptrons.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. a., Veness, J., Bellemare, M. G., 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.

Mohamed, S. and Lakshminarayanan, B. (2016). Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*.

Monaco, J. D. and Levy, W. B. (2003). T-maze training of a recurrent CA3 model reveals the necessity of novelty-based modulation of LTP in hippocampal region CA3. *Proceedings of the International Joint Conference on Neural Networks*, 3:1655–1660.

Nair, V. and Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. *Proceedings of the 27th International Conference on Machine Learning*, (3):807–814.

Neal, R. M. (1996). *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media.

Olshausen, B. A. and Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*.

Olshausen, B. A. and Field, D. J. (2004). Sparse coding of sensory inputs. *Current opinion in neurobiology*, 14(4):481–7.

Ostojic, S. (2014). Two types of asynchronous activity in networks of excitatory and inhibitory spiking neurons. *Nature neuroscience*, 17(4).

Paninski, L. (2003). Estimation of Entropy and Mutual Information. *Neural Computation*, 15(6):1191–1253.

Pascanu, R., Mikolov, T., and Bengio, Y. (2012). On the difficulty of training recurrent neural networks. *Proceedings of The 30th International Conference on Machine Learning*, pages 1310–1318.

Pastalkova, E., Itskov, V., Amarasingham, A., and Buzsáki, G. (2008). Internally generated cell assembly sequences in the rat hippocampus. *Science (New York, N.Y.)*, 321(5894):1322–7.

Pearlmutter, B. A. (1994). Fast exact multiplication by the Hessian. *Neural Computation*, 6(1):147–160.

Pfeiffer, B. E. and Foster, D. J. (2013). Hippocampal place-cell sequences depict future paths to remembered goals. *Nature*, 497(7447):74–81.

Pfister, J.-P. P. J. J.-P., Dayan, P., and Lengyel, M. (2010). Synapses with short-term plasticity are optimal estimators of presynaptic membrane potentials. *Nature Neuroscience*, 13(i):1271–5.

Priebe, N. J. and Ferster, D. (2005). Direction selectivity of excitation and inhibition in simple cells of the cat primary visual cortex. *Neuron*, 45(1):133–145.

Rahimi, A. and Recht, B. (2008). Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184.

Rajan, K., Harvey, C., and Tank, D. (2016). Recurrent Network Models of Sequence Generation and Memory. *Neuron*, pages 1–15.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. the MIT Press.

Rezende, D. J. and Mohamed, S. (2015). Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014a). Stochastic backpropagation and approximate inference in deep generative models. *Proceedings of The 31st ...*, 32:1278–1286.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014b). Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*.

Rifai, S., Bengio, Y., Dauphin, Y., and Vincent, P. (2012). A generative process for sampling contractive auto-encoders. *arXiv preprint arXiv:1206.6434*, (1):1855–1862.

Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. (2011). Contractive Auto-Encoders : Explicit Invariance During Feature Extraction. *Icml*, 85(1):833–840.

Rigotti, M., Barak, O., Warden, M. R., Wang, X.-J., Daw, N. D., Miller, E. K., and Fusi, S. (2013). The importance of mixed selectivity in complex cognitive tasks. *Nature*, 497(7451):585–90.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14:465–471.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*.

Rolls, E. T. and Tovee, M. J. (1995). Sparseness of the neuronal representation of stimuli in the primate temporal visual cortex. *Journal of Neurophysiology*, 73(2):713–726.

Roweis, S. (1998). EM algorithms for PCA and SPCA. In *Advances in neural information processing systems (NIPS)*, pages 626–632. MORGAN KAUFMANN PUBLISHERS.

Roweis, S. and Ghahramani, Z. (1999). A Unifying Review of Linear Gaussian Models. *Neural Computation*, 345(1995):305–345.

Roweis, S. T. and Saul, L. K. (2008). Nonlinear Dimensionality Reduction by Locally Linear Embedding. 290(5500):2323–2326.

Rumelhart, D. E., Hinton, G. E., and McClelland, J. L. (1987). *Parallel distributed processing*, volume 2. IEEE.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Saez, A., Rigotti, M., Ostojic, S., Fusi, S., and Salzman, C. D. (2015). Abstract Context Representations in Primate Amygdala and Prefrontal Cortex. *Neuron*, 87(4):869–881.

Savin, C., Dayan, P., and Lengyel, M. (2013). Correlations strike back (again): the case of associative memory retrieval. *Advances in Neural Information . . .* , pages 1–9.

Schmidhuber, J. (2015). Deep Learning in neural networks: An overview. *Neural Networks*, 61:85–117.

Schraudolph, N. N. (2002). Fast curvature matrix-vector products for second-order gradient descent. *Neural computation*, 14(7):1723–38.

Seung, H. S. (1997). Learning Continuous Attractors in Recurrent Networks. *Advances in Neural Information Processing Systems*, 10:654–660.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27(July 1928):379–423.

Shriki, O., Shriki, O., Sompolinsky, H., Sompolinsky, H., Lee, D. D., and Lee, D. D. (2000). An Information Maximization Approach to Overcomplete and Recurrent Representations. *Nips*, pages 612–618.

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

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic Policy Gradient Algorithms. *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 387–395.

Smith, D. M. and Mizumori, S. J. (2006). Hippocampal place cells, context, and episodic memory. *Hippocampus*, 729:716–729.

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1-2):159–216.

Sohal, V. S. and Hasselmo, M. E. (1998). GABA(B) modulation improves sequence disambiguation in computational models of hippocampal region CA3. *Hippocampus*, 8(2):171–93.

Sompolinsky, H., Crisanti, A., and Sommers, H.-J. (1988). Chaos in random neural networks. *Physical Review Letters*, 61(3):259.

Strong, S., Koberle, R., de Ruyter van Steveninck, R., and Bialek, W. (1998). Entropy and Information in Neural Spike Trains. *Physical Review Letters*, 80(1):197–200.

Sussillo, D. (2014). Neural circuits as computational dynamical systems. *Current opinion in neurobiology*, 25C:156–163.

Sussillo, D. and Abbott, L. F. (2009). Generating coherent patterns of activity from chaotic neural networks. *Neuron*, 63(4):544–557.

Sussillo, D. and Barak, O. (2013). Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural computation*, 25(3):626–49.

Sussillo, D., Churchland, M. M., Kaufman, M. T., and Krishna, V. (2015). A neural network that finds naturalistic solutions for the production of muscle activity. 18(7).

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, number 2010.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Sutton, R. S. and Barto, A. (1998). *Reinforcement Learning*. MIT Press.

Teh, Y., Welling, M., Osindero, S., and Hinton, G. (2004). Energy-based models for sparse overcomplete representations. 4:1235–1260.

Tenenbaum, J. B., de Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–23.

Tenenbaum, J. B., Kemp, C., Griffiths, T. L., and Goodman, N. D. (2011). How to grow a mind: statistics, structure, and abstraction. *Science (New York, N.Y.)*, 331(6022):1279–1285.

Tipping, M. E. and Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622.

Tishby, N., Pereira, F. C., and Bialek, W. (2000). The information bottleneck method. *arXiv preprint physics/0004057*.

Tootoonian, S. and Lengyel, M. (2014). A Dual Algorithm for Olfactory Computation in the Locust Brain. *Nips*, (Section 5):1–9.

van Seijen, H. and Sutton, R. S. (2015). A Deeper Look at Planning as Learning from Replay. In *International Conference on Machine Learning*, volume 14, page 56 p.

van Vreeswijk, C. and Sompolinsky, H. (1996). Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science*, 274(5293):1724–6.

Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008a). Extracting and composing robust features with denoising autoencoders. *Proceedings of the 25th international conference on Machine learning - ICML '08*, pages 1096–1103.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008b). Extracting and composing robust features with denoising autoencoders. Technical report.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *Journal of Machine Learning Research*, 11(3):3371–3408.

Vinje, W. E. and Gallant, J. L. (2000). Sparse Coding and Decorrelation in Primary Visual Cortex During Natural Vision. *Science*, 287(5456):1273–1276.

Wainwright, M. J. and Jordan, M. I. (2008). Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305.

Wallenstein, G. V. and Hasselmo, M. E. (1997). GABAergic modulation of hippocampal population activity: sequence learning, place field development, and the phase precession effect. *The Journal of neuroscience*, pages 393–408.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Wood, E. R., Dudchenko, P. A., Robitsek, R. J., and Eichenbaum, H. (2000). Hippocampal neurons encode information about different types of memory episodes occurring in the same location. *Neuron*, 27(3):623–633.

Yamins, D. L. K. and Dicarlo, J. J. (2016). Using goal-driven deep learning models to understand sensory cortex. (October 2015).

Yamins, D. L. K., Hong, H., Cadieu, C. F., Solomon, E. A., Seibert, D., and DiCarlo, J. J. (2014). Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences of the United States of America*, 111(23):8619–24.

Yuste, R. (2015). From the neuron doctrine to neural networks. *Nature Reviews Neuroscience*, 16(8):487–497.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

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

Appendix A

Derivations for recurrent neural networks

This appendix extends Chapters 2, 3 and 6, showing that the generative model interpretation of autoencoders (section 2.5) and the results on mixed selectivity (section 6.2) apply to recurrent neural networks trained for prediction as well. I consider temporal structure in the data by extending each data point as a sequence with T steps, such that $\mathbf{x} = \mathbf{x}_{1:T}$.

As a reminder, a simple recurrent neural network (RNN) is described by the following equations

$$\mathbf{y}_t = \mathbf{W}_{\text{in}}\mathbf{x}_t + \mathbf{W}_{\text{rec}} \cdot \mathbf{r}_{t-1} \tag{A.1}$$

$$\mathbf{r}_t = \mathbf{s}(\mathbf{y}_t) \tag{A.2}$$

$$\mathbf{z}_t = \mathbf{W}_{\text{out}} \cdot \mathbf{r}_t \tag{A.3}$$

where $\mathbf{W}_{\text{in}}$, $\mathbf{W}_{\text{out}}$ and $\mathbf{W}_{\text{rec}}$ are input, output and recurrent weights, and $\mathbf{s}()$ is a nonlinear activation function. The bias at each layer is omitted for brevity. When used for one-step prediction $\mathbf{z}_t = \hat{\mathbf{x}}_{t+1}$, this RNN can be described using the notations of autoencoders (equation 2.14 and 2.15) as

$$\mathbf{r}_t = \mathbf{g}_\phi(\mathbf{x}_t, \mathbf{r}_{t-1}) = s(\mathbf{W}_{\text{in}} \cdot \mathbf{x}_t + \mathbf{W}_{\text{rec}} \cdot \mathbf{r}_{t-1}) \tag{A.4}$$

$$\hat{\mathbf{x}}_{t+1} = \mathbf{f}_\theta(\mathbf{r}_t) = \mathbf{W}_{\text{out}} \cdot \mathbf{r}_t \tag{A.5}$$

where the parameters of the recurrent neural network are summarised as $\theta = \{\mathbf{W}_{\text{out}}, \sigma_{\text{out}}\}$ for the generative model parameters and $\phi = \{\mathbf{W}_{\text{in}}, \mathbf{W}_{\text{rec}}\}$ for the recognition model parameters. Assuming a spherical Gaussian noise model at the output, $(\mathbf{x}_{t+1} - \hat{\mathbf{x}}_{t+1}) \sim \mathcal{N}\left(\mathbf{0}, \sigma_f^2 I\right)$, this autoencoder has a log-likelihood proportional to the negative squared prediction error plus a

constant c

$$\begin{aligned} \ln p_\theta\left(\mathbf{x}_{t+1}|\mathbf{r}_t\right) &= \ln p_\theta\left(\mathbf{z}_t = \mathbf{x}_{t+1}|\mathbf{r}_t\right) \\ &\propto -\left\|\mathbf{x}_{t+1} - \hat{\mathbf{x}}_{t+1}\right\|^2 + c \end{aligned} \tag{A.6}$$

While this chapter only proves in the case of one-step prediction, the extension to more general k-step prediction is straightforward.

A.1 Recurrent variational auto-encoders

This section proves that a variational lower-bound of the log-likelihood (equation A.9) exists and it is optimised by minimising the one-step prediction error. Using the generative and recognition model defined above, we have the following recognition and generative distribution:

$$q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t}) = \delta(\mathbf{r}_t - \mathbf{g}_\phi(\mathbf{x}_t, \mathbf{r}_{t-1})) \tag{A.7}$$

$$p_\theta(\mathbf{x}_{t+1}|\mathbf{r}_t) = \mathcal{N}\left(\mathbf{x}_{t+1}; \mathbf{f}_\theta(\mathbf{r}_t), \sigma_f^2 \mathbf{I}\right) \tag{A.8}$$

While $p_\theta(\mathbf{x}_{t+1}|\mathbf{r}_t)$ is a spherical Gaussian distribution because of the output noise model, $q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t})$ is a delta distribution given the deterministic inference model. The graphical illustrations of the generative and inference model is shown in figure A.1. As in feed-forward autoencoders, the prior of the generative model is unspecified in the RNN (section 2.6).

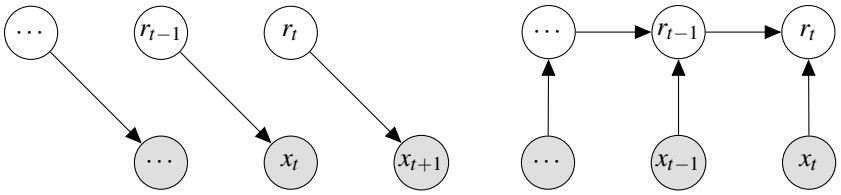


Fig. A.1 Directed graphical models of the generative model (left) and inference model (right).

Using these notations, we can write the likelihood of the generative model given a whole sequence as

$$\ln p_\theta(\mathbf{x}_{1:T}) = \ln p_x(\mathbf{x}_1) + \sum_{t=1}^{T-1} \ln p_\theta(\mathbf{x}_{t+1}|\mathbf{x}_{1:t}) \tag{A.9}$$

Using the decomposition in equation A.9 and ignoring the constant $p^*(\mathbf{x})$, the log-likelihood of the model can be written as:

$$\begin{aligned}
\ln p_\theta(\mathbf{x}_{2:T}|\mathbf{x}_1) &= \sum_{t=1}^{T-1} \ln p_\theta(\mathbf{x}_{t+1}|\mathbf{x}_{1:t}) \\
&= \sum_{t=1}^{T-1} \int q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t}) \ln \frac{p_\theta(\mathbf{x}_{t+1},\mathbf{r}_t|\mathbf{x}_{1:t})}{p_\theta(\mathbf{r}_t|\mathbf{x}_{t+1,1:t})} \mathrm{d}\mathbf{r}_t \\
&= \sum_{t=1}^{T-1} \int q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t}) \ln \frac{p_\theta(\mathbf{x}_{t+1}|\mathbf{r}_t)}{p_\theta(\mathbf{r}_t|\mathbf{x}_{t+1})} \mathrm{d}\mathbf{r}_t \\
&= \sum_{t=1}^{T-1} \underbrace{\int q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t}) \ln \frac{p_\theta(\mathbf{x}_{t+1}|\mathbf{r}_t)}{q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t})} \mathrm{d}\mathbf{r}_t}_{\mathcal{L}_t} + \mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t}) \| p_\theta(\mathbf{r}_t|\mathbf{x}_{t+1})\right]
\end{aligned} \tag{A.10}$$

where $p_\theta(\mathbf{x}_{t+1},\mathbf{r}_t|\mathbf{x}_{1:t}) = p_\theta(\mathbf{x}_{t+1},\mathbf{r}_t)$ and $p_\theta(\mathbf{r}_t|\mathbf{x}_{t+1,1:t}) = p_\theta(\mathbf{r}_t|\mathbf{x}_{t+1})$ since different steps in the generative model are independent (figure A.1). The KL-divergence suggests that the inference model uses history of inputs to match the posterior $p_\theta(\mathbf{r}_t|\mathbf{x}_{t+1})$. Therefore, although the generative model assumes independence between time steps, contextual information (here as history of inputs) can still be encoded in the representation $\mathbf{r}_t$ through the inference model. The first term, a variational lower-bound, can be further decomposed as

$$\begin{aligned}
\mathcal{L}_t &= \int q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t}) \ln \frac{p_\theta(\mathbf{x}_{t+1}|\mathbf{r}_t)}{q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t})} \mathrm{d}\mathbf{r}_t \\
&= \int q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t}) \ln \frac{p_\theta(\mathbf{x}_{t+1}|\mathbf{r}_t)p(\mathbf{r}_t)}{q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t})} \mathrm{d}\mathbf{r}_t \\
&= \int q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t}) \ln p_\theta(\mathbf{x}_{t+1}|\mathbf{r}_t) \mathrm{d}\mathbf{r}_t - \mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t}) \| p_\theta(\mathbf{r}_t)\right]
\end{aligned} \tag{A.11}$$

The first term in the above equation is proportional to the negative reconstruction error, and is minimised when optimising the prediction error (equation A.6). However, the second term, another KL-divergence is not minimised in this process, although this is required in maximising the lower-bound.

A.2 Discussions and related works

Other recent works treating recurrent neural networks as probabilistic generative models with explicit latent variables include stochastic recurrent networks (STORNs) (Chung et al., 2015) and variational RNNs(VRNN) (Bayer and Osendorfer, 2014). Both of them also

use amortised inference models to approximate the otherwise intractable posteriors. In general, they are more expressive models that required more complicated structures, while our approach provides a probabilistic explanation of RNNs without any additional structural complexity.

The most important conceptual difference from our approach is that both STORNs and VRNNs introduce additional latent variables separated from the network states, while we consider the states $\mathbf{r}_t$ themselves functions as latent variables (i.e., representations). Between STORNs and VRNNs, the major difference is that STORNs assume independent latent variables across steps (Bayer and Osendorfer, 2014) and the latent variables in VRNNs depend on all previous states (Chung et al., 2015). Separating the network states causing the observed data and network states governing the dynamics allows more flexibility, as demonstrated by their models' performance on challenging real-world tasks. On the other hand, our model is easier to analyse, and, more importantly, answers how the experimentally observed neural representations may benefit computation in a minimalist setting.

Unlike in feed-forward neural networks, where assigning the roles of priors and posteriors are unambiguous, these roles can be interpreted differently in RNNs. This can be seen from possibly different factorisations of the generative models. In Chung et al. (2015), the latent variables explicitly depend on the previous network states. Under this factorisation, the generative and recognition model need to be implemented separately. On the other hand, under our simplifying assumption of independent latent variables A.1, the same inference model also functions as a generative model when generating sequences. Although this dual-role of the inference model may sounds odd, it is formally justified in equation A.10 and A.11, where the approximate posterior distribution $q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t})$ is optimised to approximate both the posterior $p_\theta(\mathbf{x}_{t+1}|\mathbf{r}_t)$ and the prior $p_\theta(\mathbf{r}_t)$.

Note that although we assumed independence of the priors $\mathbf{r}_t$, this prior can be highly structured. As shown in section 2.6.3, the denoising training process adaptively supports such structured priors. However, the sequential order of observation is not imposed by this generative model — they are at least statistically exchangeable (Aldous, 1985). It is not uncommon to generate data from non-sequential distributions using a sequential model, such as the Chinese Restaurant Process (Aldous, 1985).

A.3 Information maximisation

Similar to section 6.1, we need to show that, as a part of optimising the lower bound objective $\mathcal{L}_t$, minimising the reconstruction error $\|\mathbf{x}_{t+1} - \hat{\mathbf{x}}_{t+1}\|^2$ or, equivalently, maximising the likelihood $\langle p_\theta(\mathbf{x}_{t+1}|\mathbf{r}_t)\rangle_{q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t})}$ maximises the marginal mutual information $\mathrm{I}_\phi[\mathbf{r}_t; \mathbf{x}_{1:t+1}]$.

As an intermediate step, we first derive the lemma as an extension of information maximisation from Barber and Agakov (2004).

Lemma 1. *Maximising* $\langle \ln p_\theta(\mathbf{x}_{t+1}|\mathbf{r}_t)\rangle_{q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t})p(\mathbf{x}_{1:t})}$ *maximises a lower-bound of the mutual information* $\mathrm{I}_\phi[\mathbf{r}_t;\mathbf{x}_{t+1}|\mathbf{x}_{1:t}]$

Proof.

$$\begin{aligned}\mathrm{I}_\phi[\mathbf{r}_t;\mathbf{x}_{t+1}|\mathbf{x}_{1:t}] &= \mathrm{H}[\mathbf{x}_{t+1}|\mathbf{x}_{1:t}] - \mathrm{H}_\phi[\mathbf{x}_{t+1}|\mathbf{r}_t,\mathbf{x}_{1:t}] \\ &= \mathrm{H}[\mathbf{x}_{t+1}|\mathbf{x}_{1:t}] + \left\langle \ln q_\phi(\mathbf{x}_{t+1}|\mathbf{r}_t,\mathbf{x}_{1:t})\right\rangle_{q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t})p(\mathbf{x}_{1:t})} \\ &\geqslant \mathrm{H}[\mathbf{x}_{t+1}|\mathbf{x}_{1:t}] + \langle \ln p_\theta(\mathbf{x}_{t+1}|\mathbf{r}_t)\rangle_{q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t})p(\mathbf{x}_{1:t})}\end{aligned} \tag{A.12}$$

The difference is

$$\begin{aligned}\mathrm{I}_\phi[\mathbf{r}_t;\mathbf{x}_{t+1}|\mathbf{x}_{1:t}] - \mathrm{H}[\mathbf{x}_{t+1}|\mathbf{x}_{1:t}] - \langle \ln p_\theta(\mathbf{x}_{t+1}|\mathbf{r}_t)\rangle_{q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t})p(\mathbf{x}_{1:t})} \\ = \left\langle \mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{x}_{t+1}|\mathbf{r}_t,\mathbf{x}_{1:t})\|p_\theta(\mathbf{x}_{t+1}|\mathbf{r}_t)\right]\right\rangle_{p(\mathbf{x}_{1:t})}\end{aligned} \tag{A.13}$$

Notice $q_\phi(\mathbf{x}_{t+1}|\mathbf{r}_t,\mathbf{x}_{1:t})$ is an (intractable) posterior of the recognition model. □

Theorem 3. *Minimising* $\langle \ln p_\theta(\mathbf{x}_{t+1}|\mathbf{r}_t)\rangle_{q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t})p(\mathbf{x}_{1:t})}$ *maximises a lower-bound of the mutual information* $\mathrm{I}_\phi[\mathbf{r}_t;\mathbf{x}_{1:t+1}]$.

Proof. Using the property of conditional mutual information, we have the following decomposition

$$\mathrm{I}_\phi[\mathbf{x}_{t+1};\mathbf{r}_t,\mathbf{x}_{1:t}] = \mathrm{I}_\phi[\mathbf{r}_t;\mathbf{x}_{t+1}|\mathbf{x}_{1:t}] + \sum_{k=1}^{t}\mathrm{I}[\mathbf{x}_{t+1};\mathbf{x}_k] \tag{A.14}$$

From the definition of mutual information,

$$\begin{aligned}\mathrm{I}_\phi[\mathbf{r}_t;\mathbf{x}_{1:t+1}] - \mathrm{I}_\phi[\mathbf{x}_{t+1};\mathbf{r}_t,\mathbf{x}_{1:t}] &= \left\langle \ln \frac{q_\phi(\mathbf{r}_t,\mathbf{x}_{1:t})}{q_\phi(\mathbf{r}_t)p(\mathbf{x}_{1:t+1})}\right\rangle_{q_\phi(\mathbf{r}_t,\mathbf{x}_{1:t})} - \left\langle \ln \frac{q_\phi(\mathbf{r}_t,\mathbf{x}_{1:t})}{p(\mathbf{x}_{t+1})q_\phi(\mathbf{r}_t,\mathbf{x}_{1:t})}\right\rangle_{q_\phi(\mathbf{r}_t,\mathbf{x}_{1:t})} \\ &= \mathrm{H}_\phi[\mathbf{r}_t] + \mathrm{H}[\mathbf{x}_{1:t+1}] - \mathrm{H}[\mathbf{x}_{t+1}] - \mathrm{H}_\phi[\mathbf{r},\mathbf{x}_{1:t}] \\ &= \mathrm{H}[\mathbf{x}_{1:t}|\mathbf{x}_{t+1}] - \mathrm{H}_\phi[\mathbf{x}_{1:t}|\mathbf{r}_t]\end{aligned} \tag{A.15}$$

Therefore, combining the above equations,

$$\begin{aligned}
\mathrm{I}_\phi\left[\mathbf{r}_t;\mathbf{x}_{1:t+1}\right] &= \mathrm{I}_\phi\left[\mathbf{x}_{t+1};\mathbf{r}_t,\mathbf{x}_{1:t}\right] + \mathrm{H}\left[\mathbf{x}_{1:t}|\mathbf{x}_{t+1}\right] - \mathrm{H}_\phi\left[\mathbf{x}_{1:t}|\mathbf{r}_t\right] \\
&= \mathrm{I}_\phi\left[\mathbf{r}_t;\mathbf{x}_{t+1}|\mathbf{x}_{1:t}\right] + \sum_{k=1}^{t}\mathrm{I}\left[\mathbf{x}_{t+1};\mathbf{x}_k\right] + \mathrm{H}\left[\mathbf{x}_{1:t}|\mathbf{x}_{t+1}\right] - \mathrm{H}_\phi\left[\mathbf{x}_{1:t}|\mathbf{r}_t\right] \\
&\geqslant \mathrm{I}_\phi\left[\mathbf{r}_t;\mathbf{x}_{t+1}|\mathbf{x}_{1:t}\right] + \sum_{k=1}^{t}\mathrm{I}\left[\mathbf{x}_{t+1};\mathbf{x}_k\right] + \mathrm{H}\left[\mathbf{x}_{1:t}|\mathbf{x}_{t+1}\right] - \mathrm{H}\left[\mathbf{x}_{1:t}\right] \\
&= \mathrm{I}_\phi\left[\mathbf{r}_t;\mathbf{x}_{t+1}|\mathbf{x}_{1:t}\right] + c
\end{aligned} \tag{A.16}$$

where $c = \sum_{k=1}^{t}\mathrm{I}\left[\mathbf{x}_{t+1};\mathbf{x}_k\right] - \mathrm{I}\left[\mathbf{x}_{t+1};\mathbf{x}_{1:t}\right]$ is a constant that depends only on the data. Applying Lemma 1 finishes the proof. □

Parallel to our analysis of feed-forward autoencoders, section B.3 shows that the above effect of maximising $\mathrm{I}_\phi\left[\mathbf{r}_t;\mathbf{x}_{1:t+1}\right]$ can be cancel by minimising the KL-divergence term $\mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t})\|p_\theta(\mathbf{r}_t)\right]$. Therefore, merely training for one-step prediction is not enough to increase mixed selectivity, which additionally requires training for robust prediction.

A.4 Training RNNs for robust prediction

Following the same derivation in section 3.3, we have the contractive regulariser for RNNs:

$$\mathcal{R} = \sum_{t=1}^{T}\sum_{k=1}^{t}\left\|\frac{\partial \mathbf{r}_t}{\partial \mathbf{x}_k}\right\|_F^2 \tag{A.17}$$

Notice that $\mathbf{r}_t$ can be written as a function of $\mathbf{x}_{1:t}$ by recursively apply equation A.4:

$$r_t(\mathbf{x}_{1:t}) = \mathbf{g}_\phi(\mathbf{x}_t, \mathbf{g}_\phi(\mathbf{x}_{t-1}, \mathbf{g}_\phi\phi(\cdots))) \tag{A.18}$$

Therefore, the same derivation in theorem 2 in section 6.2 can be used here to show that minimising the contractive regulariser $\mathcal{R}_t = \sum_{k=1}^{t}\left\|\frac{\partial \mathbf{r}_t}{\partial \mathbf{x}_k}\right\|_F^2$ minimises the mutual information $\sum_{k=1}^{t}\mathrm{I}_\phi\left[\mathbf{r}_t;\mathbf{x}_k\right]$. In practice, this can be approximated by simply training with noise-corrupted data (section 3.3). Similarly, the discussion in section 2.6 can be used to show that denoising training in RNN decreases the KL-divergence $\mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t})\|p_\theta(\mathbf{r}_t)\right]$ by implicitly imposing a GMM prior on the generative model.

A.5 Mixed selectivity

Finally, we extend our definition of mixing index, as a measure of mixed selectivity (definition 2) to RNNs on sequential data as

$$\kappa(\mathbf{r}_t, \mathbf{x}_{1:t+1}) = \mathrm{I}\phi\left[\mathbf{r}_t; \mathbf{x}_{1:t+1}\right] - \sum_{k=1}^{t+1} \mathrm{I}_\phi\left[r_{t+1}; \mathbf{x}_k\right] \tag{A.19}$$

Theorem 3 shows denoising training maximises a lower bound of the first term. Section A.4 extends theorem 2 in section 6.2, showing that denoising training as the same time minimises an upper bound of $\mathrm{I}_\phi\left[\mathbf{r}_t; \mathbf{x}_k\right]$ for $k < t+1$. Therefore, the overall effect of training on prediction error tends to increase $\kappa(\mathbf{r}_t, \mathbf{x}_{1:t+1})$, the mixed selectivity on the history of inputs (although one term, $-\mathrm{I}_\phi\left[\mathbf{r}_t; \mathbf{x}_{t+1}\right]$, in the definition of mixing index is left). Within each time step, theorem 2 in section 6.2 still applies, so the mixed selectivity on different input dimensions also tends to increase. To end this chapter, we adapt table 6.1 in chapter 6 for RNNs in table A.1.

Change	Generative models	Autoencoders	Information theory
$\Uparrow$	$\mathcal{L}_t$	$-\left\|\tilde{\mathbf{x}}_{t+1} - f_\theta(g_\phi(\tilde{\mathbf{x}}_{1:t}))\right\|^2$	$\kappa(\mathbf{r}_t, \mathbf{x}_{1:t+1})$
$\Uparrow$	$\left\langle \ln p_\theta(\mathbf{x}_{t+1}\|\mathbf{r}_t)\right\rangle_{q_\phi(\mathbf{r}_t\|\mathbf{x}_{1:t})}$	$-\left\|\mathbf{x}_{t+1} - f_\theta(g_\phi(\mathbf{x}_{1:t}))\right\|^2$	$\mathrm{I}_\phi\left[\mathbf{r}_t; \mathbf{x}_{1:t+1}\right]$
$\Downarrow$	$\mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{r}_t\|\mathbf{x}_{1:t})\|p_\theta(\mathbf{r}_t)\right]$	$\mathcal{R}$ or $\mathcal{T}$	$\sum_{k=1}^{N} \mathrm{I}_\phi\left[\mathbf{r}_t; \mathbf{x}_k\right]$

Table A.1 The effects of training RNNs for robust prediction. "Change" shows the expected change after training. $\tilde{\mathbf{x}}_t$ are noise-corrupted data $\mathbf{x}_t$. Notice that using $\tilde{\mathbf{x}}_t$ and $\mathbf{x}_t$ as targets are equivalent when the added noise has zero mean. Within each column, the item in the first row can be decomposed as the item in the second row minus the item in the third row, except that the term $-\mathrm{I}_\phi\left[\mathbf{r}_t; \mathbf{x}_{t+1}\right]$ has been left in the last column. This decomposition is exact for generative models and in the notion of information theory, and is approximated for autoencoders.

Appendix B

KL-divergence and mutual information

B.1 Information minimisation

This section show that that simply training an autoencoder as a generative model does not increase mixed-selectivity, since the effect of training the two terms in the variational lower bound (equation 2.8) may cancel each other. As we have seen in chapter 4, to maximise the variational lower bound, we need to both minimise a reconstruction error and minimise the KL-divergence $\left\langle \mathrm{D_{KL}}\left[q_\phi(\mathbf{r}|\mathbf{x})\|p_\theta(\mathbf{r})\right]\right\rangle_{p_x(\mathbf{x})}$. This KL-divergence is related to mutual information by the following theorem:

Theorem 4. *The KL-divergence* $\left\langle \mathrm{D_{KL}}\left[q_\phi(\mathbf{r}|\mathbf{x})\|p_\theta(\mathbf{r})\right]\right\rangle_{p_x(\mathbf{x})}$ *is an upper-bound of the mutual information* $\mathrm{I}_\phi[\mathbf{r};\mathbf{x}]$.

Proof. Using the definition of KL-divergence,

$$\begin{aligned}\left\langle \mathrm{D_{KL}}\left[q_\phi(\mathbf{r}|\mathbf{x})\|p_\theta(\mathbf{r})\right]\right\rangle_{p_x(\mathbf{x})} &= \int p_x(\mathbf{x})\left\{\int q_\phi(\mathbf{r}|\mathbf{x})\ln q_\phi(\mathbf{r}|\mathbf{x})\mathrm{d}\mathbf{r} - \int q_\phi(\mathbf{r}|\mathbf{x})\ln p_\theta(\mathbf{r})\,\mathrm{d}\mathbf{r}\right\}\mathrm{d}\mathbf{x}\\ &= -\mathrm{H}_\phi[\mathbf{r}|\mathbf{x}] - \int q_\phi(\mathbf{r}|\mathbf{x})\ln p_\theta(\mathbf{r})\,\mathrm{d}\mathbf{r}\\ &\geqslant -\mathrm{H}_\phi[\mathbf{r}|\mathbf{x}] - \int q_\phi(\mathbf{r})\ln q_\phi(\mathbf{r})\mathrm{d}\mathbf{r}\\ &= \mathrm{I}_\phi[\mathbf{r};\mathbf{x}]\end{aligned} \tag{B.1}$$

while the difference

$$\left\langle \mathrm{D_{KL}}\left[q_\phi(\mathbf{r}|\mathbf{x})\|p_\theta(\mathbf{r})\right]\right\rangle_{p_x(\mathbf{x})} - \mathrm{I}_\phi[\mathbf{r};\mathbf{x}] = \mathrm{D_{KL}}\left[q_\phi(\mathbf{r})\|p_\theta(\mathbf{r})\right] \tag{B.2}$$

is the KL-divergence between the marginal distribution of the representations and the prior. □

This result is intuitive in that since the prior $p_\theta(\mathbf{r})$ does not depend on any input $\mathbf{x}$, matching the prior naturally decrease the information $\mathbf{r}$ has about $\mathbf{x}$.

To study the effect of jointly training both terms in equation 2.8 in optimising $\mathcal{L}$, we re-arrange the results from theorem 1 and theorem 4:

$$\left\langle \ln p_\theta(\mathbf{x}|\mathbf{r}) \right\rangle_{q_\phi(\mathbf{r}|\mathbf{x})p_x(\mathbf{x})} = \mathrm{I}_\phi[\mathbf{r};\mathbf{x}] - \mathrm{H}[\mathbf{x}] - \left\langle \mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{x}|\mathbf{r})\|p_\theta(\mathbf{x}|\mathbf{r})\right] \right\rangle_{p_x(\mathbf{x})} \tag{B.3}$$

$$-\left\langle \mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{r}|\mathbf{x})\|p_\theta(\mathbf{r})\right] \right\rangle_{p_x(\mathbf{x})} = -\mathrm{I}_\phi[\mathbf{r};\mathbf{x}] - \mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{r})\|p_\theta(\mathbf{r})\right] \tag{B.4}$$

Comparing the above two equations, we can see that, through training, both KL-divergences decrease, while their effects on $\mathrm{I}\phi[\mathbf{r};\mathbf{x}]$ may cancel each other ($\mathrm{H}[\mathbf{x}]$ is a constant). When $\mathcal{L}$ reaches its (possibly local) maximum, $\mathrm{I}_\phi[\mathbf{r};\mathbf{x}]$ reaches a level where both $\left\langle \mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{x}|\mathbf{r})\|p_\theta(\mathbf{x}|\mathbf{r})\right] \right\rangle_{p_x(\mathbf{x})}$ and $\mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{r})\|p_\theta(\mathbf{r})\right]$ reach their minima.

B.2 Noise approximation of the marginal mutual information

The effects on the mutual information between $\mathbf{r}$ and the joint distribution of $\mathbf{x}$ can be seen from a multi-dimensional variant of the derivation in theorem 2 in section 6.2:

$$\begin{aligned} \mathrm{I}[\mathbf{r};\mathbf{x}] &\approx \mathrm{I}[\hat{\mathbf{r}};\mathbf{x}] \\ &= \mathrm{H}[\mathbf{x}] - \int p_x(\mathbf{x}) \frac{1}{2} \ln\left(\frac{2\pi e}{\det\left(\mathcal{J}(\mathbf{x})\right)}\right) \mathrm{d}\mathbf{x} \end{aligned} \tag{B.5}$$

where $\mathcal{J}(\mathbf{x})$ is the Fisher information matrix:

$$\mathcal{J}(\mathbf{x})_{i,j} = \left\langle \frac{\partial \ln p_\phi(\hat{\mathbf{r}}|\mathbf{x})}{\partial x_i} \cdot \frac{\partial \ln p_\phi(\hat{\mathbf{r}}|\mathbf{x})}{\partial x_j} \right\rangle_{\xi, p^*(\mathbf{x})} \tag{B.6}$$

which can be computed using the Gaussian approximation (equation 6.4)

$$\frac{\partial \ln p_\phi(\hat{\mathbf{r}}|\mathbf{x})}{\partial x_i} = \sum_k \frac{\xi_k}{\sigma_r} \cdot \frac{\partial r_k}{\partial x_i} \tag{B.7}$$

$$\frac{\partial \ln p_\phi(\hat{\mathbf{r}}|\mathbf{x})}{\partial x_i} \cdot \frac{\partial \ln p_\phi(\hat{\mathbf{r}}|\mathbf{x})}{\partial x_j} = \sum_{k,l} \frac{\xi_k \xi_l}{\sigma_r^2} \cdot \frac{\partial r_k}{\partial x_i} \frac{\partial r_l}{\partial x_j} \tag{B.8}$$

$$\left\langle \frac{\partial \ln p_\phi(\hat{\mathbf{r}}|\mathbf{x})}{\partial x_i} \cdot \frac{\partial \ln p_\phi(\hat{\mathbf{r}}|\mathbf{x})}{\partial x_j} \right\rangle_\xi = \sum_k \frac{\partial r_k}{\partial x_i} \frac{\partial r_k}{\partial x_j} \tag{B.9}$$

From the corollary of Jacobi's formula, we have the trace identity:

$$\begin{aligned} \mathrm{Tr}\left(\mathcal{J}\left(\mathbf{x}\right)\right) &= \ln \det \left(e^{\mathcal{J}(\mathbf{x})} \right) \\ &= \ln \det \left(\sum_{k=0}^{\infty} \frac{1}{k!} \left(\mathcal{J}\left(\mathbf{x}\right)\right)^k \right) \\ &\geqslant \ln \det \left(\mathcal{J}\left(\mathbf{x}\right)\right) \end{aligned} \tag{B.10}$$

where $e^{\mathcal{J}}\left(\mathbf{x}\right)$ is matrix exponential of the fisher information matrix. The inequality uses Minkowski determinant theorem and the fact that the determinants of positive semi-definite matrices ($\left(\mathcal{J}\left(\mathbf{x}\right)\right)^k$) are non-negative.

From equation B.9, the trace $\mathrm{Tr}\left(\mathcal{J}\left(\mathbf{x}\right)\right)$ has exactly the same form of the contractive regulariser (Bishop, 1995):

$$\mathrm{Tr}\left(\mathcal{J}\left(\mathbf{x}\right)\right) = \left\langle \sum_{k,i} \left(\frac{\partial r_k}{\partial x_i} \right)^2 \right\rangle_{p^*(\mathbf{x})} \tag{B.11}$$

Therefore, combining equation B.5 and equation B.10, we conclude that minimising the contractive regulariser is equivalent to minimising the $\mathrm{Tr}\left(\mathcal{J}\left(\mathbf{x}\right)\right)$, thus minimising an upper-bound of the mutual information $\mathrm{I}\left[\mathbf{r};\mathbf{x}\right]$. From Bishop (1995), this can be approximated by training with noise-corrupted data.

B.3 Information minimisation for RNN

Theorem 5. *The KL-divergence* $\left\langle \mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t}) \| p_\theta\left(\mathbf{r}_t\right)\right] \right\rangle_{p(\mathbf{x}_{1:t})}$ *is an upper-bound of the sum of mutual information* $\sum_{k=1}^{t} \mathrm{I}_\phi\left[\mathbf{r}_t; x_k\right]$.

Proof. Using the definition of KL-divergence,

$$\begin{aligned}
\left\langle \mathrm{D}_{\mathrm{KL}}\left[q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t}) \| p_\theta(\mathbf{r}_t)\right]\right\rangle_{p(\mathbf{x}_{1:t})} &= \int p(\mathbf{x}_{1:t}) \left\{ \int q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t}) \ln q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t}) \mathrm{d}r - \int q_\phi(\mathbf{r}_t|\mathbf{x}_{1:t}) \ln p_\theta(\mathbf{r}_t)\, \mathrm{d}\mathbf{r}_t \right\} \mathrm{d}\mathbf{x}_1 \\
&= -\mathrm{H}_\phi\left[\mathbf{r}_t|\mathbf{x}_{1:t}\right] - \int q_\phi(\mathbf{r}_t) \ln p_\theta(\mathbf{r}_t)\, \mathrm{d}\mathbf{r}_t \\
&\geqslant -\mathrm{H}_\phi\left[\mathbf{r}_t|\mathbf{x}_{1:t}\right] - \int q_\phi(\mathbf{r}_t) \ln q_\phi(\mathbf{r}_t) \mathrm{d}\mathbf{r}_t \\
&= \mathrm{I}_\phi\left[\mathbf{r}_t; \mathbf{x}_{1:t}\right]
\end{aligned} \tag{B.12}$$

□

Appendix C

Details of training and Regularised Model

The parameters used in the experiments are summarised in table C.1. Most parameters, except those directly related to training samples are the same for both denoising training (denoising) and training with regulariser (regularisation, see section 3.3).

Performance in the sequence learning task was not sensitive to moderate changes of most parameters (see also Appendix D). The connectivity (the percentage of non-zero recurrent connections) was set to 10% to reflect the sparse recurrent connection in the brain. Due to the small volume of data set, mini-batch was not necessary, and single batches containing all training examples were used. In denoising training, the training batch included 200 difference noisy samples of the trajectories. Although denoising training is usually implemented by adding different noise samples in different training iterations under large data set, this setting traded training time with (negligible amount of) memory. On the other hand, only 1 noiseless trajectory was used when employing the denoising regulariser I developed. Gradients were computed using the whole batch. The 2-norm of gradients were clipped to 2 when they were larger than 2 to avoid exploding gradients. A large momentum of 0.95 was used in combination with a weight decay rate of $1 - 10^{-3}$. The learning rate was initialised as 0.1, before decreased linearly to 0.01 in the first 200 iterations, then continued with the same value of 0.01.

The performance and neural representation of an RNN trained using the regulariser are illustrated in figure C.1, which can be compared with the same results obtained from denoising training reported in Chapter 3. After training, the changes of training losses are shown in table C.2, and the changes in mixing index κ as well as its mutual information terms are illustrated in figure C.2 and C.3. See Chapter 6 for details of their implication in the main theoretical result developed in this thesis (see also table A.1).

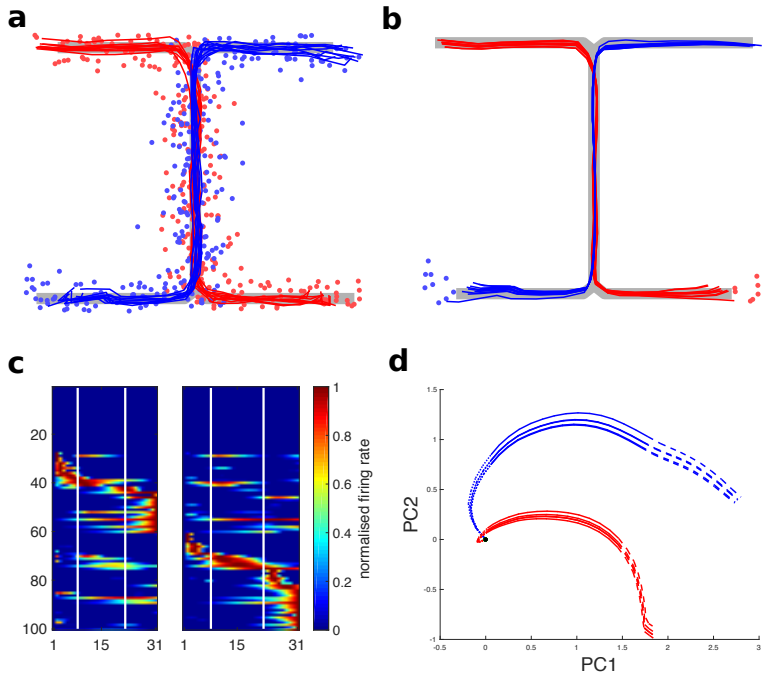


Fig. C.1 Illustration of performance and neural representation from an RNN trained using regulariser and a single instance of noiseless trajectory. Prediction and recall were still tested using input with the same noise level (0.1 or 5 cm) as in the main thesis. (**a**) and (**b**) illustrate the performance in prediction and recall respectively (see also Figure 4.1). (**c**) shows the neural representation during prediction (see also FIgure 4.3). (**d**) shows the state space trajectories of the RNN during recall projected to the first two principal components (see also Figure 4.6).

Parameter	value	
	denoising	regularisation
Number of input /output neurons	2	
Number of hidden neurons	100	
Connectivity	10%	
Gradient norm (clipped)	2	
Momentum	0.95	
Initial learning rate (first 200 steps)	0.1	
Learning rate	0.01	
Weight decay rate	$1-1\times10^{-3}$	
Iterations of gradient descent	1000	1500
Number of trials per batch	200	1
Standard deviation of additive noise	0.1 (5 cm)	0
η_1	0	0.02
η_2	0	0.01

Table C.1 Training parameters

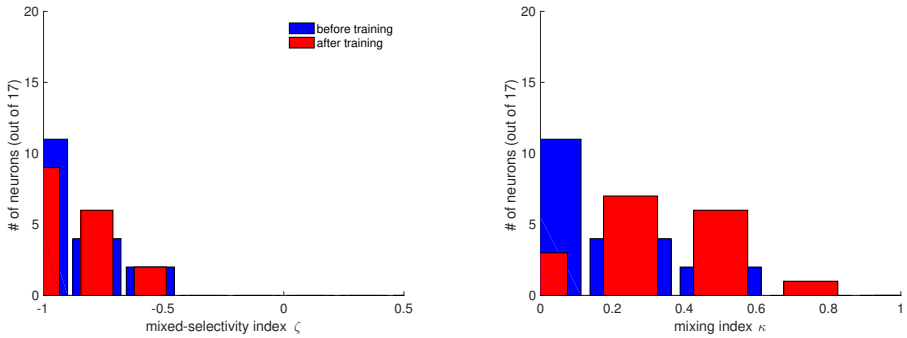


Fig. C.2 Mixed selectivity measured by both ζ and κ, for neurons fired on the central arm. The scatter plot (top) and histograms shows that overall the mixed selectivity of neurons increased after training in both measures.

	before training	after training
reconstruction loss	887.933	22.983
regularisation loss	1.697	0.002

Table C.2 Change of Losses

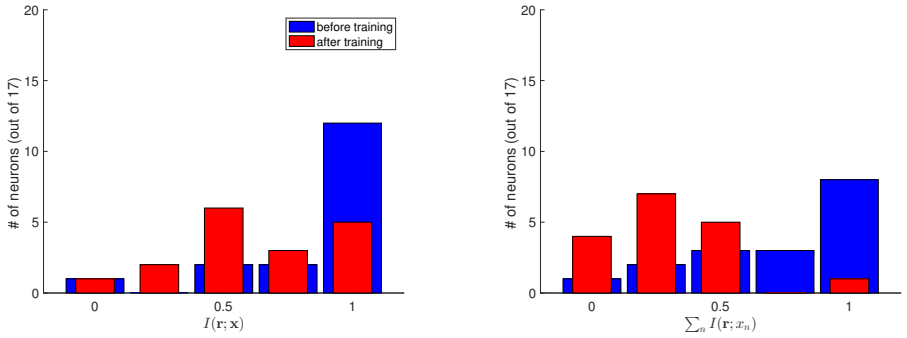


Fig. C.3 Mixing index decomposed as the joint mutual information and the sum of marginal mutual information. Both quantities decreased after training. However, their difference, which defines the mixing index κ, increased as suggested by the theory (figure C.2).

Appendix D

Tests of robustness

The results reported in this thesis were build on the competent performance of RNNs on the T-maze alternation task (Figure 1.2). To test whether such performance is robust and, how the performance depends on the particular settings (section 3.2 and Appendix C), here I test the performance of RNNs on a number of different configurations of both the model and the task.

D.1 Variants of the T-maze alternation task

This section looks at whether the same RNNs are competent in variants of the T-maze alternation task as well. First, I assess whether the RNNs trained under only a certain noise level are robust to noise at various other levels. Such generalisation is necessary for animals in changing environments. For this, I test the RNNs training in Chapter 3 with inputs at different noise levels. Figure D.1 shows that the RMSEs only grew approximately linearly when the testing noise level is higher than that in training.

Next, I altered the way steps on the trajectories are sampled. The first variant changes the total number of steps sampled along the same trajectories. The steps are still evenly but more densely sampled (with less space between two steps), so the sequence length increases. Figure D.2 (**a**) shows the RMSE indeed increased as the sequences became longer — although prediction is little affected, recall was significantly worse with longer sequences. Recall requires memorising the entire sequences, which became harder with longer sequences, while prediction can use mechanisms require relatively fewer memory, such as filtering. Interestingly, the performance also decreases when the number of steps were too few (< 15). This is because consecutive steps became less correlated and less information could be accumulated along the trails. However, the phenomenon observed in Figure D.2 (**a**) gives rise to the questions of whether there exists an optimal sequence length

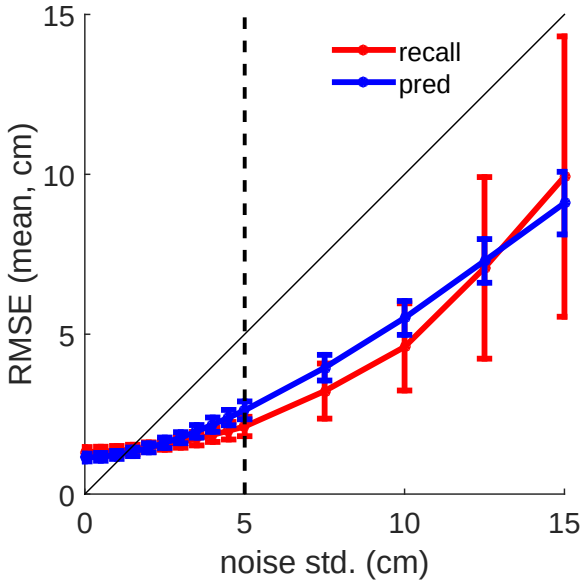


Fig. D.1 The same RNNs' performance at different noise levels. 10 RNNs are trained under input noise with the standard deviation of 5cm (dashed horizontal line). They are tested under input noise with standard deviations up to 15cm. For all the figures in this appendix, blue and red colour codes prediction and recall tasks respectively. Numbers were computed using 100 testing trials on each of the 10 RNNs.

and how can RNNs deal with very long sequences. These questions will be the topics for future investigation, and I expect hierarchical representations can be used to maintain the actually length of sequences processed by the brain at a manageable scale.

The second way to modify the steps keeps the same number of steps but changes the spaces between steps. the steps are no longer evenly sampled, corresponding to the more realistic assumption of traversing the T-maze with non-constant speed. This is realised by sample the step sizes s (i.e., speeds) from a first order auto-regressive (AR1) process:

$$s_t = c + \varphi \cdot s_{t-1} + \sigma_\varepsilon \cdot \varepsilon_t \tag{D.1}$$

where ε_t is a white noise process sampled from a standard normal distribution. I set the parameters c, φ and σ_ε so that s_t has a mean μ_s the same as the (constant) speed of the original task, and standard deviations σ_s varying between 0 (corresponding to constant speed) and 0.5μ. This can be done using the relationships $\mu_s = \frac{c}{1-\varphi}$ and $\sigma_s^2 = \frac{\sigma_\varepsilon}{1-\varphi^2}$. Sequences with a higher σ_s is, by construction, less predictable. Figure D.2 (**b**, solid lines) shows the RMSE of both prediction and recall indeed scaled with this standard deviation. However, when the error was measured as between the output location and the nearest location on

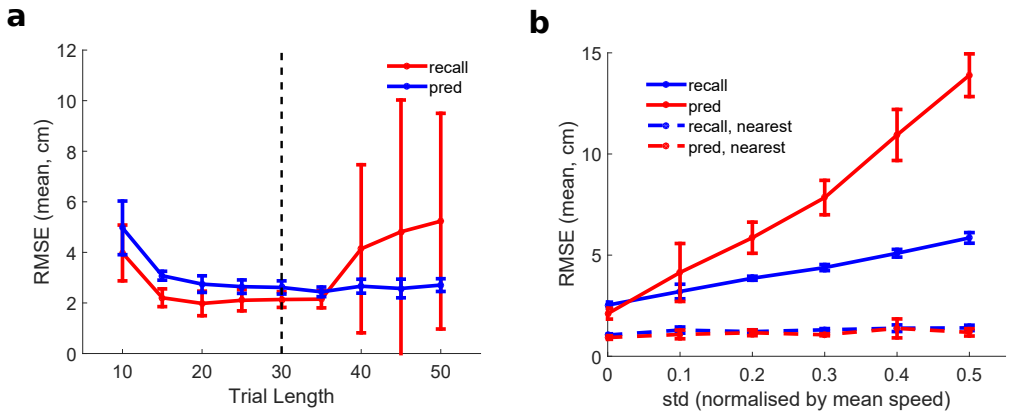


Fig. D.2 RNNs' performance when the steps along the trajectories were sampled differently. (**a**) shows the RMSEs of prediction and recall when the trajectories were sampled evenly but with different total number of steps, which varied the lengths of the sequences. (**b**) shows the RMSEs when keeping the number of steps the same (30), but allowed variances of step size measured by their standard deviations. Numbers were computed using 100 testing trials on each of the 10 RNNs.

the maze — which is fairer since the exact step sizes are unpredictable by definition — the RMSEs stayed almost the same despite the increased σ_s (dashed lines).

Finally, I tested more dramatic variants of the T-maze task. To characterise the essential structure of the T-maze alternation task, we can notice that, in the task presented in Chapter 2 (Figure 4.1), the central arm alone occupies one dimension in the 2-dimensional space, and the incoming and out-going arms always lay in different sides of the other dimension. We can thus generalise the task with additional dimensions while keeping the essential structure. For example, by allowing the third dimension (perpendicular to this paper), the task can be generalised by allowing the incoming and outgoing arms to *rotate* into this third dimension, as long as they do not end in the same side of any dimension for the same trial; as a result, 4, instead of 2 sequences can be squeezed into this 3-dimensional space (the incoming arm can be in any of the 4 quadrant, in the 2 dimensional space without the central arm, and each of the outgoing arms simply needs to avoid the same quadrant). Generally, in an N-dimensional space, there are $(N-1)\times 2$ valid sequences that can be obtained from such rotation. Figure D.3 (**a**) shows the performance of prediction and recall in a 5-dimensional space as a function of the number of sequences (all other parameters were the same, except additional input and output units to allow the extra dimensions). The RMSEs only scaled slowly with increased number of sequences, which suggests the RNNs captures the structure of the task, which changed little despite the increase of dimensions and number of sequences.

As a control, I tested the performance of RNNs for completely random sequences in the same 5-dimensional space. Each step in these sequences were sampled completely randomly

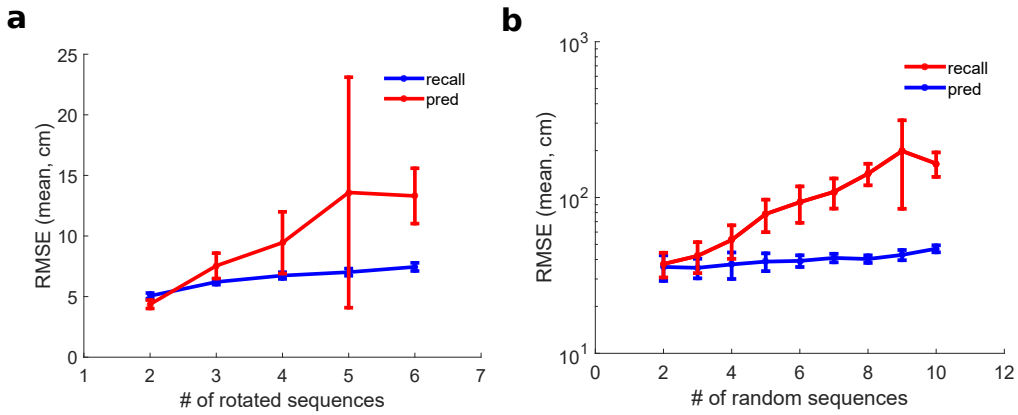


Fig. D.3 RNNs' performance on a high-dimensional T-maze task and random sequence learning. (**a**) shows the performance on a generalised T-maze in a 5-dimensional space. In contrast, (**b**) shows the results from learning unstructured random sequences. Numbers were computed using 100 testing trials on each of the 10 RNNs.

and independently from a standard normal distribution, so there was no structure at all. For the number of sequences I tested (up to 10), the overlapping between sequences was small. Figure D.3 (**b**) shows that, unlike learning the structured (**a**), testing errors increased dramatically, especially for recall, for random sequences — the amount of information to be stored in the RNNs was much larger for these unstructured sequences.

D.2 RNN configurations

In addition to changing the task settings, as discussed in the previous section, I also tested the dependencies of task performance on configurations of the RNNs. This section focuses on two parameters that are of particular interests for both neuroscientists and engineers: the connectivity (the percentage of inter-neuron connections in the hidden layer), and the number of hidden neurons. In the brain, both of these parameters are physically constrained in a neural population. In particular, the all-to-all connection scheme that is usually assumed in computational models is implausible in even the most densely connected CA3 region of the hippocampus. In practise, dense recurrent connections bring extra communication burden in distributed implementations.

Figure D.4 (**a**) shows the performance as a function of connectivity. The performance of recall was severely damaged when the connectivity was below 5%. For the sequence learning task we tested, merely about 7% of connectivity, slightly below the 10% level I used in the main thesis, was enough to support a competent level of performance. Interestingly, the performance started to decrease when the connectivity went beyond 40%, implying the RNNs

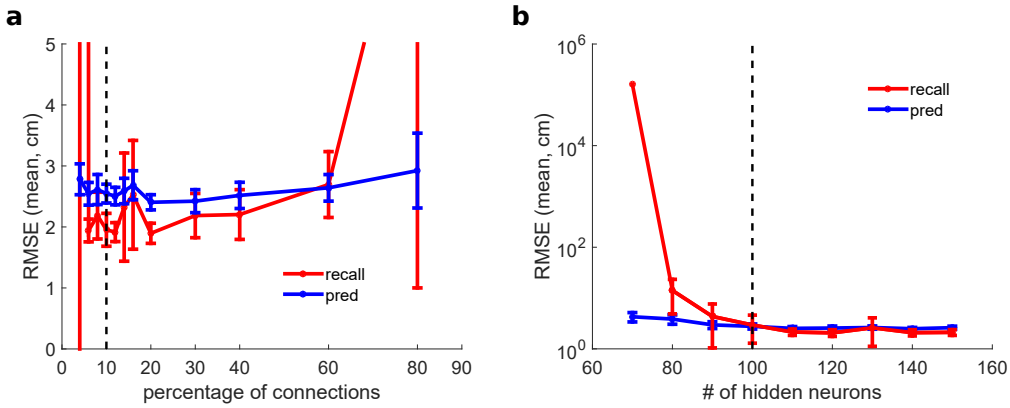


Fig. D.4 Performance of prediction and recall as a function of connectivity (the percentage of recurrent connections between hidden neurons) and the number of hidden neurons. The vertical dashed lines indicate the settings used in the main thesis. Numbers were computed using 100 testing trials on each of the 10 RNNs.

were under-fitted with the extra parameters while keeping the same training iterations. This result suggests an appropriate level of connectivity may strike a balance between capacity and training time. On the other hand, Figure D.4 (**b**) shows the performance as a function of the number of hidden neurons, while keeping the connectivity at 10%. Again, we can see the capacity of the RNNs were largely determined by the number of hidden neurons, and the performance of recall, which requires storing information about full sequences, rapidly deteriorated when the number of hidden neurons shrunk below sufficient. However, unlike extra connectivity, extra hidden neurons did not introduce the problem of under-fitting, despite the increase of parameter number.

D.3 Conclusion

In conclusion, the performance of RNNs reported in the main thesis (Chapter 3) was robust when varying a number of task or model related parameters. The RNN was able to discover the underlying structure of the task despite changes of low-level features such as the speed and the dimensionality. This ability of abstraction is helpful for adaptive behaviours in ever-changing environments. In addition, I presented a intriguing phenomenon when testing changing the connectivity and number of hidden neurons. While both modifications change the number of parameters, increasing the number of hidden neurons did not introduce under-fitting as increasing the connectivity — this might be why the brain have so many neurons but relatively sparse recurrent connections.

Index